

赢在项目开发



- ◎ 超值**DVD**教学光盘，送视频讲解、开发源码
- ◎ 增值的网络资源库，送大量论文资料、电子书

精选项目开发案例，深入浅出地讲解，带领您快速走进编程世界，成为职场达人。

# C# 项目开发实战密码

扶松柏 / 编著



清华大学出版社





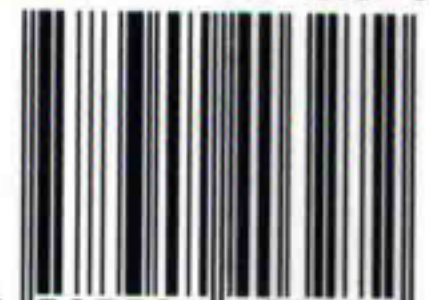
# C# 项目开发实战密码

赢在项目开发

清华大学出版社数字出版网站

WQBook 书文局泉  
www.wqbook.com

ISBN 978-7-302-40327-2



9 787302 403272 >  
定价: 69.80元



赢在项目开发

# C#项目开发实战密码

扶松柏 编著

清华大学出版社  
北 京



## 内 容 简 介

C#是当今使用最为频繁的编程语言之一，一直在开发领域中占据重要的地位。本书通过 12 个综合案例的实现过程，详细讲解 C#在实践项目中的综合运用过程，这些项目从作者的学生时代写起，到项目经理结束，一直贯穿于作者最重要的开发时期。

第 1 章讲解一个俄罗斯方块游戏的具体实现流程；第 2 章讲解多媒体学习社区系统的具体实现流程；第 3 章讲解大东科技人事管理系统的具体实现流程；第 4 章讲解在线留言簿系统的具体实现流程；第 5 章讲解浪漫满屋通信录系统的具体实现流程；第 6 章讲解在线点歌系统的具体实现流程；第 7 章讲解在线商城系统的具体实现流程；第 8 章讲解一个企业交互系统的具体实现流程；第 9 章讲解一个餐饮管理系统的具体实现流程；第 10 章讲解一个短信群发系统的具体实现流程。第 11 章讲解超市进销存系统的具体实现流程；第 12 章讲解家庭视频监控系统的实现流程。

在具体讲解每个实例时，都遵循项目的进度来展开，从接到项目到具体开发，直到最后的调试和发布。内容循序渐进，并穿插学习技巧和职场生存法则，引领读者全面掌握 C#。

本书不但适合 C#初学者阅读，也可供有一定 C#基础的读者学习，亦可作为有一定造诣的程序员参考书。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

### 图书在版编目(CIP)数据

C#项目开发实战密码/扶松柏编著. —北京：清华大学出版社，2015  
(赢在项目开发)

ISBN 978-7-302-40327-2

I. ①C… II. ①扶… III. ①C 语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2015)第 113522 号

责任编辑：魏 莹 宋延清

封面设计：杨玉兰

责任校对：马素伟

责任印制：宋 林

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](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈：010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

课 件 下 载：<http://www.tup.com.cn>, 010-62791865

印 刷 者：清华大学印刷厂

装 订 者：三河市溧源装订厂

经 销：全国新华书店

开 本：185mm×260mm

印 张：35.5

字 数：860 千字

(附 DVD1 张)

版 次：2015 年 8 月第 1 版

印 次：2015 年 8 月第 1 次印刷

印 数：1~3000

定 价：69.80 元

产品编号：061319-01



# 前言

## C#语言的重要性

C#作为微软在 21 世纪推出的新语言，有着其他语言无法比拟的优势。C#编程语言的应用非常广泛，在软件行业的多个应用领域中，已成为基于 .NET Framework 解决方案的首选语言。C#作为一门新的程序设计语言，集中了 C、C++ 和 Java 等语言的优点，是一门现代的、优越的、具有广阔发展前景的程序设计语言。

## 本书内容

本书共分 12 章，各章的内容如下：

从菜鸟到高手，从学生到系统架构师，详细记录了作者在项目开发过程中如鱼得水的经历，传授了赢在项目开发的秘籍。

第 1 章 介绍俄罗斯方块游戏的运行流程，并讲解其具体的实现过程。

赢在规划，做好职业规划和项目分析

第 2 章 介绍多媒体学习社区的运行流程，并讲解其具体的实现过程。

赢在自身，快速提升自身的开发素质

第 3 章 介绍大东科技人事管理系统的运作流程，并讲解其具体的实现过程。

赢在职场，讲解程序员职场修炼秘籍

第 4 章 介绍在线留言簿系统的运行流程，并讲解其具体的实现过程。

赢在公司，探讨部门的沟通之道

第 5 章 介绍如何开发浪漫满屋通信录系统，展示 C# 在 WPF 项目中的功能和技巧。

赢在代码，体现程序开发之美

第 6 章 介绍如何创建一个在线点歌系统，讲解 C# 在桌面项目中的巨大优势。

赢在构思，让程序具有更好的可伸缩性

第 7 章 介绍在线商城系统的开发流程，讲解电商系统的具体实现过程。

赢在 OO，实现高内聚和低耦合

第 8 章 介绍开发一个企业交互系统的过程，讲解企业办公类软件的构建和实现过程。

赢在技术，通过可移植性实现跨平台

第 9 章 介绍开发餐饮管理系统的流程，讲解其具体实现过程，并剖析技术核心和实现技巧。

赢在质量，提高程序的健壮性

第 10 章 介绍开发短信群发系统的方法，讲解 C# 在连接硬件资源领域中的作用。

赢在管理，运转一个健步如飞的团队

第 11 章 介绍开发超市进销存系统的流程，讲解其具体实现过程，并剖析技术核心和实现技巧。

赢在沉淀，使用计算机中的算法技术

第 12 章 介绍家庭视频监控系统的构建方法，讲解使用 C# 开发流媒体软件的方法。

赢在设计，打造一个美丽的架构



## 读者服务

为方便读者解决学习过程中遇到的疑难问题，本书的编写团队特为广大读者提供了丰富的学习资源：

- 配书光盘——书中各开发项目的源代码和语音视频讲解。
- 网络下载资源——配套各章学习的电子书以及海量论文资料。

我们还特别开通了读者学习 QQ 群，群号是 105621466，欢迎广大读者加入本群，一起讨论并分享学习开发过程中的点点滴滴。

## 致谢

本书的主要编写人员有扶松柏、陈强、李佐彬、李淑芳、蒋凯、王梦、王书鹏、张子言、张建敏、陈德春、李藏、关立勋、秦雪薇、薛多鸯、李强、刘海洋、唐凯、吴善财、王石磊、习国庆、张家春、杨靖宇、王东华、罗红仙、曹文龙、胡郁、孙宇、于洋、李冬艳、代林峰、谭贞军、张玲玲、朱桂英、徐璐、徐娜子。

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

感谢您购买本书，希望本书能成为您编程路上的领航者。祝您读书快乐！

编 者

# 目 录

|                                       |  |
|---------------------------------------|--|
| 第 1 章 俄罗斯方块游戏.....1                   |  |
| 1.1 赢在规划.....2                        |  |
| 1.1.1 赢在起点——程序员的职业<br>规划.....2        |  |
| 1.1.2 赢在项目开发伊始——做好<br>项目分析.....2      |  |
| 1.2 第一个项目.....4                       |  |
| 1.3 功能描述.....4                        |  |
| 1.3.1 功能模块分析.....6                    |  |
| 1.3.2 游戏的模块结构.....6                   |  |
| 1.3.3 游戏的运行流程.....6                   |  |
| 1.4 搭建开发平台.....7                      |  |
| 1.4.1 安装 Visual Studio 2013.....7     |  |
| 1.4.2 规划项目文件.....9                    |  |
| 1.5 界面设计.....10                       |  |
| 1.5.1 制作游戏窗体.....11                   |  |
| 1.5.2 窗体元素设置文件.....14                 |  |
| 1.6 具体编码.....18                       |  |
| 1.6.1 事件处理程序.....18                   |  |
| 1.6.2 游戏控制、处理方法.....21                |  |
| 1.7 测试运行.....36                       |  |
| 第 2 章 多媒体学习社区.....37                  |  |
| 2.1 修炼自身.....38                       |  |
| 2.1.1 “码农”和“高大上”.....38               |  |
| 2.1.2 赢在自身——快速提升自身<br>修养.....38       |  |
| 2.2 开发背景简介.....40                     |  |
| 2.3 系统设计分析.....40                     |  |
| 2.3.1 互动媒体学习社区的优势.....40              |  |
| 2.3.2 系统的特点.....41                    |  |
| 2.3.3 系统目标.....41                     |  |
| 2.3.4 确定设计方案——B/S 体系<br>结构.....42     |  |
| 2.4 需求分析.....42                       |  |
| 2.4.1 可行性分析.....43                    |  |
| 2.4.2 功能分析.....43                     |  |
| 2.4.3 业务流程.....44                     |  |
| 2.5 总体设计.....44                       |  |
| 2.6 系统预览.....46                       |  |
| 2.7 构建开发环境.....47                     |  |
| 2.8 数据库设计.....48                      |  |
| 2.8.1 概念设计.....48                     |  |
| 2.8.2 实体 E-R 图.....49                 |  |
| 2.8.3 逻辑设计.....50                     |  |
| 2.9 设计文件夹组织结构和功能模块.....53             |  |
| 2.9.1 文件组织结构的设计.....53                |  |
| 2.9.2 用户功能模块设计.....54                 |  |
| 2.10 公共类的设计.....57                    |  |
| 2.10.1 数据库操作类的设计.....57               |  |
| 2.10.2 业务功能类设计.....60                 |  |
| 2.11 网站首页设计.....62                    |  |
| 2.12 实现用户注册模块.....63                  |  |
| 2.12.1 login.aspx 页面部分代码<br>分析.....64 |  |
| 2.12.2 用户登录设计.....64                  |  |
| 2.13 发布并管理教程.....66                   |  |
| 2.13.1 发布教程.....68                    |  |
| 2.13.2 查看教程页设计.....70                 |  |
| 2.14 后台管理页面的设计.....71                 |  |
| 2.14.1 实现用户管理的页面.....72               |  |
| 2.14.2 视频管理模块设计.....73                |  |
| 2.15 系统测试.....74                      |  |
| 第 3 章 大东科技人事管理系统.....77               |  |
| 3.1 程序员职场生存必杀技.....78                 |  |
| 3.1.1 程序员的生存现状.....78                 |  |
| 3.1.2 中外 IT 领域的企业文化.....78            |  |
| 3.1.3 赢在职场——修炼程序员<br>职场秘籍.....79      |  |

|                                  |            |                                 |            |
|----------------------------------|------------|---------------------------------|------------|
| 3.2 系统介绍.....                    | 81         | 4.4 规划系统文件.....                 | 144        |
| 3.2.1 系统背景介绍.....                | 82         | 4.4.1 规划文件.....                 | 144        |
| 3.2.2 应用的目的与意义.....              | 82         | 4.4.2 选择开发工具.....               | 145        |
| 3.2.3 人事管理系统的发展趋势.....           | 83         | 4.5 设计数据库.....                  | 145        |
| 3.3 系统需求分析.....                  | 83         | 4.5.1 后台数据库及数据库访问<br>接口的选择..... | 145        |
| 3.4 系统设计.....                    | 85         | 4.5.2 数据库结构的设计.....             | 145        |
| 3.4.1 系统设计目标.....                | 85         | 4.6 系统配置和数据库访问层.....            | 146        |
| 3.4.2 系统功能设计.....                | 86         | 4.6.1 系统配置.....                 | 146        |
| 3.5 数据库设计.....                   | 89         | 4.6.2 数据库访问层设计.....             | 149        |
| 3.5.1 数据库描述.....                 | 89         | 4.7 具体编码.....                   | 155        |
| 3.5.2 数据库分析.....                 | 89         | 4.7.1 留言数据显示.....               | 155        |
| 3.5.3 数据库概念设计.....               | 89         | 4.7.2 留言分页列表显示模块.....           | 159        |
| 3.6 实现公共类.....                   | 94         | 4.7.3 留言回复模块.....               | 162        |
| 3.6.1 实现 MyMeans 公共类.....        | 94         | 4.7.4 添加留言信息模块.....             | 164        |
| 3.6.2 实现 MyModule 公共类.....       | 96         | 4.7.5 留言管理模块.....               | 165        |
| 3.7 实现用户登录模块.....                | 104        | 4.8 测试运行.....                   | 168        |
| 3.7.1 登录模块技术分析.....              | 104        |                                 |            |
| 3.7.2 具体实现.....                  | 105        | <b>第 5 章 浪漫满屋通信录系统.....</b>     | <b>171</b> |
| 3.8 主窗体详细设计.....                 | 106        | 5.1 体验语言之美.....                 | 172        |
| 3.8.1 主窗体技术分析.....               | 107        | 5.1.1 程序员经常忽视的问题.....           | 172        |
| 3.8.2 具体实现.....                  | 108        | 5.1.2 赢在程序自身——体现<br>代码之美.....   | 172        |
| 3.9 实现人事档案浏览模块.....              | 112        | 5.2 新的项目.....                   | 173        |
| 3.10 实现人事资料查询模块.....             | 133        | 5.2.1 系统分析.....                 | 174        |
| 3.10.1 人事资料查询窗体的技术<br>分析.....    | 133        | 5.2.2 系统目标.....                 | 174        |
| 3.10.2 具体实现.....                 | 133        | 5.3 功能模块划分.....                 | 175        |
| <b>第 4 章 在线留言簿系统.....</b>        | <b>139</b> | 5.4 设计窗体.....                   | 176        |
| 4.1 企业沟通之道.....                  | 140        | 5.4.1 设置和启动应用程序.....            | 176        |
| 4.1.1 开发公司的部门现状.....             | 140        | 5.4.2 设计用户界面.....               | 178        |
| 4.1.2 赢在职场——探讨部门之间的<br>沟通之道..... | 141        | 5.4.3 实现三维动画效果.....             | 182        |
| 4.2 第一个盈利项目.....                 | 142        | 5.4.4 遍历窗体可视化树.....             | 183        |
| 4.2.1 组建团队.....                  | 142        | 5.4.5 添加联系人.....                | 184        |
| 4.2.2 系统规划.....                  | 142        | 5.4.6 实现多媒体.....                | 188        |
| 4.3 系统概述和总体设计.....               | 143        | 5.4.7 添加图片.....                 | 189        |
| 4.3.1 在线留言簿模块的功能<br>原理.....      | 143        | 5.4.8 保存联系人资料.....              | 190        |
| 4.3.2 在线留言簿系统的构成<br>模块.....      | 144        | 5.5 系统测试.....                   | 193        |
|                                  |            | <b>第 6 章 在线点歌系统.....</b>        | <b>197</b> |
|                                  |            | 6.1 架构中的可扩展性.....               | 198        |



|        |                                        |     |
|--------|----------------------------------------|-----|
| 6.1.1  | 软件的发展是一个不断完善的过程.....                   | 198 |
| 6.1.2  | 赢在项目本身——让程序具有更好的可扩展性.....              | 198 |
| 6.2    | 系统分析.....                              | 199 |
| 6.2.1  | 背景介绍.....                              | 199 |
| 6.2.2  | 需求分析.....                              | 200 |
| 6.2.3  | 可行性分析.....                             | 200 |
| 6.2.4  | 编写项目计划书.....                           | 201 |
| 6.3    | 系统模块划分.....                            | 203 |
| 6.4    | 设计数据库.....                             | 203 |
| 6.4.1  | 数据库概念结构设计.....                         | 204 |
| 6.4.2  | 数据库逻辑结构的设计.....                        | 206 |
| 6.5    | 设计公共类.....                             | 207 |
| 6.5.1  | 数据库连接.....                             | 207 |
| 6.5.2  | 歌曲信息参数.....                            | 208 |
| 6.5.3  | 歌曲信息操作处理.....                          | 209 |
| 6.6    | 设计窗体.....                              | 213 |
| 6.7    | 具体编码工作.....                            | 217 |
| 6.7.1  | 登录验证模块.....                            | 217 |
| 6.7.2  | 后台维护模块.....                            | 218 |
| 6.7.3  | 明星管理模块.....                            | 221 |
| 6.7.4  | 系统点歌模块.....                            | 226 |
| 6.7.5  | 歌曲信息模块.....                            | 228 |
| 6.7.6  | 播放歌曲模块.....                            | 232 |
| 6.8    | 项目调试.....                              | 234 |
| 6.9    | 系统升级.....                              | 236 |
| 6.9.1  | 功能升级——升级前的思考.....                      | 236 |
| 6.9.2  | 增加维护歌曲信息模块.....                        | 237 |
| 6.9.3  | 增加维护明星信息模块.....                        | 239 |
| 6.9.4  | 以“人性化”为目标的功能升级.....                    | 240 |
| 6.10   | 数据库工具升级.....                           | 245 |
| 6.10.1 | 导入数据.....                              | 245 |
| 6.10.2 | 修改连接程序.....                            | 246 |
| 6.10.3 | 用 OleDbConnection 对象连接 OLE DB 数据源..... | 248 |

|        |                            |     |
|--------|----------------------------|-----|
| 第 7 章  | 在线商城系统.....                | 249 |
| 7.1    | 模块化编程.....                 | 250 |
| 7.1.1  | 谈模块化设计思想.....              | 250 |
| 7.1.2  | 赢在模块化思想——实现高内聚和低耦合的代码..... | 251 |
| 7.2    | 新的项目.....                  | 253 |
| 7.3    | 项目规划分析.....                | 254 |
| 7.4    | 规划项目文件.....                | 254 |
| 7.5    | 系统配置文件.....                | 255 |
| 7.6    | 搭建数据库.....                 | 256 |
| 7.6.1  | 数据库的设计.....                | 257 |
| 7.6.2  | 设置系统参数.....                | 259 |
| 7.7    | 数据访问层.....                 | 260 |
| 7.7.1  | 商品显示.....                  | 260 |
| 7.7.2  | 订单处理.....                  | 265 |
| 7.7.3  | 商品评论.....                  | 272 |
| 7.7.4  | 商品分类.....                  | 275 |
| 7.7.5  | 商品管理.....                  | 285 |
| 7.8    | 显示商品.....                  | 287 |
| 7.8.1  | 主框架页.....                  | 288 |
| 7.8.2  | 顶部导航页面.....                | 288 |
| 7.8.3  | 左侧导航——分类列表页面.....          | 289 |
| 7.8.4  | 右侧导航——商品列表页面.....          | 289 |
| 7.8.5  | 按被点击次数显示.....              | 290 |
| 7.8.6  | 按商品名称显示模块.....             | 292 |
| 7.8.7  | 商品详情页面.....                | 293 |
| 7.9    | 商品分类处理.....                | 295 |
| 7.9.1  | 设置分类的层次结构.....             | 296 |
| 7.9.2  | 添加分类模块.....                | 298 |
| 7.9.3  | 分类修改模块.....                | 299 |
| 7.9.4  | 分类管理模块.....                | 300 |
| 7.10   | 商品管理.....                  | 302 |
| 7.10.1 | 商品添加模块.....                | 302 |
| 7.10.2 | 商品修改模块.....                | 303 |
| 7.10.3 | 商品管理列表模块.....              | 305 |
| 7.10.4 | 商品图片修改模块.....              | 306 |
| 7.11   | 购物车.....                   | 310 |

|        |                 |     |
|--------|-----------------|-----|
| 7.11.1 | 购物车组件的设计 .....  | 310 |
| 7.11.2 | 购物车商品添加模块 ..... | 315 |
| 7.11.3 | 购物车管理 .....     | 316 |
| 7.12   | 订单处理 .....      | 319 |
| 7.12.1 | 生成订单编号 .....    | 319 |
| 7.12.2 | 提交并创建新订单 .....  | 320 |
| 7.12.3 | 查看订单详情 .....    | 322 |
| 7.12.4 | 订单列表模块 .....    | 323 |
| 7.12.5 | 订单状态处理模块 .....  | 324 |
| 7.13   | 商品评论模块 .....    | 326 |
| 7.13.1 | 评论显示模块 .....    | 327 |
| 7.13.2 | 评论管理模块 .....    | 327 |
| 7.14   | 商品搜索模块 .....    | 329 |
| 7.15   | 项目调试 .....      | 330 |

## 第8章 企业交互系统 .....

|       |                            |     |
|-------|----------------------------|-----|
| 8.1   | 程序的可移植性 .....              | 334 |
| 8.1.1 | 什么是程序的可移植性 .....           | 334 |
| 8.1.2 | 赢在项目——实现跨开发<br>平台的转换 ..... | 334 |
| 8.2   | 新的挑战 .....                 | 335 |
| 8.3   | 项目规划和分析 .....              | 336 |
| 8.3.1 | 在线交互系统的背景 .....            | 336 |
| 8.3.2 | 企业在线交互系统的构成<br>模块 .....    | 336 |
| 8.4   | 规划项目文件 .....               | 338 |
| 8.5   | 系统配置文件 .....               | 338 |
| 8.6   | 搭建数据库 .....                | 339 |
| 8.6.1 | 数据库设计 .....                | 340 |
| 8.6.2 | 系统参数设置文件 .....             | 343 |
| 8.7   | 数据访问层 .....                | 344 |
| 8.7.1 | 数据访问层——用户登录<br>验证 .....    | 344 |
| 8.7.2 | 数据访问层——客户分组 .....          | 348 |
| 8.7.3 | 数据访问层——团队模块 .....          | 353 |
| 8.8   | 实现身份验证模块 .....             | 357 |
| 8.8.1 | 用户登录验证模块 .....             | 357 |
| 8.8.2 | 登录用户注销模块 .....             | 359 |
| 8.9   | 客户分组处理模块 .....             | 359 |

|        |                 |     |
|--------|-----------------|-----|
| 8.9.1  | 用户分组添加模块 .....  | 360 |
| 8.9.2  | 用户分组修改模块 .....  | 360 |
| 8.9.3  | 用户组管理列表模块 ..... | 362 |
| 8.9.4  | 客户检索模块 .....    | 363 |
| 8.9.5  | 客户管理列表模块 .....  | 366 |
| 8.9.6  | 客户移动转换模块 .....  | 367 |
| 8.9.7  | 客户信息显示模块 .....  | 369 |
| 8.10   | 系统团队处理模块 .....  | 370 |
| 8.10.1 | 添加团队模块 .....    | 370 |
| 8.10.2 | 修改团队处理模块 .....  | 371 |
| 8.10.3 | 团队管理列表模块 .....  | 372 |
| 8.10.4 | 加入团队处理模块 .....  | 374 |
| 8.11   | 在线交互模块 .....    | 375 |
| 8.11.1 | 系统主页显示模块 .....  | 376 |
| 8.11.2 | 一对一交互处理模块 ..... | 377 |
| 8.11.3 | 团队交互处理模块 .....  | 379 |
| 8.11.4 | 文件发送模块 .....    | 383 |
| 8.12   | 项目调试 .....      | 388 |

## 第9章 餐饮管理系统 .....

|       |                          |     |
|-------|--------------------------|-----|
| 9.1   | 考虑所有可能会发生的情形 .....       | 392 |
| 9.1.1 | 一段代码所引发的思考 .....         | 392 |
| 9.1.2 | 赢在项目——提高程序的<br>健壮性 ..... | 392 |
| 9.2   | 新的项目 .....               | 394 |
| 9.3   | 项目规划分析 .....             | 395 |
| 9.3.1 | 开发背景 .....               | 395 |
| 9.3.2 | 项目模块分析 .....             | 395 |
| 9.3.3 | 构成模块 .....               | 395 |
| 9.4   | 搭建数据库 .....              | 396 |
| 9.4.1 | 数据库概念设计 .....            | 396 |
| 9.4.2 | 数据库逻辑结构设计 .....          | 398 |
| 9.5   | 设计窗体 .....               | 400 |
| 9.6   | 具体编码 .....               | 402 |
| 9.6.1 | 数据库连接 .....              | 402 |
| 9.6.2 | 登录模块 .....               | 403 |
| 9.6.3 | 主窗体模块 .....              | 404 |
| 9.6.4 | 开台模块 .....               | 410 |
| 9.6.5 | 点菜模块 .....               | 412 |



|                                   |            |                                    |            |
|-----------------------------------|------------|------------------------------------|------------|
| 9.6.6 结账模块.....                   | 418        | 11.2 新的项目 .....                    | 484        |
| 9.6.7 员工管理模块.....                 | 420        | 11.3 系统需求分析 .....                  | 484        |
| 9.6.8 修改密码模块.....                 | 423        | 11.3.1 系统背景介绍 .....                | 485        |
| 9.6.9 桌台信息模块.....                 | 426        | 11.3.2 功能模块划分 .....                | 485        |
| 9.7 项目调试.....                     | 431        | 11.4 规划和运作 .....                   | 486        |
| <b>第 10 章 短信群发系统.....</b>         | <b>435</b> | 11.4.1 规划系统文件 .....                | 486        |
| 10.1 做好项目管理者.....                 | 436        | 11.4.2 运作流程 .....                  | 487        |
| 10.1.1 软件工程师到项目管理者<br>之路.....     | 436        | 11.5 设计数据库 .....                   | 487        |
| 10.1.2 赢在管理——运转一个高效<br>的开发团队..... | 436        | 11.5.1 数据库的概念设计 .....              | 487        |
| 10.2 做好需求分析 .....                 | 439        | 11.5.2 逻辑结构设计 .....                | 489        |
| 10.2.1 开发背景.....                  | 439        | 11.6 设计公共类 .....                   | 492        |
| 10.2.2 需求分析.....                  | 440        | 11.7 具体编码 .....                    | 502        |
| 10.3 项目规划.....                    | 440        | 11.7.1 用户登录模块 .....                | 502        |
| 10.3.1 系统目标.....                  | 441        | 11.7.2 主窗体模块.....                  | 503        |
| 10.3.2 划分功能模块.....                | 441        | 11.7.3 进货管理模块 .....                | 506        |
| 10.3.3 规划运作流程.....                | 441        | 11.7.4 进货查询模块 .....                | 510        |
| 10.4 搭建数据库.....                   | 443        | 11.7.5 商品销售管理模块 .....              | 512        |
| 10.4.1 数据库 E-R 图分析 .....          | 443        | 11.7.6 退货管理模块 .....                | 515        |
| 10.4.2 数据表的结构.....                | 444        | 11.7.7 库存管理模块 .....                | 517        |
| 10.5 设计公共类.....                   | 446        | 11.7.8 库存查询模块 .....                | 519        |
| 10.5.1 ConnClass 类 .....          | 447        | 11.7.9 数据备份模块 .....                | 520        |
| 10.5.2 GSM 类 .....                | 447        | 11.8 项目调试 .....                    | 522        |
| 10.6 具体编码.....                    | 454        | <b>第 12 章 家庭视频监控系统.....</b>        | <b>525</b> |
| 10.6.1 登录验证模块.....                | 454        | 12.1 走向架构师之路 .....                 | 526        |
| 10.6.2 主窗体模块.....                 | 456        | 12.1.1 什么是架构师 .....                | 526        |
| 10.6.3 短信群发模块.....                | 459        | 12.1.2 赢在架构——如何成为一名<br>架构师 .....   | 526        |
| 10.6.4 短信接收模块.....                | 463        | 12.1.3 赢在架构——如何成就一个<br>美丽的架构 ..... | 527        |
| 10.6.5 电话簿管理模块.....               | 468        | 12.2 新的项目 .....                    | 528        |
| 10.6.6 常用短语管理模块.....              | 471        | 12.3 需求分析 .....                    | 529        |
| 10.6.7 修改密码模块.....                | 474        | 12.3.1 系统背景介绍 .....                | 529        |
| 10.7 项目调试.....                    | 476        | 12.3.2 系统需求 .....                  | 529        |
| <b>第 11 章 超市进销存系统.....</b>        | <b>479</b> | 12.3.3 可行性分析 .....                 | 529        |
| 11.1 算法是程序的灵魂.....                | 480        | 12.3.4 编写项目计划书 .....               | 530        |
| 11.1.1 什么是算法.....                 | 480        | 12.4 系统设计 .....                    | 532        |
| 11.1.2 赢在技术沉淀——计算机<br>中的算法.....   | 481        | 12.5 数据库设计 .....                   | 533        |
|                                   |            | 12.5.1 数据库分析 .....                 | 533        |

|        |                      |     |        |                |     |
|--------|----------------------|-----|--------|----------------|-----|
| 12.5.2 | 数据库的概念设计.....        | 533 | 12.6.5 | 类 PelcoD ..... | 540 |
| 12.5.3 | 数据库的逻辑结构设计.....      | 533 | 12.7   | 具体编码 .....     | 542 |
| 12.6   | 设计公共类.....           | 534 | 12.7.1 | 登录模块 .....     | 542 |
| 12.6.1 | DataCon 类.....       | 534 | 12.7.2 | 视频监控模块 .....   | 543 |
| 12.6.2 | DataOperate 类.....   | 535 | 12.7.3 | 监控管理模块 .....   | 551 |
| 12.6.3 | SoftReg 类.....       | 535 | 12.7.4 | 录像回放模块 .....   | 553 |
| 12.6.4 | 类 VideoOperate ..... | 538 | 12.8   | 项目调试 .....     | 554 |



# 第 1 章 俄罗斯方块游戏

高帆项目开发

俄罗斯方块游戏是一款曾经风靡全球的电视游戏机和掌上游戏机游戏产品，它造就过令人惊奇的商业价值，影响过一代游戏产业链。

这款游戏最初是由苏联的游戏制作人 Alex Pajitnov 制作的，它看似简单，却变化无穷，令人玩起来上瘾。

本章将介绍如何在 Visual Studio 2013 环境下开发一款俄罗斯方块游戏，从而使读者能够迅速了解使用 Visual Studio 2013 集成开发环境创建小型、简单游戏的方法。

在开发过程中，读者将可以体验到 C# 所具有的强大功能，并熟悉其可视化的编程方式。



## 赠送的超值电子书

- 001 走进 C# 的世界
- 002 .NET Framework 简介
- 003 几个重要的概念
- 004 C# 的地位
- 005 C# 与 Java、C++ 的关系
- 006 C# 的基本语法
- 007 C# 的数据类型
- 008 基本类型
- 009 给变量命名
- 010 变量的声明和赋值

## 1.1 赢在规划

视频讲解 光盘：视频\第1章\赢在规划.avi

当一名程序员从实习生开始做起，依次经历码农、软件工程师、架构师、CTO 等职位的磨砺后，蓦然回首，会发现自己的成功并非偶然！如果需要总结出自己的成功秘诀，那么秘诀就是“比别人更加细致地做工作”。

在众多应届毕业生中，我们要想胜出，就要坦然面对职场竞争，作为刚刚步入职场的程序员，应该从细节上为自己的成功做好准备，这里的细节就是指职业规划。良好的职业规划决定了程序员以后的发展方向和具体轨迹，沿着规划的足迹行走的人，其成功率要远远高于那些好似“无头苍蝇”的迷茫者。

### 1.1.1 赢在起点——程序员的职业规划

通常来说，程序开发人员的职业发展有如下所示的几个选择：

- 专注于技术，成为技术专家。
- 转型到技术型销售、技术支持等。
- 随着技术的成长，从技术性管理走向高级管理。

上述三个方向是都能看得很清楚的，并且这三个方向都是以技术为基础的。在扎实的技术基础上，如果有比较强的抽象设计能力，而且又打算专注于技术开发，则做架构师是一个不错的选择；如果待人接物能力突出，善于跟客户打交道，则可以转型到销售部门，做技术支持；如果性格更适合于管理，情商表现很突出，则技术管理岗位乃至高级管理岗位应该是下一步的方向。

所以，对于已经工作两年以上的程序员来说，一般可以有几种基本的职业选择：技术专家、软件架构师、实施顾问或销售。并且，无论是 C、C++、C#、Java、.NET 还是数据库领域，都要首先成为专家，然后才可能继续发展为架构师。尽管架构师的职位待遇优厚，可以工作一辈子，但这种工作职位是很有限的，目前在我国 IT 行业中，对架构师的条件要求比较苛刻，且并不是很合理的，与国际上同行业的现状相比，是有一定差距的。

### 1.1.2 赢在项目开发伊始——做好项目分析

很多开发者，特别是一些初级开发者，写程序时，总是在看到功能需求后，就立即投入到代码编写工作中，需要什么功能，就编写函数去一一实现。

按照这种习惯做事情，在后期调试时，却总会出现这样或那样的错误，可能需要返工，重新做大量的修改。

幸运的是，初学者所接触到的，一般都是小项目，修改的工作量也不是很大。

但是，如果在大型项目中，要对几千行代码返回修改，则是一件很恐怖的事情！

可见，提前做好项目分析和规划是非常重要的。

一个软件项目的开发主要分为 5 个阶段，分别是需求分析阶段、设计阶段、编码阶段、



测试阶段和维护阶段。这里，需求分析阶段所得到的结果，是软件项目开发中其他 4 个阶段的必备条件。从以往的经验来看，需求分析中的一个小小的偏差，就可能导致整个项目无法达到预期的效果，或者说，可能导致最终开发出的产品不是用户所需要的。

软件需求分析的任务，不是确定系统是怎样完成工作的，而是确定系统必须完成哪些工作，也就是对目标系统提出完整、准确、清晰、具体的要求。它所做的工作是深入描述软件的功能和性能，确定软件设计的限制，以及软件同其他系统的接口细节，定义软件的其他有效性要求。

我们可以把软件需求分析的过程具体分为 4 个阶段，分别是对问题的识别、分析与综合、制定规格说明和评审。

### 1. 对问题的识别

对问题的识别是指系统分析人员研究可行性分析报告和软件项目实施计划，确定目标系统的综合要求，并提出这些需求实现的条件，以及需要达到的标准。这些需求主要分为功能性需求和非功能性需求两种，具体如下。

(1) 功能需求：列举出所开发的软件在功能上应具备什么。

(2) 性能需求：给出所开发软件的技术性能指标，如存储容量限制、运行时间限制、安全保密性要求等。

(3) 环境需求：软件系统运行时所处环境的要求。如硬件方面的机型、外部设备、数据通信接口；软件方面的系统软件(包括操作系统)、网络软件、数据库管理系统；使用方面的部门制度、操作人员的技术水平等。

(4) 可靠性需求：对所开发的软件在投入运行后不发生故障的概率按实际的运行环境提出要求。所以对于重要的软件，或是运行失效会造成严重后果的软件，应提出较高的可靠性要求。

(5) 安全保密要求：应当在这方面恰当地做出规定，对所开发的软件给予特殊的设计，使其在运行中的安全保密性能得到必要的保证。

(6) 用户界面需求：为用户界面细致地规定应该达到的要求。

(7) 资源使用需求：开发的软件在运行时和开发时所需要的各种资源。

(8) 软件成本消耗和开发进度需求：在软件项目立项后，要根据合同规定，对软件开发的进度和各步骤的费用提出要求，作为开发管理的依据。

(9) 预先估计以后系统可能达到的目标，这样，可以比较容易对系统进行必要的补充和修改。除了这些必需的需求，问题识别的另一个工作是建立分析所需要的通信途径，以保证能顺利地对问题进行分析。

### 2. 分析与综合

分析与综合的目标，是给出目标系统的详细逻辑模型。在此步骤中，分析和综合工作需反复地进行。

### 3. 制定规格说明

需要编制需求分析文档，这种文档又称为软件需求规格说明书。除了编写软件需求规格说明书之外，还要制定数据要求说明书，以及编写初步的用户手册。

#### 4. 评审

需求分析评审，是指在需求分析的最后一步，对系统功能的正确性、完整性和清晰性以及其它需求给予评价。

## 1.2 第一个项目

视频讲解 光盘：视频\第 1 章\第一个项目.avi

在做一个项目之前，一定要做好构思和规划工作，并根据需要制定开发流程。本项目的开发流程如图 1-1 所示。



图 1-1 本项目的开发流程

对于初次开发完整软件项目的程序员来说，开发的第一个项目十分重要。在开发伊始，可能会信心不足。此时，就需要建立充分的自信心。

作为一名程序员，面对项目时，我们要仔细分析，想法尝试，想法去实现，这样才能进步，才能找到自己的不足。

## 1.3 功能描述

视频讲解 光盘：视频\第 1 章\功能描述.avi

在程序员开发一个应用系统之前，需要彻底弄清这个应用系统的使用过程和必备的具体功能。几乎所有的程序员都会知道这一点，但是绝大多数开发者都对此不重视，认为太基本、太简单和太理所当然。

在此我们提醒开发人员，一定要重视市场调研工作。因为市场的发展是瞬息万变的，一夜之间可能会诞生很多的新奇好用的应用。

所以，要想更好地做好俄罗斯方块游戏项目的功能分析工作，需要将这款游戏彻底试玩几次，全面了解俄罗斯方块游戏的具体玩法。

为此，作者专门从网上下载了一款俄罗斯方块游戏，并详细地进行了试玩。

其游戏界面效果如图 1-2 所示。很华丽，是不是？我们将取其精华，做个简单的。

根据俄罗斯方块游戏的游戏规则和要求，可以总结出俄罗斯方块游戏的基本功能模块。当然，因为俄罗斯方块游戏是一款在市面中流行多年的游戏，所以游戏的基本玩法和功能



大家都耳熟能详。这就有利于在项目规划伊始进行玩法规划设计。显然，这是一个比较“庞大”的工程，相关的要点将在接下来的内容中进行讲解。

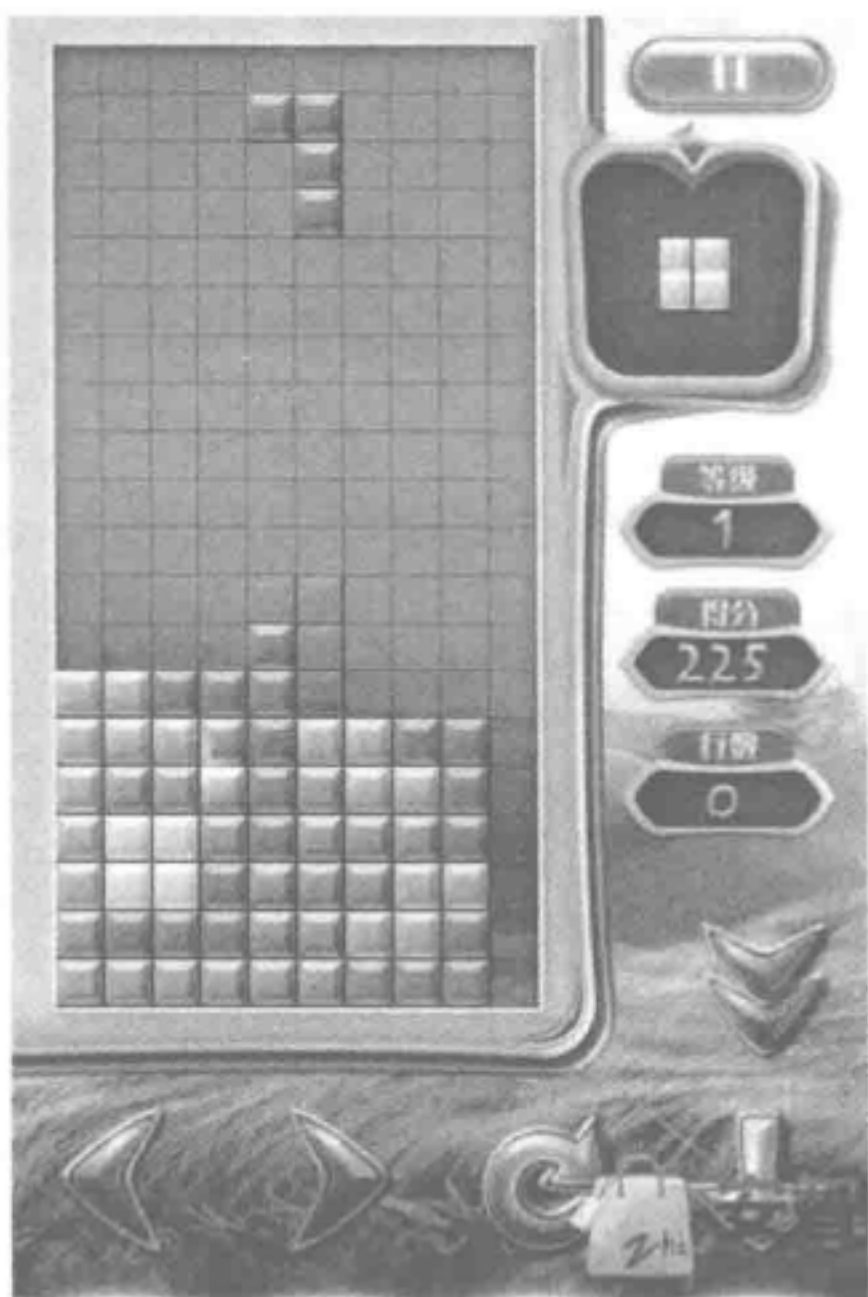


图 1-2 某款俄罗斯方块游戏的界面

(1) 由软件项目的开发流程，可以做出一个简单的项目规划书，整个规划书分为如下两个部分：

- 系统需求分析。
- 结构规划。

(2) 俄罗斯方块游戏项目的具体开发流程如图 1-3 所示。



图 1-3 俄罗斯方块游戏项目的具体开发流程

- 功能分析：分析整个系统所需要的功能。
- 结构规划：规划系统所需要的功能模块。
- 总体设计：分析系统处理流程，探索系统核心模块的运作。
- 数据结构：设计系统中需要的数据结构。
- 规划函数：预先规划系统中需要的功能函数。
- 具体编码：编写系统的具体实现代码。

### 1.3.1 功能模块分析

本项目实例的主要功能就是控制游戏的运行，实现游戏的完整过程。具体来说，主要包括如下功能模块。

- (1) 游戏运行界面：供用户在可视平台下控制游戏。
- (2) 游戏控制菜单：可以控制游戏的开始、退出和级别选择。
- (3) 设置菜单：可以设置游戏的显示样式，并提供游戏帮助等信息。

### 1.3.2 游戏的模块结构

游戏的模块结构如图 1-4 所示。

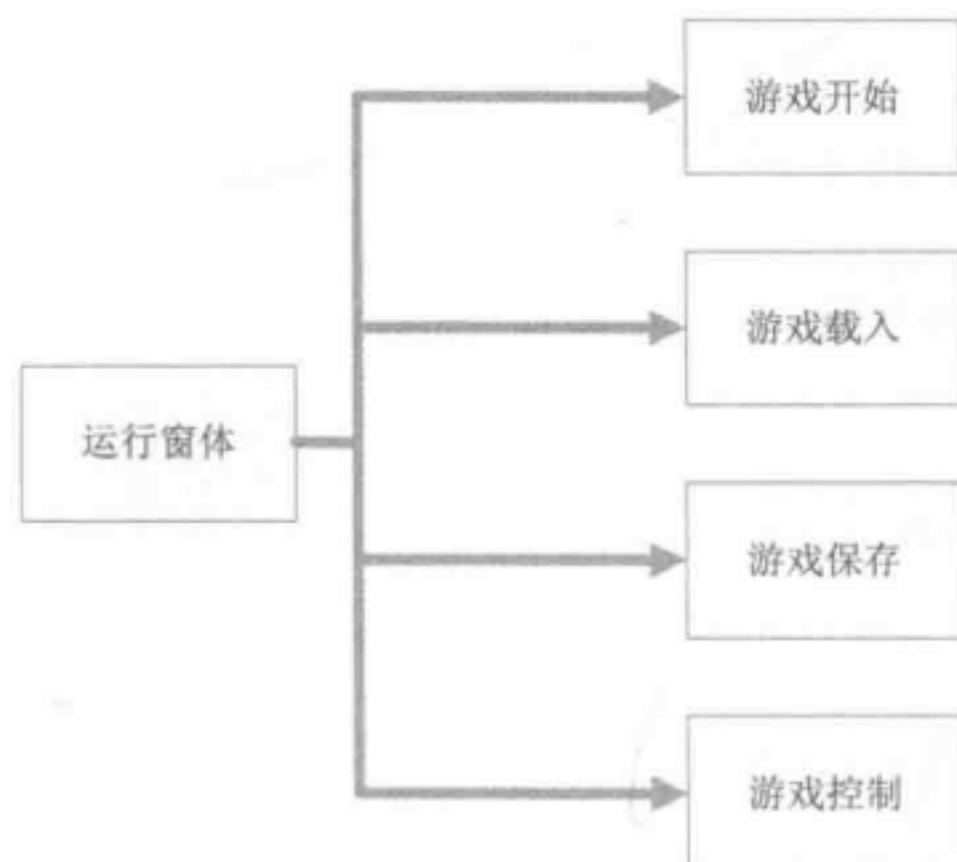


图 1-4 游戏的模块结构

### 1.3.3 游戏的运行流程

具体的运行流程如图 1-5 所示。

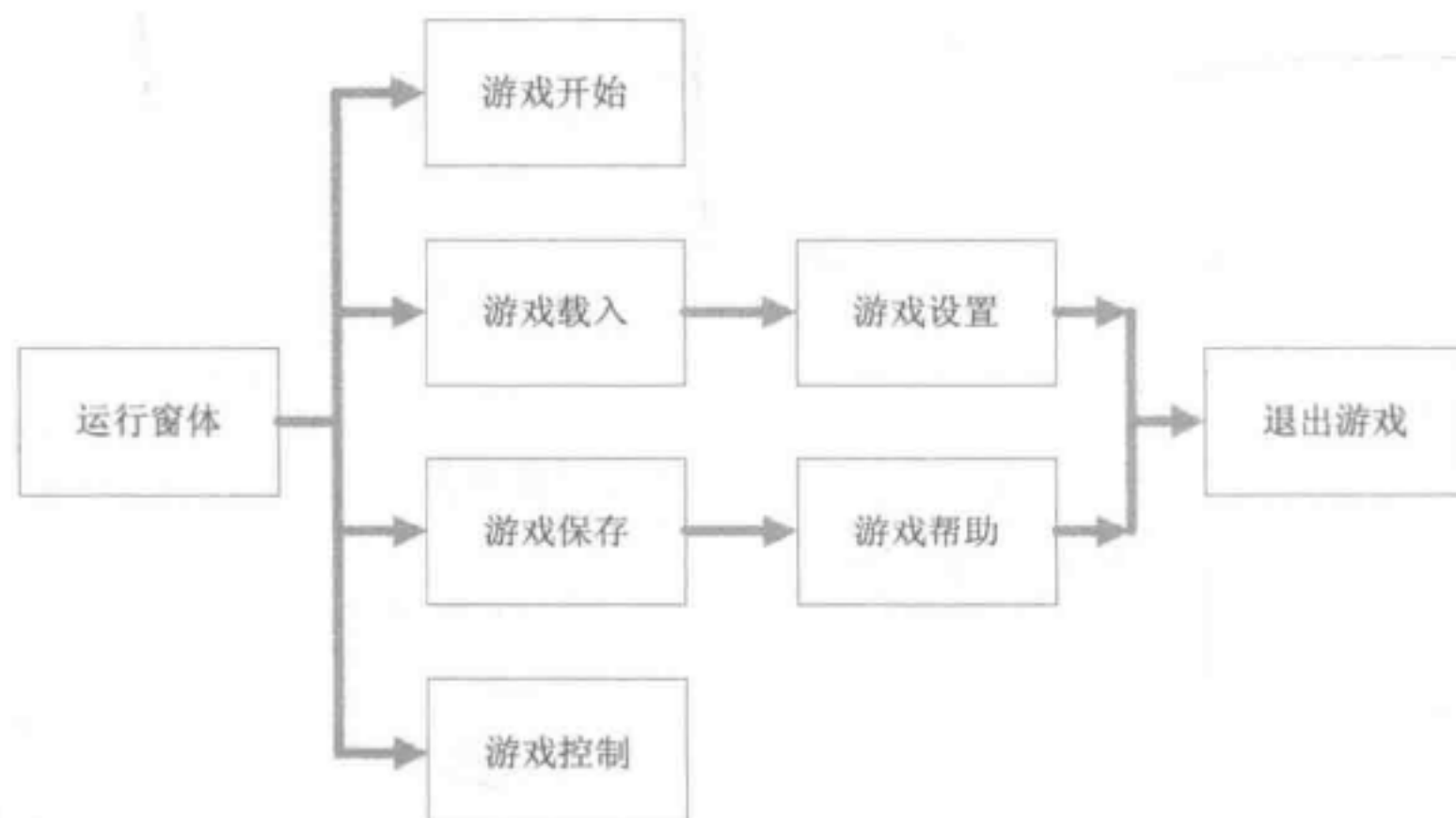


图 1-5 具体的运行流程



## 1.4 搭建开发平台

视频讲解 光盘：视频\第 1 章\搭建开发平台.avi

一款好的开发工具能够决定整个项目进展的顺利性。但是长久以来，我们一直很困惑，因为 Visual Studio .NET 已经推出好几个版本，市面中最流行的是 Visual Studio 2010，而当前最新的版本是 Visual Studio 2013。

工具的不断更新虽然提升了开发效能，但也提高了学习成本。本项目将使用 Visual Studio 2013 工具开发，这样可以体验 Visual Studio 2013 的最新功能。

### 1.4.1 安装 Visual Studio 2013

在安装 Visual Studio 2013 之前，需要先明确如下硬件要求：

- 最好有酷睿 II 2.0GHz 以上的 CPU。
- 至少应有 2GB 的 RAM 内存，其中 1GB 用于维持操作系统运行。
- 至少 10GB 的硬盘空间。

Visual Studio 2013 的具体安装步骤如下所示。

(1) 将安装盘放入光驱，或双击存储在硬盘内的安装文件 autorun.exe，弹出“开始安装”界面，如图 1-6 所示。

(2) 在出现的“安装路径”界面中选择安装路径，并勾选“同意安装条款”选项，单击“下一步”按钮，进入安装起始页界面，在这里选择将要安装的功能，如图 1-7 所示。在此建议全部选中，避免以后安装时遇到不可预知的麻烦。

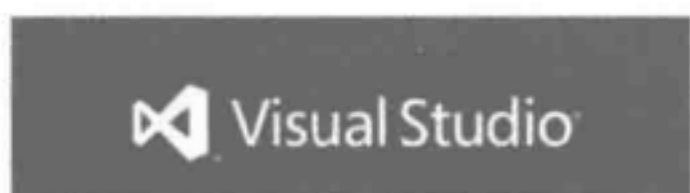


图 1-6 “开始安装”界面



图 1-7 选择安装的功能

(3) 单击“安装”按钮，出现安装进度界面，开始安装，如图 1-8 所示。

(4) 安装完成后，出现“重启”界面，在该界面中单击“立即重新启动”按钮。

(5) 重启后，将会继续安装，进度完成后，将完成所有的安装工作，如图 1-9 所示。



图 1-8 安装进度界面

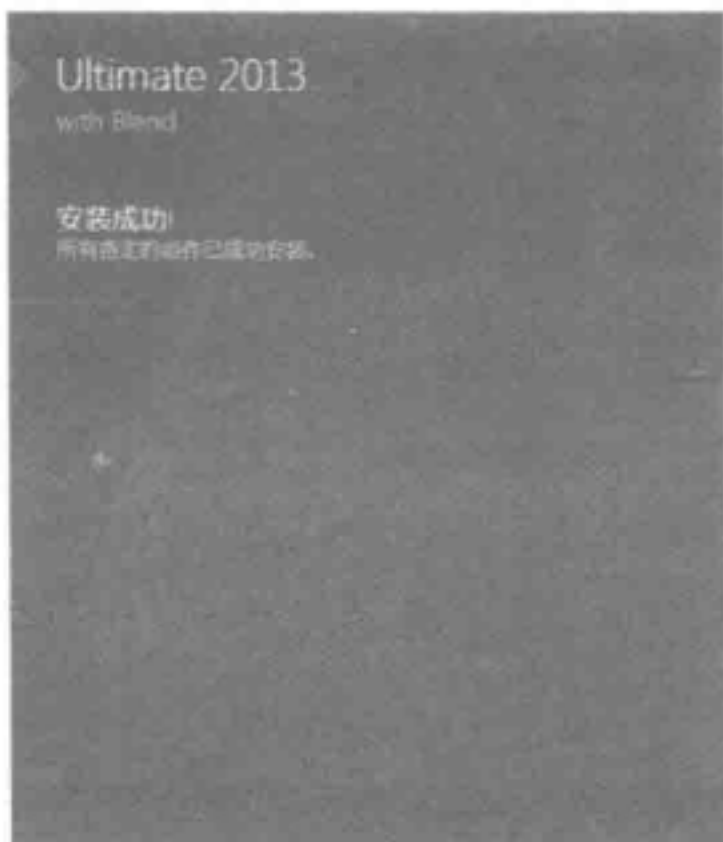


图 1-9 安装成功

(6) 完成安装后，可以从“开始”菜单中启动 Visual Studio 2013，如图 1-10 所示。



图 1-10 启动 Visual Studio 2013

首次打开安装后的 Visual Studio 2013 时，将会弹出“选择默认环境设置”对话框，如图 1-11 所示。本书中用 C# 开发 ASP.NET 程序，所以选择“Visual C# 开发设置”选项。

然后单击“启动 Visual Studio”按钮，开始加载用户环境设置，如图 1-12 所示。



图 1-11 “选择默认环境设置”对话框

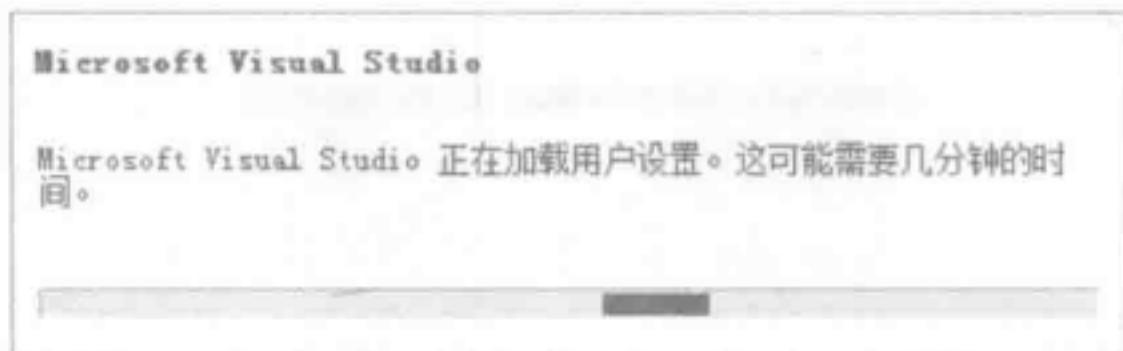


图 1-12 加载用户环境设置

配置完成后，将来到 Visual Studio 2013 的集成开发界面，如图 1-13 所示。





图 1-13 Visual Studio 2013 默认的综合开发界面

## 1.4.2 规划项目文件

安装 Visual Studio 2013 之后，发现比以往版本的界面好看了，具体增强的功能还得在后面的使用中才能体会到。接下来，需要新建一个名为“youxi”的窗体项目文件。并新建项目中需要的程序文件，在 VS 资源管理器中的效果如图 1-14 所示。

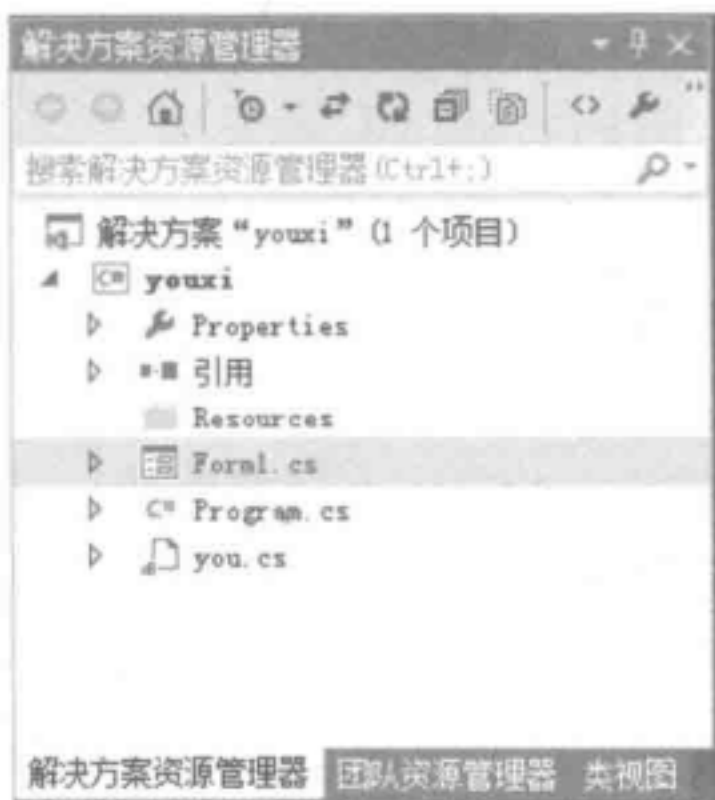


图 1-14 实例中资源管理器的效果

各个构成文件的具体说明如下。

- 文件 Form1.cs: 是项目的窗体文件，项目运行后，将调用各窗体元素的对应属性来显示窗体，并且设置各控件的对应事件处理程序。
- 方法定义文件 you.cs: 功能是使用 C# 设计项目所需要的各种功能方法，当其他文件需要时，只需调用方法的文件名即可。
- 文件 Program.cs: 是整个项目的入口文件。
- 文件 Form1.Designer.cs: 设置窗体各控件和组件的属性。

历时两天，确定好了整个项目的功能模块，做好了整体规划，并选好开发工具开始开发了。这几天体会到总体设计是一个项目的开始，也是后续工作得以顺利进行的前提。所以在此阶段应当是一丝不苟的，考虑到一切影响因素，尽量为后续工作打好坚实的基础。这样看似前面的工作使用了较多的时间，但实际上是节约了后面的时间。总结完毕之后，决定早点休息，为接下来的界面设计养精蓄锐。

## 1.5 界面设计

视频讲解 光盘：视频\第1章\界面设计.avi

界面设计是实现窗体项目的第一个步骤，需要在 Visual Studio 2013 窗体中插入布局控件。系统窗体界面是项目运行后显示的界面，也是游戏的运行界面，用户可以在此窗体上实现对游戏的控制。

使用 Visual Studio 2013 可以轻松地实现对游戏窗体界面的设计。在界面设计的过程中，需要遵循软件用户界面的设计原则。

软件用户界面(Software User Interface)是指软件用于同用户交流的外观、部件和程序等。软件界面的设计，既要从外观上进行创意以达到吸引眼球的目的，又要符合图形和版面设计的相关原理。

通常来讲，软件用户界面的设计应遵循以下几个基本原则。

### (1) 面向用户(User-oriented)

设计软件首先要明确到底谁是使用者，要站在用户的观点和立场上来设计软件。要做到这一点，必须要与用户进行沟通，了解他们的需求、目标、期望和偏好等。界面的设计者要清楚，用户之间差别很大，他们的能力各有不同。另外，用户使用的计算机机器配置也是千差万别的，包括显卡、声卡、内存、操作系统等都可能不同。设计者如果忽视了这些差别，设计出的程序在不同机器上的表现就会比较混乱。

### (2) KISS(Keep It Simple and Stupid)

KISS 原则就是 Keep It Simple and Stupid(简单明快)，简洁和易于操作是设计的最重要的原则。毕竟，软件建设出来是为普通人群服务的，没有必要在界面上设置过多的操作。该原则一般的要求是减少大幅图片和动画的使用，确保操作尽量简单，并且有明确的操作提示，软件所有的内容和服务都在显眼处向用户做出说明等。

### (3) 布局控制

在界面布局方面，很多设计者重视不够，界面设计得过于死板。如果界面布局凌乱，堆砌了大量的信息，就会干扰正常使用。

### (4) 色彩的搭配和文字的可阅读性

颜色是影响界面的重要因素，不同的颜色对人的感觉有不同的影响，例如，红色和橙色使人兴奋和心跳加快，黄色使人联想到阳光，是一种快活的颜色，黑色显得比较庄重。

为方便阅读软件上的信息，可以参考报纸的编排方式，将软件内容分栏设计。

另一种能够提高文字可读性的因素是所选择的字体，通用的字体(Arial、Courier New、Garamond、Times New Roman、中文宋体等)最易阅读，适合用于正文。特殊字体用于标题



效果较好,但是不适合正文。如果在整个界面中使用很多特殊字体(如 Cloister、Gothic、Script、Westminster、华文彩云、华文行楷),则人们阅读起来感觉一定会很糟糕,会使阅读颇为费力,眼睛很快就会疲劳,不得不转移到其他界面。

#### (5) 和谐一致性

通过对软件的各种元素(颜色、字体、图形、空白等)使用一定的规格,使得设计良好的界面看起来显得和谐。

或者说,软件的众多单独界面应该看起来像一个整体。

软件设计上要保持一致性,这是很重要的一点。

一致性的结构设计可以让浏览者对软件的形象有深刻的记忆,一致的导航设计可以让浏览者迅速而又有效地进入软件中自己所需要的部分,一致的操作设计可以让浏览者快速学会在整个软件中的各种功能操作。如果破坏这一原则,就会误导浏览者,并且让整个软件显得杂乱无章,给人留下不良的印象。

#### (6) 个性化

##### ① 符合网络文化

企业软件不同于传统的企业商务活动,要符合 Internet 网络文化的要求。整个互联网的文化是一种休闲的、非正式性的、轻松活泼的文化。

在软件上使用幽默的网络语言,可以创造出一种休闲的、轻松愉快的、非正式的氛围,会使软件的访问量大增。

##### ② 塑造软件个性

软件的整体风格和整体气氛表达要与企业形象相符合,并应该很好地体现企业 CI。

在这方面,比较经典的案例有:可口可乐个性鲜明的前卫软件 Life Tastes Good; 工整、全面、细致的通用电气公司软件 We bring good things to life(GE 带来美好的生活); 崇尚科技创新文化的 3M 公司软件 Creating solutions for business, industry and home 等。

在接下来的内容中,将遵循上述界面设计原则,创建俄罗斯方块游戏软件的界面。

### 1.5.1 制作游戏窗体

本游戏项目窗体界面的设计过程如下。

(1) 打开 Visual Studio 2013,新建一个名为“youxi”的 Windows 窗体应用程序,“新建项目”界面如图 1-15 所示。

(2) 根据面向用户的设计原则,从工具箱中依次将 4 个 Button 控件和 1 个 TrackBar 控件拖入窗体中,各控件从上到下的具体设置如下。

- 第一个 Button 控件:设置 name 属性为“buttonReplay”,Text 属性为“重新开始”,设置其 Click 事件为“buttonReplay\_Click”。
- 第二个 Button 控件:设置 name 属性为“buttonSave”,Text 属性为“保存”,设置其 Click 事件为“buttonSave\_Click”。
- 第三个 Button 控件:设置 name 属性为“buttonLoad”,Text 属性为“载入”,设置其 Click 事件为“buttonLoad\_Click”。
- 设置 TrackBar 控件:设置 name 属性为“trackBarReviewSpeed”,Maximum 属性为“15”,Minimum 属性为“1”,设置其 Scroll 事件为“trackBar1\_Scroll”。

- 第四个 Button 控件：设置 name 属性为“buttonReview”，Text 属性为“Re&view”，设置其 Click 事件为“buttonReview\_Click”。



图 1-15 新建 Windows 窗体应用程序

设置完毕后的窗体界面效果如图 1-16 所示。

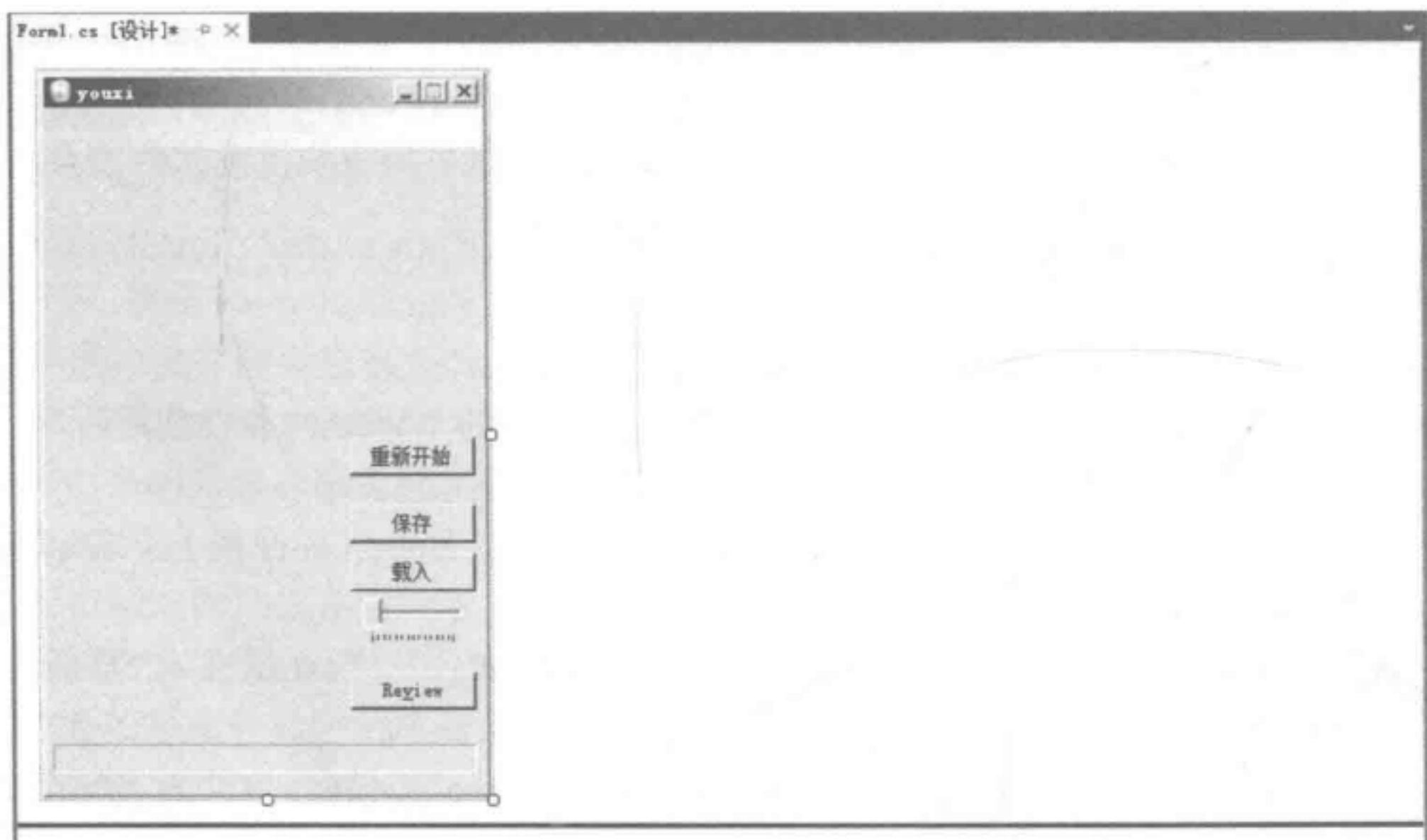


图 1-16 窗体界面的效果

(3) 为了方便用户灵活地使用软件，从工具箱中拖入 MenuStrip 控件到窗体顶部，并分别设置“文件”、“设置选项”和“游戏帮助”这 3 个菜单，如图 1-17 所示。

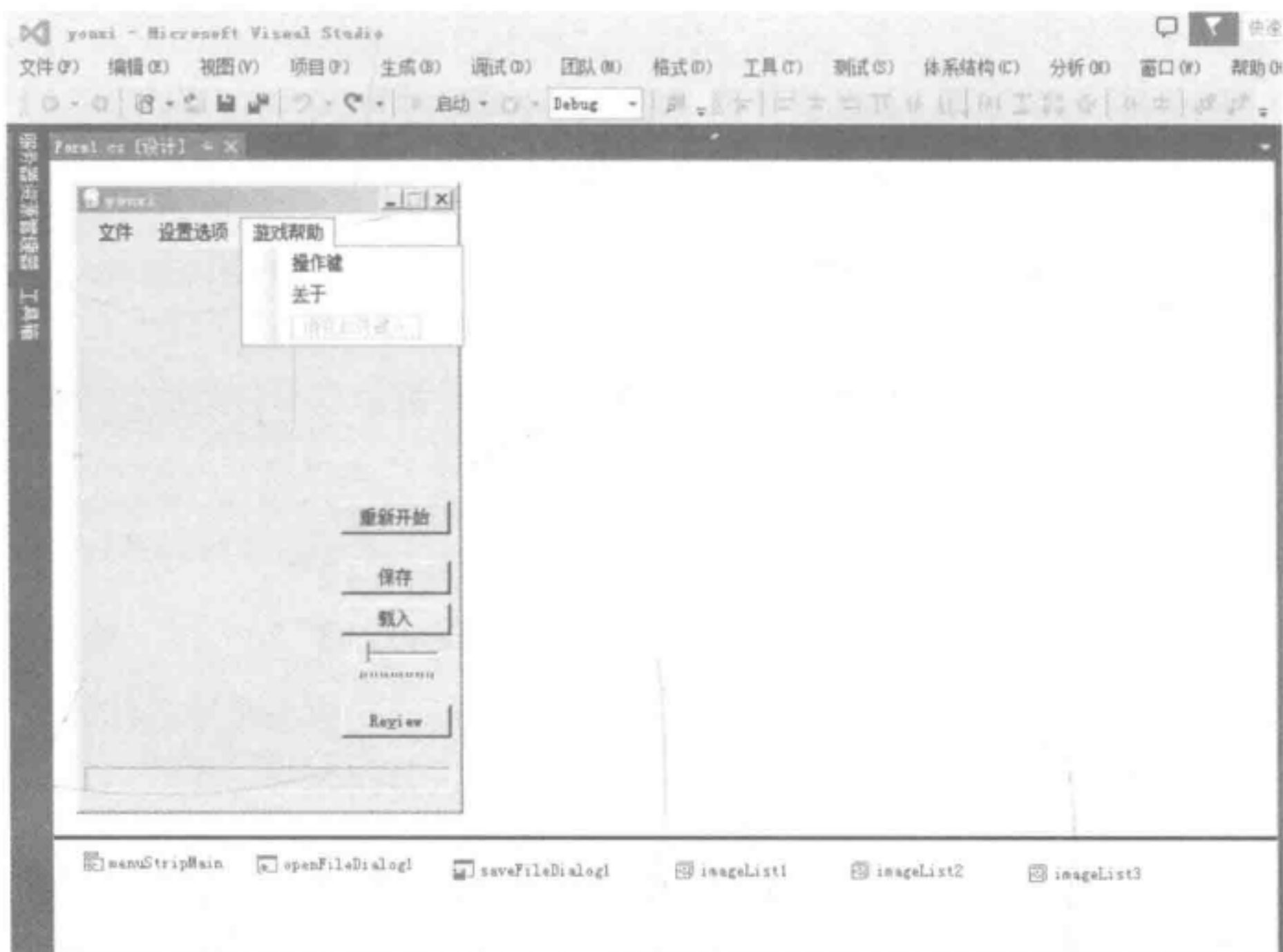


图 1-17 设置主菜单

(4) 设置主菜单中的各个菜单命令，首先设置“文件”菜单，在其下面设置 6 个菜单项。从上到下各个菜单项的具体信息如下。

- 第一个菜单项：设置 name 属性为“replayToolStripMenuItem”，Text 属性为“重新&开始”，设置其 Click 事件为“replayToolStripMenuItem\_Click”。
- 第二个菜单项：设置 name 属性为“replayExtendedToolStripMenuItem”，Text 属性为“Replay(Extended)”，设置其 Click 事件为“replayExtendedToolStripMenuItem\_Click”。
- 第三个菜单项：设置 name 属性为“saveToolStripMenuItem”，Text 属性为“&Save”，设置其 Click 事件为“saveToolStripMenuItem\_Click”。
- 第四个菜单项：设置 name 属性为“loadToolStripMenuItem”，Text 属性为“&Load”，设置其 Click 事件为“loadToolStripMenuItem\_Click”。
- 第五个菜单项：设置 name 属性为“reviewToolStripMenuItem”，Text 属性为“Re&view”，设置其 Click 事件为“reviewToolStripMenuItem\_Click”。
- 第六个菜单项：设置 name 属性为“exiToolStripMenuItem”，Text 属性为“E&xit”，设置其 Click 事件为“exiToolStripMenuItem\_Click”。

(5) 设置“设置选项”菜单，在其下面设置 1 个名为“styleToolStripMenuItem”的菜单项，然后在此菜单项下再设置 3 个子菜单，从上到下各子菜单项的具体信息如下。

- 第一个子菜单项：设置 name 属性为“style1ToolStripMenuItem”，Text 属性为“style1”，设置其 Click 事件为“style1ToolStripMenuItem\_Click”。
- 第二个子菜单项：设置 name 属性为“style2ToolStripMenuItem”，Text 属性为“style2”，设置其 Click 事件为“style2ToolStripMenuItem\_Click”。



- 第三个子菜单项：设置 name 属性为“style3ToolStripMenuItem”，Text 属性为“style3”，设置其 Click 事件为“style3ToolStripMenuItem\_Click”。

(6) 从工具箱插入组件 OpenFileDialog 和 SaveFileDialog，分别设置 name 属性为“openFileDialog1”和“saveFileDialog1”。

(7) 从工具箱中拖入 ProgressBar 控件到窗体底部，并且设置该控件的 name 属性为“progressBarReview”。

(8) 从工具箱中分别拖入 3 个 ImageList 控件，用于设置游戏窗体界面的外观显示样式。各控件的具体设置信息如下：

- 第一个 ImageList 控件，设置 name 属性为“imageList1”。
- 第二个 ImageList 控件，设置 name 属性为“imageList2”。
- 第三个 ImageList 控件，设置 name 属性为“imageList3”。

## 1.5.2 窗体元素设置文件

经过 1.5.1 小节中的设置操作后，项目的窗体界面基本设计完毕。在窗体元素设置文件 Form1.Designer.cs 内，定义了各窗体控件的具体属性设置。在下面的内容中将分别介绍。

### 1. 控制按钮

在窗体右侧的各控制按钮控件的功能是控制游戏的进程，例如游戏开始、载入和保存。对应属性的设置代码如下：

```
// buttonReplay
this.buttonReplay.Location = new System.Drawing.Point(180, 197);
this.buttonReplay.Name = "buttonReplay";
this.buttonReplay.Size = new System.Drawing.Size(75, 23);
this.buttonReplay.TabIndex = 0;
this.buttonReplay.Text = "重新开始";
this.buttonReplay.UseVisualStyleBackColor = true;
this.buttonReplay.Click += new System.EventHandler(this.buttonReplay_Click);
// buttonReview
this.buttonReview.Location = new System.Drawing.Point(180, 338);
this.buttonReview.Name = "buttonReview";
this.buttonReview.Size = new System.Drawing.Size(75, 23);
this.buttonReview.TabIndex = 1;
this.buttonReview.Text = "Re&view";
this.buttonReview.UseVisualStyleBackColor = true;
this.buttonReview.Click += new System.EventHandler(this.buttonReview_Click);
// trackBarReviewSpeed
this.trackBarReviewSpeed.Location = new System.Drawing.Point(180, 292);
this.trackBarReviewSpeed.Maximum = 15;
this.trackBarReviewSpeed.Minimum = 1;
this.trackBarReviewSpeed.Name = "trackBarReviewSpeed";
this.trackBarReviewSpeed.Size = new System.Drawing.Size(75, 42);
this.trackBarReviewSpeed.TabIndex = 2;
this.trackBarReviewSpeed.Value = 1;
this.trackBarReviewSpeed.Scroll += new System.EventHandler(this.trackBar1_Scroll);
// buttonSave
this.buttonSave.Location = new System.Drawing.Point(180, 238);
this.buttonSave.Name = "buttonSave";
this.buttonSave.Size = new System.Drawing.Size(75, 23);
this.buttonSave.TabIndex = 3;
this.buttonSave.Text = "保存";
```

```

this.buttonSave.UseVisualStyleBackColor = true;
this.buttonSave.Click += new System.EventHandler(this.buttonSave_Click);
// buttonLoad
this.buttonLoad.Location = new System.Drawing.Point(180, 267);
this.buttonLoad.Name = "buttonLoad";
this.buttonLoad.Size = new System.Drawing.Size(75, 23);
this.buttonLoad.TabIndex = 4;
this.buttonLoad.Text = "载入";
this.buttonLoad.UseVisualStyleBackColor = true;
this.buttonLoad.Click += new System.EventHandler(this.buttonLoad_Click);
// progressBarReview
this.progressBarReview.Location = new System.Drawing.Point(3, 381);
this.progressBarReview.Name = "progressBarReview";
this.progressBarReview.Size = new System.Drawing.Size(252, 17);
this.progressBarReview.TabIndex = 5;

```

## 2. 窗体菜单

在窗体顶部的各菜单选项都包括对应的子选项，甚至还包含孙选项，其功能是对游戏进行控制和设置，并显示游戏的使用帮助信息。对应于窗体菜单各选项的属性设置代码如下所示：

```

// fileToolStripMenuItem
this.fileToolStripMenuItem.DropDownItems.AddRange(
    new System.Windows.Forms.ToolStripItem[] {
        this.replayToolStripMenuItem,
        this.replayExtendedToolStripMenuItem,
        this.saveToolStripMenuItem,
        this.loadToolStripMenuItem,
        this.reviewToolStripMenuItem,
        this.toolStripMenuItemLine,
        this.exitToolStripMenuItem});
this.fileToolStripMenuItem.Name = "fileToolStripMenuItem";
this.fileToolStripMenuItem.Size = new System.Drawing.Size(41, 32);
this.fileToolStripMenuItem.Text = "&文件";
// replayToolStripMenuItem
this.replayToolStripMenuItem.Name = "replayToolStripMenuItem";
this.replayToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
this.replayToolStripMenuItem.Text = "重新&开始";
this.replayToolStripMenuItem.Click +=
    new System.EventHandler(this.replayToolStripMenuItem_Click);
// replayExtendedToolStripMenuItem
this.replayExtendedToolStripMenuItem.Name = "replayExtendedToolStripMenuItem";
this.replayExtendedToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
this.replayExtendedToolStripMenuItem.Text = "Replay(Extended)";
this.replayExtendedToolStripMenuItem.Click +=
    new System.EventHandler(this.replayExtendedToolStripMenuItem_Click);
// saveToolStripMenuItem
this.saveToolStripMenuItem.Name = "saveToolStripMenuItem";
this.saveToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
this.saveToolStripMenuItem.Text = "&Save";
this.saveToolStripMenuItem.Click +=
    new System.EventHandler(this.saveToolStripMenuItem_Click);
// loadToolStripMenuItem
this.loadToolStripMenuItem.Name = "loadToolStripMenuItem";
this.loadToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
this.loadToolStripMenuItem.Text = "&Load";
this.loadToolStripMenuItem.Click +=
    new System.EventHandler(this.loadToolStripMenuItem_Click);

```



```

// reviewToolStripMenuItem
this.reviewToolStripMenuItem.Name = "reviewToolStripMenuItem";
this.reviewToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
this.reviewToolStripMenuItem.Text = "Re&view";
this.reviewToolStripMenuItem.Click +=
    new System.EventHandler(this.reviewToolStripMenuItem_Click);
// toolStripMenuItemLine
this.toolStripMenuItemLine.Name = "toolStripMenuItemLine";
this.toolStripMenuItemLine.Size = new System.Drawing.Size(163, 6);
// exitToolStripMenuItem
this.exitToolStripMenuItem.Name = "exitToolStripMenuItem";
this.exitToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
this.exitToolStripMenuItem.Text = "E&xit";
this.exitToolStripMenuItem.Click +=
    new System.EventHandler(this.exitToolStripMenuItem_Click);
// optionToolStripMenuItem
this.optionToolStripMenuItem.DropDownItems.AddRange(
    new System.Windows.Forms.ToolStripItem[] {
        this.styleToolStripMenuItem});
this.optionToolStripMenuItem.Name = "optionToolStripMenuItem";
this.optionToolStripMenuItem.Size = new System.Drawing.Size(65, 32);
this.optionToolStripMenuItem.Text = "&设置选项";
// styleToolStripMenuItem
this.styleToolStripMenuItem.DropDownItems.AddRange(
    new System.Windows.Forms.ToolStripItem[] {
        this.style1ToolStripMenuItem,
        this.style2ToolStripMenuItem,
        this.style3ToolStripMenuItem});
this.styleToolStripMenuItem.Name = "styleToolStripMenuItem";
this.styleToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.styleToolStripMenuItem.Text = "&Style";
//
// style1ToolStripMenuItem
this.style1ToolStripMenuItem.Name = "style1ToolStripMenuItem";
this.style1ToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.style1ToolStripMenuItem.Tag = "";
this.style1ToolStripMenuItem.Text = "style1";
this.style1ToolStripMenuItem.Click +=
    new System.EventHandler(this.style1ToolStripMenuItem_Click);
// style2ToolStripMenuItem
this.style2ToolStripMenuItem.Name = "style2ToolStripMenuItem";
this.style2ToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.style2ToolStripMenuItem.Text = "style2";
this.style2ToolStripMenuItem.Click +=
    new System.EventHandler(this.style1ToolStripMenuItem_Click);
// style3ToolStripMenuItem
this.style3ToolStripMenuItem.Name = "style3ToolStripMenuItem";
this.style3ToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.style3ToolStripMenuItem.Text = "style3";
this.style3ToolStripMenuItem.Click +=
    new System.EventHandler(this.style1ToolStripMenuItem_Click);
// helpToolStripMenuItem
this.helpToolStripMenuItem.DropDownItems.AddRange(
    new System.Windows.Forms.ToolStripItem[] {
        this.keyboardToolStripMenuItem,
        this.aboutToolStripMenuItem});
this.helpToolStripMenuItem.Name = "helpToolStripMenuItem";
this.helpToolStripMenuItem.Size = new System.Drawing.Size(65, 32);
this.helpToolStripMenuItem.Text = "&游戏帮助";
// keyboardToolStripMenuItem
this.keyboardToolStripMenuItem.Name = "keyboardToolStripMenuItem";

```



```

this.keyboardToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.keyboardToolStripMenuItem.Text = "&操作键";
this.keyboardToolStripMenuItem.Click +=
    new System.EventHandler(this.keyBoardToolStripMenuItem_Click);
// aboutToolStripMenuItem
this.aboutToolStripMenuItem.Name = "aboutToolStripMenuItem";
this.aboutToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.aboutToolStripMenuItem.Text = "&关于";
this.aboutToolStripMenuItem.Click +=
    new System.EventHandler(this.aboutToolStripMenuItem_Click);
// menuStripMain
this.menuStripMain.Items.AddRange(
    new System.Windows.Forms.ToolStripItem[] {
        this.fileToolStripMenuItem,
        this.optionToolStripMenuItem,
        this.helpToolStripMenuItem});
this.menuStripMain.Location = new System.Drawing.Point(0, 0);
this.menuStripMain.Name = "menuStripMain";
this.menuStripMain.Size = new System.Drawing.Size(260, 24);
this.menuStripMain.TabIndex = 6;
this.menuStripMain.Text = "menuStrip1";

```

### 3. 图片列表控件和窗体的总体设置

在窗体内插入了图片列表控件，其功能是设置游戏的外观显示样式，供用户根据个人喜好进行选择。而窗体总体设置属性的功能是设置窗体 Form1 的对应属性。

上述各窗体选项属性的设置代码如下所示：

```

// imageList1
this.imageList1.ImageStream = ((System.Windows.Forms.ImageListStreamer)
    (resources.GetObject("imageList1.ImageStream")));
this.imageList1.TransparentColor = System.Drawing.Color.Transparent;
this.imageList1.Images.SetKeyName(0, "1");
this.imageList1.Images.SetKeyName(1, "2");
this.imageList1.Images.SetKeyName(2, "3");
this.imageList1.Images.SetKeyName(3, "4");
this.imageList1.Images.SetKeyName(4, "5");
this.imageList1.Images.SetKeyName(5, "6");
this.imageList1.Images.SetKeyName(6, "7");
// imageList2
this.imageList2.ImageStream = ((System.Windows.Forms.ImageListStreamer)
    (resources.GetObject("imageList2.ImageStream")));
this.imageList2.TransparentColor = System.Drawing.Color.Transparent;
this.imageList2.Images.SetKeyName(0, "1");
this.imageList2.Images.SetKeyName(1, "2");
this.imageList2.Images.SetKeyName(2, "3");
this.imageList2.Images.SetKeyName(3, "4");
this.imageList2.Images.SetKeyName(4, "5");
this.imageList2.Images.SetKeyName(5, "6");
this.imageList2.Images.SetKeyName(6, "7");
// saveFileDialog1
this.saveFileDialog1.Filter = "youxi records files (*.trf)|*.trf";
// imageList3
this.imageList3.ImageStream = ((System.Windows.Forms.ImageListStreamer)
    (resources.GetObject("imageList3.ImageStream")));
this.imageList3.TransparentColor = System.Drawing.Color.Transparent;
this.imageList3.Images.SetKeyName(0, "1");
this.imageList3.Images.SetKeyName(1, "2");
this.imageList3.Images.SetKeyName(2, "3");

```


```

this.imageList3.Images.SetKeyName(3, "4");
this.imageList3.Images.SetKeyName(4, "5");
this.imageList3.Images.SetKeyName(5, "6");
this.imageList3.Images.SetKeyName(6, "7");
// Form1
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.BackColor = System.Drawing.SystemColors.ActiveBorder;
this.ClientSize = new System.Drawing.Size(260, 410);
this.Controls.Add(this.progressBarReview);
this.Controls.Add(this.buttonLoad);
this.Controls.Add(this.buttonSave);
this.Controls.Add(this.trackBarReviewSpeed);
this.Controls.Add(this.buttonReview);
this.Controls.Add(this.buttonReplay);
this.Controls.Add(this.menuStripMain);
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
this.MainMenuStrip = this.menuStripMain;
this.MaximizeBox = false;
this.Name = "Form1";
this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
this.Text = "youxi";
this.Load += new System.EventHandler(this.Formyouxi_Load);
((System.ComponentModel.ISupportInitialize)(this.trackBarReviewSpeed)).EndInit();
this.menuStripMain.ResumeLayout(false);
this.menuStripMain.PerformLayout();
this.ResumeLayout(false);
this.PerformLayout();

```

作为桌面系统，窗体界面的美感十分重要。好的窗体效果能给用户带来视觉冲击，也会留下好的印象。当然，最直接的好处是增加用户的满意度，让项目能够通过。在具体美化时，我们通常会用一些漂亮的素材图片作为背景，并用 CSS 来配合控件和信息的显示。至于具体怎样搭配，就要根据客户和设计师的喜好而确定了。

## 1.6 具体编码

视频讲解  光盘：视频\第 1 章\具体编码.avi

系统窗体界面的设计工作完毕后，接下来，开始步入正式的编码阶段的工作。在本节的内容中，将详细讲解本章俄罗斯方块游戏项目的具体编码过程。

### 1.6.1 事件处理程序

本章的项目实例通过编写的事件处理程序实现对游戏的控制，例如单击“重新开始”按钮后，将激活 `buttonReplay_Click` 事件，用户可以重新开始玩新的游戏。在 1.5 节的窗体设计过程中，已经涉及了项目内的各个事件，各事件的定义代码在文件 `Form1.cs` 内定义。

#### 1. 初始设置

这里的初始设置，是指通过 `using` 引用指令定义命名空间，分别定义窗体类 `Form1` 和对应的构造函数，具体代码如下所示：



```

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using youxi;
using System.IO;
namespace youxiApp
{
    public partial class Form1 : Form
    {
        private youxiControl youxiControl = new youxiControl();
        private youxiNext youxiNext = new youxiNext();
        private youxiScore youxiScore = new youxiScore();
        public Form1()
        {
            InitializeComponent();
            youxiControl.Top = menuStripMain.Height + 2;
            youxiControl.Left = 2;
            youxiControl.Parent = this;
            youxiControl.ImageList = imageList1;
            youxiNext.Parent = this;
            youxiNext.Top = youxiControl.Top;
            youxiNext.Left = youxiControl.Left + youxiControl.Width + 4;
            youxiControl.TetrisNext = youxiNext;
            youxiScore.Parent = this;
            youxiScore.Top = youxiNext.Top + youxiNext.Height + 4;
            youxiScore.Left = youxiNext.Left;
            youxiControl.TetrisScore = youxiScore;
            style1ToolStripMenuItem.Image = imageList1.Images[0];
            style2ToolStripMenuItem.Image = imageList2.Images[0];
            style3ToolStripMenuItem.Image = imageList3.Images[0];
            youxiControl.ProgressBar = progressBarReview;
            openFileDialog1.FileName =
                Path.GetDirectoryName(Application.ExecutablePath) + @"\sample.trf";
            saveFileDialog1.FileName =
                Path.GetDirectoryName(Application.ExecutablePath) + @"\sample.trf";
        }
    }
}

```

## 2. 编写事件处理代码

通过单击窗体 Form1 内的各个控件和组件，可以执行对应的事件处理程序，实现游戏的运行和对游戏的控制。在本章 1.5 节的窗体设计过程中，已经了解了各控件的执行事件名称，在此只对事件的具体处理代码进行介绍。各事件的具体代码如下。

### (1) 右侧按钮控件的处理事件

右侧按钮控件是指“重新开始”、“保存”、“载入”、“滑动条”和“Review”控件，各控件对应事件的处理代码如下所示：

```

private void buttonReplay_Click(object sender, EventArgs e)
{
    youxiControl.Replay(true, false);
}
private void buttonReview_Click(object sender, EventArgs e)
{
    youxiControl.Review();
}

```



```

private void trackBar1_Scroll(object sender, EventArgs e)
{
    youxiControl.ReviewSpeed = trackBarReviewSpeed.Value;
}
private void buttonSave_Click(object sender, EventArgs e)
{
    if (saveFileDialog1.ShowDialog() != DialogResult.OK) return;
    if (Path.GetExtension(saveFileDialog1.FileName) == string.Empty)
        saveFileDialog1.FileName =
            Path.ChangeExtension(saveFileDialog1.FileName, ".trf");
    youxiControl.SaveToFile(saveFileDialog1.FileName);
}
private void buttonLoad_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() != DialogResult.OK) return;
    if (!File.Exists(openFileDialog1.FileName)) return;
    youxiControl.LoadFromFile(openFileDialog1.FileName);
}

```

## (2) 窗体菜单的处理事件

窗体菜单是指窗体顶部的“文件”、“设置选项”、“游戏帮助”三个主菜单，并包含它们的各子菜单和孙菜单。上述各菜单对应事件的处理代码如下所示：

```

private void replayToolStripMenuItem_Click(object sender, EventArgs e)
{
    buttonReplay_Click(buttonReplay, new EventArgs());
}
private void saveToolStripMenuItem_Click(object sender, EventArgs e)
{
    buttonSave_Click(buttonSave, new EventArgs());
}
private void loadToolStripMenuItem_Click(object sender, EventArgs e)
{
    buttonLoad_Click(buttonLoad, new EventArgs());
}
private void reviewToolStripMenuItem_Click(object sender, EventArgs e)
{
    buttonReview_Click(buttonReview, new EventArgs());
}
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}
private void keyBoardToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("左移: Left、A\r\n右移: Right、D\r\n"
        + "下移: Down、S\r\n变换: Up、W\r\n变化: Back、F\r\n直下: Space、J、Enter",
        "按键说明");
}
private void blogToolStripMenuItem_Click(object sender, EventArgs e)
{
    Process.Start("http://blog.sina.com.cn/u/589d32f50100081s");
}
private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("俄罗斯方块\r\n设计: 天涯沦落人\r\n"
        + "日期: 2013-11-24\r\n联系: xxx@126.com", "关于本程序");
}
private void style1ToolStripMenuItem_Click(object sender, EventArgs e)
{
}

```

```

if (sender == style1ToolStripMenuItem)
    youxiControl.ImageList = imageList1;
else if (sender == style2ToolStripMenuItem)
    youxiControl.ImageList = imageList2;
if (sender == style3ToolStripMenuItem)
    youxiControl.ImageList = imageList3;
}
private void replayExtendedToolStripMenuItem_Click(object sender, EventArgs e)
{
    youxiControl.Replay(true, true);
}

```

## 1.6.2 游戏控制、处理方法

普通的社会规则是倡导助人为乐的，而我们玩的游戏却设置了很多等级和关口，这不是因为开发者偏要跟社会规则闹别扭，而是因为游戏就是游戏，是要吊玩家胃口的。在游戏中设置分数功能后，就可以激发用户永远去追逐高分。高分伴随高等级，等级能够提高玩家的争强好胜之心，吸引玩家继续玩下去，从而产生更多的乐趣。

从前面的实现过程可以看出，整个游戏项目的控制和处理是通过事件处理程序控制的。每个事件处理程序只是调用处理方法的名称，而实现具体功能的各种处理方法在文件 you.cs 中定义。

### 1. 初始设置

这里的初始设置是指定义游戏的控制类 youxiControl，并设置玩游戏过程中的返回行数、参数、等级、积分和方块的操作等参数。具体代码如下所示：

```

public class youxiControl : Control
{
    private const int rowCount = 21;           // 行数
    private const int colCount = 11;           // 列数
    private int brickWidth = 16;                // 小块宽度
    private int brickHeight = 16;              // 小块高度
    private ImageList imageList;                // 方块素材
    private Bitmap backBitmap;                  // 背景图片
    private List<List<List<Point>>>> brickTemplets =
        new List<List<List<Point>>>>();          // 方块模板[模板序号, 朝向]
    private byte[,] points = new byte[colCount, rowCount]; // 点阵
    private byte brickIndex = 0;                // 模板序号
    private byte facingIndex = 0;               // 当前变化号
    private Point brickPoint = new Point();      // 方块的位置
    private byte afterBrickIndex = 0;            // 下一个模板序号
    private byte afterFacingIndex = 0;           // 下一个变化号
    private System.Windows.Forms.Timer timer;    // 控制下落的定时器
    private int lines;                          // 消行数
    private Random random = new Random();        // 随机数
    private int level = 0;                      // 当前速度
    private int score = 0;                      // 成绩
    /// 下落速度, 数值表示每次下落的时间差, 以毫秒为单位
    private int[] speeds = new int[] { 700, 500, 400, 300, 320, 320, 100, 80, 70, 60, 50 };
    /// 每次消除行所增加的积分
    private int[] scoress = new int[] { 0001, 0100, 0300, 0500, 1000, 3200 };
    private bool playing = false;               // 玩家是否正在游戏
    private youxiNext youxiNext;               // 下一个方块的显示控件
    private youxiScore youxiScore;             // 积分显示控件
}

```



```

private int stepIndex = -1;           // 当前回放的步数
private bool reviewing = false;       // 是否正在回放中
private Thread threadReview = null;   // 回放使用的线程
private int reviewSpeed = 1;         // 回放的速度, 数值表示倍数
private List<StepInfo> StepInfos = new List<StepInfo>(); // 记录玩家每一步的操作
private int lastRecordTime = 0;       // 最后记录的时间
private bool recordMode = false;      // 是否采用记录模式
private ProgressBar progressBar;      // 回放进度条
private bool extended = false;        // 扩展方块
/// 消除的行数
public int Lines { get { return lines; } }
/// 当前的积分
public int Score { get { return score; } }
/// 当前的关数
public int Level { get { return level; } }
/// 方块的操作
public enum BrickOperates
{
    boMoveLeft = 0,           // 左移
    boMoveRight = 1,          // 右移
    boMoveDown = 2,           // 下移
    boMoveBottom = 3,         // 直下
    boTurnLeft = 4,           // 左旋
    boTurnRight = 5,          // 右旋
}

```

## 2. Replay 处理

单击窗体内的“重新开始”按钮时, 会执行 buttonReplay\_Click 事件, 该事件将会调用 Replay 方法, 设置重新开始游戏。Replay 方法的具体代码如下所示:

```

public void Replay(bool ARecordMode, bool AExtended)
{
    if (threadReview != null)
    {
        threadReview.Abort();
        threadReview = null;
    }
    if (AExtended != extended)
        NewTemplets(AExtended);
    reviewing = false;
    playing = true;
    recordMode = ARecordMode;
    Clear();
    StepInfos.Clear();
    afterBrickIndex = (byte)random.Next(brickTemplets.Count);
    afterFacingIndex = (byte)random.Next(brickTemplets[afterBrickIndex].Count);
    if (youxiNext != null) youxiNext.Update(this);
    if (recordMode && !reviewing)
    {
        StepInfos.Add(new StepInfo(0, 0, afterBrickIndex, afterFacingIndex));
        lastRecordTime = Environment.TickCount;
    }
    level = 0;
    score = 0;
    lines = 0;
    stepIndex = -1;
    if (progressBar != null) progressBar.Value = 0;
    NextBrick();
}

```



```

timer.Interval = speeds[level];
timer.Enabled = true;
if (youxiScore != null) youxiScore.Update(this);
if (CanFocus) Focus();
}

```

### 3. Review()处理

单击窗体内的 Review 按钮时, 将会执行 buttonReview\_Click 事件, 该事件将会调用 Review()方法, 设置查看游戏的历史回放进程。具体代码如下所示:

```

public void Review()
{
    if (threadReview != null)
    {
        threadReview.Abort();
        threadReview = null;
    }
    timer.Enabled = false;
    reviewing = true;
    playing = true;
    Clear();
    level = 0;
    score = 0;
    lines = 0;
    stepIndex = 0;
    NextStep();
    NextStep();
    if (progressBar != null)
        progressBar.Maximum = StepInfos.Count;
    threadReview = new Thread(new ThreadStart(DoReview));
    threadReview.Start();
}

```

在执行上述方法时, 先通过 DoReview 方法设置回放速度。DoReview 方法的具体实现代码如下:

```

private void DoReview()
{
    while (reviewing)
    {
        if (stepIndex < 0 || stepIndex >= StepInfos.Count)
        {
            reviewing = false;
            break;
        }
        Thread.Sleep((int)((double)StepInfos[stepIndex].timeTick / reviewSpeed));
        if (!reviewing) break;

        Invoke(new EventHandler(DoInvoke));
    }
    Invoke(new EventHandler(DoInvoke));
    threadReview = null;
}

```

### 4. 载入处理

载入处理的范围比较广, 包括游戏载入和统计数据载入, 在此, 将着重介绍统计数据的载入。具体过程如下。

(1) 通过 LoadFromFile 方法，从指定文件中将玩家的操作记录载入，变量 AFileName 是载入的源文件。LoadFromFile 方法的具体代码如下所示：

```
public void LoadFromFile(string AFileName)
{
    if (!File.Exists(AFileName)) return;
    FileStream vFileStream = new FileStream(AFileName, FileMode.Open, FileAccess.Read);
    LoadFromStream(vFileStream);
    vFileStream.Close();
}
```

(2) 通过 LoadFromStream 方法从指定流中将玩家的操作记录载入，变量 AStream 是指载入的流。LoadFromStream 方法的具体代码如下所示：

```
/// 从流中载入玩家操作记录
/// </summary>
public void LoadFromStream(Stream AStream)
{
    if (AStream == null) return;
    byte[] vBuffer = new byte[3];
    if (AStream.Read(vBuffer, 0, vBuffer.Length) != 3) return;
    if (vBuffer[0] != 116 || vBuffer[1] != 114 || vBuffer[2] != 102) return;
    if (colCount != (byte)AStream.ReadByte()) return;
    if (rowCount != (byte)AStream.ReadByte()) return;
    if (threadReview != null)
    {
        threadReview.Abort(); // 如果正在回放
        threadReview = null;
    }
    timer.Enabled = false;
    playing = false;
    reviewing = false;
    brickTemplets.Clear();
    if (progressBar != null) progressBar.Value = 0;
    int vTempletsCount = AStream.ReadByte();
    for (int i=0; i<vTempletsCount; i++)
    {
        List<List<Point>> templets = new List<List<Point>>();
        int vPointsLength = AStream.ReadByte();
        for (int j=0; j<vPointsLength; j++)
        {
            List<Point> bricks = new List<Point>();
            int vPointCount = AStream.ReadByte();
            for (int k=0; k<vPointCount; k++)
            {
                int vData = AStream.ReadByte();
                if (vData < 0) break;

                bricks.Add(new Point(vData & 3, vData >> 4 & 3));
            }
            templets.Add(bricks);
        }
        brickTemplets.Add(templets);
    }
    StepInfos.Clear();
    vBuffer = new byte[sizeof(int)];
    if (AStream.Read(vBuffer, 0, vBuffer.Length) != vBuffer.Length) return;
    int vStepCount = BitConverter.ToInt32(vBuffer, 0);
    for (int i=0; i<vStepCount; i++)
    {
```

```

StepInfo vStepInfo = new StepInfo();
vStepInfo.param1 = (byte)AStream.ReadByte();
vBuffer = new byte[sizeof(ushort)];
if (AStream.Read(vBuffer, 0, vBuffer.Length) != vBuffer.Length) return;
vStepInfo.timeTick = (ushort)BitConverter.ToInt16(vBuffer, 0);
int vData = AStream.ReadByte();
vStepInfo.command = (byte)(vData & 3);
vStepInfo.param2 = (byte)(vData >> 4 & 3);
StepInfos.Add(vStepInfo);
}
Clear();
Invalidate();
}

```

(3) 通过 SaveToFile 方法将玩家的操作记录保存到指定文件中，变量 AFileName 是保存的文件。SaveToFile 方法的具体代码如下所示：

```

/// 将玩家操作记录保存到文件中
public void SaveToFile(string AFileName)
{
    FileStream vFileStream =
        new FileStream(AFileName, FileMode.Create, FileAccess.Write);
    SaveToStream(vFileStream);
    vFileStream.Close();
}

```

(4) 通过 SaveToStream 方法将玩家的操作记录保存到指定流中，变量 AStream 是保存的流。SaveToStream 方法的具体代码如下所示：

```

/// 将玩家操作记录保存到流中
public void SaveToStream(Stream AStream)
{
    if (AStream == null) return;
    byte[] vBuffer = Encoding.ASCII.GetBytes("trf");
    AStream.Write(vBuffer, 0, vBuffer.Length); // 写头信息
    AStream.WriteByte((byte)colCount);
    AStream.WriteByte((byte)rowCount);
    byte vByte = (byte)brickTemplets.Count;
    AStream.WriteByte(vByte);
    foreach (List<List<Point>> vList in brickTemplets)
    {
        vByte = (byte)vList.Count;
        AStream.WriteByte(vByte);
        foreach (List<Point> vPoints in vList)
        {
            vByte = (byte)vPoints.Count;
            AStream.WriteByte(vByte);
            foreach (Point vPoint in vPoints)
            {
                vByte = (byte)(vPoint.Y << 4 | vPoint.X);
                AStream.WriteByte(vByte);
            }
        }
    }
    AStream.Write(BitConverter.GetBytes(StepInfos.Count), 0, sizeof(int));
    foreach (StepInfo vStepInfo in StepInfos)
    {
        AStream.WriteByte(vStepInfo.param1);
        AStream.Write(BitConverter.GetBytes(vStepInfo.timeTick), 0, sizeof(ushort));
        vByte = (byte)(vStepInfo.param2 << 4 | vStepInfo.command);
    }
}

```



```

        AStream.WriteByte(vByte);
    }
}

```

## 5. 绘制方块处理

绘制方块处理是指在窗体内绘制指定的俄罗斯方块，并且在绘制过程中能够控制它的位置和显示样式。在 C# 程序中，使用 GDI+ 实现窗体绘图功能。GDI+ 是 Graphics Device Interface Plus 的缩写，是公共语言运行时用来创建图形、绘制文本以及进行图形对象操作的技术。GDI+ 提供了在 Windows 窗体和控件上绘制图形、图像和文本的方法，是 C# 的核心应用技术之一。

Graphics 是 GDI+ 的核心，在里面提供了将对象绘制到显示设备的方法。Graphics 可以与特定设备的上下文相关联，来创建图形的对象。Graphics 类封装了绘制直线、曲线、图形、图像和文本的方法，是 GDI+ 实现绘制直线、曲线、图形、图像和文本的类，是 GDI+ 操作的基础类。GDI+ 绘制的实现流程是：首先创建 Graphics 对象；然后通过 Graphics 对象绘制线条、形状或文本。

有如下 3 种方法创建 GDI+ 对象。

在窗体或控件的 Paint 事件处理方法中创建，PaintEventArgs 类参数中包含了对图形对象的引用。例如下面的代码：

```

private void Form1_Paint(object sender, System.Windows.Forms.PaintEventArgs pe)
{
    Graphics g = pe.Graphics;
}

```

调用窗体或控件的 CreateGraphics 方法创建 Graphics 对象，例如下面的代码：

```

Graphics g = pe.Graphics;
g = this.CreateGraphics();

```

通过从 Image 继承的对象创建 Graphics 对象，例如下面的代码：

```

Bitmap mm = new Bitmap(@"D:\123.bmp");
Graphics g = Graphics.FromImage(mm);

```

创建上述 Graphics 对象后，可以用该对象来绘制线条、形状或文本。

本项目中，窗体图像绘制的具体实现流程如下。

(1) 通过 DrawPoint 方法在窗体内绘制一个点的方块，其中变量 AGraphics 是绘制的图像，APoint 变量是绘制的坐标，ABrick 变量是绘制的方块图案。

DrawPoint 方法的具体代码如下所示：

```

public void DrawPoint(Graphics AGraphics, Point APoint, byte ABrick)
{
    if (ImageList == null) return;
    if (ImageList.Images.Count <= 0) return;
    if (APoint.X < 0 || APoint.X >= colCount) return;
    if (APoint.Y < 0 || APoint.Y >= rowCount) return;
    Rectangle vRectangle = new Rectangle(
        APoint.X * brickWidth, APoint.Y * brickHeight, brickWidth, brickHeight);
    AGraphics.FillRectangle(new SolidBrush(BackColor), vRectangle);
    if (ABrick <= 0) return;
    ABrick = (byte)((ABrick - 1) % ImageList.Images.Count);
}

```

```

Image vImage = ImageList.Images[ABrick];
AGraphics.DrawImage(vImage, vRectangle.Location);
}

```

(2) 通过 DrawPoints 方法在窗体内绘制整个点阵, 其中变量 AGraphics 表示绘制的图像。DrawPoints 方法的具体代码如下所示:

```

public void DrawPoints (Graphics AGraphics)
{
    if (ImageList == null) return;
    if (ImageList.Images.Count <= 0) return;
    for (int i=0; i<colCount; i++)
        for (int j=0; j<rowCount; j++)
            DrawPoint(AGraphics, new Point(i, j), points[i, j]);
}

```

(3) 通过 DrawCurrent 方法在窗体内绘制当前被控制的方块, 其中变量 AGraphics 表示要绘制的图像, AClear 设置是否采用清除绘制。DrawCurrent 方法的具体代码如下所示:

```

public void DrawCurrent(Graphics AGraphics, bool AClear)
{
    if (ImageList == null) return;
    if (ImageList.Images.Count <= 0) return;
    foreach (Point vPoint in brickTemplets[brickIndex][facingIndex])
        DrawPoint(AGraphics, new Point(vPoint.X + brickPoint.X,
            vPoint.Y + brickPoint.Y), AClear? (byte)0 : (byte)(brickIndex + 1));
}

```

(4) 通过 DrawNext 方法在窗体内绘制下一个出现的方块, 其中变量 AGraphics 表示要绘制的图像。

DrawNext 方法的具体代码如下所示:

```

public void DrawNext(Graphics AGraphics)
{
    if (AGraphics == null) return;
    if (ImageList == null) return;
    if (ImageList.Images.Count <= 0) return;
    foreach (Point vPoint in brickTemplets[afterBrickIndex][afterFacingIndex])
        DrawPoint (AGraphics, new Point(vPoint.X, vPoint.Y), (byte) (afterBrickIndex + 1));
}

```

(5) 通过 DrawScore 方法在窗体内绘制积分框, 其中变量 AGraphics 表示要绘制的图像。DrawScore 方法的具体代码如下所示:

```

public void DrawScore(Graphics AGraphics)
{
    if (AGraphics == null) return;
    RectangleF vRectangleF = new Rectangle(0, 0, brickWidth * 4, brickHeight);
    StringFormat vStringFormat = new StringFormat();
    vStringFormat.FormatFlags |= StringFormatFlags.LineLimit;
    vStringFormat.Alignment = StringAlignment.Center;
    AGraphics.DrawString("Score", new Font(Font, FontStyle.Bold),
        Brushes.White, vRectangleF, vStringFormat);
    vRectangleF.Offset(0, brickHeight);
    AGraphics.DrawString(score.ToString(), Font,
        Brushes.White, vRectangleF, vStringFormat);
    vRectangleF.Offset(0, brickHeight);
    AGraphics.DrawString("Level", new Font(Font, FontStyle.Bold),
        Brushes.White, vRectangleF, vStringFormat);
}

```



```

vRectangleF.Offset(0, brickHeight);
AGraphics.DrawString(level.ToString(), Font,
    Brushes.White, vRectangleF, vStringFormat);
vRectangleF.Offset(0, brickHeight);
AGraphics.DrawString("Lines", new Font(Font, FontStyle.Bold),
    Brushes.White, vRectangleF, vStringFormat);
vRectangleF.Offset(0, brickHeight);
AGraphics.DrawString(lines.ToString(), Font,
    Brushes.White, vRectangleF, vStringFormat);
}

```

## 6. 游戏过程处理

游戏过程处理是指在窗体内的游戏在玩的过程中的处理控制，即对窗体内方块的移动操作和行数变化的处理。游戏过程处理的具体实现流程如下。

(1) 通过 CheckBrick 方法检查方块是否可以移动或变化到此位置，其中变量 ABrickIndex 表示模板序号，AFacingIndex 表示朝向序号，ABrickPoint 代表位置坐标。

CheckBrick 方法的具体代码如下所示：

```

public bool CheckBrick(byte ABrickIndex, byte AFacingIndex, Point ABrickPoint)
{
    foreach (Point vPoint in brickTemplets[ABrickIndex][AFacingIndex])
    {
        if (vPoint.X + ABrickPoint.X < 0 || vPoint.X + ABrickPoint.X >= colCount)
            return false;
        if (vPoint.Y + ABrickPoint.Y < 0 || vPoint.Y + ABrickPoint.Y >= rowCount)
            return false;
        if (points[vPoint.X + ABrickPoint.X, vPoint.Y + ABrickPoint.Y] != 0)
            return false;
    }
    return true;
}

```

(2) 通过 FreeLine 方法进行消行处理，即操作成功后将方块行删除。FreeLine 方法的具体实现代码如下所示：

```

public void FreeLine()
{
    int vFreeCount = 0;
    for (int j=rowCount-1; j>=0; j--)
    {
        bool vExistsFull = true; // 是否存在满行
        for (int i=0; i<colCount && vExistsFull; i++)
            if (points[i, j] == 0)
                vExistsFull = false;
        if (!vExistsFull) continue;
        #region 图片下移
        Graphics vGraphics = Graphics.FromImage(backBitmap);
        Rectangle srcRect = new Rectangle(0, 0, backBitmap.Width, j * brickHeight);
        Rectangle destRect = srcRect;
        destRect.Offset(0, brickHeight);
        Bitmap vBitmap = new Bitmap(srcRect.Width, srcRect.Height);
        Graphics.FromImage(vBitmap).DrawImage(backBitmap, 0, 0);
        vGraphics.DrawImage(vBitmap, destRect, srcRect, GraphicsUnit.Pixel);
        vGraphics.FillRectangle(new SolidBrush(BackColor), 0, 0,
            backBitmap.Width, brickHeight);
        #endregion 图片下移
        lines++;
    }
}

```



```

vFreeCount++;
for (int k=j; k>=0; k--)
{
    for (int i=0; i<colCount; i++)
        if (k == 0)
            points[i, k] = 0;
        else
            points[i, k] = points[i, k - 1];
}
j++;
}
score += scoress[vFreeCount];
if (vFreeCount > 0)
{
    level = Math.Min(lines/30, speeds.Length-1);
    timer.Interval = speeds[level];
    Invalidate();
}
if (youxiScore != null) youxiScore.Update(this);
}

```

(3) 通过 `BrickOperate` 方法设置对方块的变化处理,即变化方块的旋转位置。其中参数 `ABrickOperates` 代表变化指令,具体含义是上、下、左、右移动和翻转处理。`BrickOperate` 方法的具体实现代码如下所示:

```

public bool BrickOperate(BrickOperates ABrickOperates)
{
    byte vFacingIndex = facingIndex;
    Point vBrickPoint = brickPoint;
    switch (ABrickOperates)
    {
        case BrickOperates.boTurnLeft:
            vFacingIndex = (byte)((vFacingIndex + 1) % brickTemplets[brickIndex].Count);
            break;
        case BrickOperates.boTurnRight:
            vFacingIndex = (byte)((brickTemplets[brickIndex].Count
                + vFacingIndex - 1) % brickTemplets[brickIndex].Count);
            break;
        case BrickOperates.boMoveLeft:
            vBrickPoint.Offset(-1, 0);
            break;
        case BrickOperates.boMoveRight:
            vBrickPoint.Offset(+1, 0);
            break;
        case BrickOperates.boMoveDown:
            vBrickPoint.Offset(0, +1);
            break;
        case BrickOperates.boMoveBottom:
            vBrickPoint.Offset(0, +1);
            while (CheckBrick(brickIndex, vFacingIndex, vBrickPoint))
                vBrickPoint.Offset(0, +1);
            vBrickPoint.Offset(0, -1);
            break;
    }
    if (CheckBrick(brickIndex, vFacingIndex, vBrickPoint))
    {
        if (playing && recordMode && !reviewing)
        {
            StepInfos.Add(new StepInfo(Environment.TickCount - lastRecordTime,
                2, (byte)ABrickOperates, 0));

```

```

        lastRecordTime = Environment.TickCount;
    }

    Graphics vGraphics = Graphics.FromImage(backBitmap);
    DrawCurrent(vGraphics, true);
    facingIndex = vFacingIndex;
    brickPoint = vBrickPoint;
    DrawCurrent(vGraphics, false);
    if (ABrickOperates == BrickOperates.boMoveBottom) Downfall();
    else Invalidate();
}
else if (ABrickOperates == BrickOperates.boMoveDown)
{
    if (playing && recordMode && !reviewing)
    {
        StepInfos.Add(new StepInfo(Environment.TickCount - lastRecordTime,
            2, (byte)ABrickOperates, 0));
        lastRecordTime = Environment.TickCount;
    }
    Downfall();
}
return true;
}

```

(4) 通过 NextBrick 方法预览显示下一个出现的方框，即下一个要操作控制的方框。方法 NextBrick 的具体实现代码如下所示：

```

private void NextBrick()
{
    brickIndex = afterBrickIndex;
    facingIndex = afterFacingIndex;
    brickPoint.X = colCount / 2 - 1;
    brickPoint.Y = 0;
    afterBrickIndex = (byte)random.Next(brickTemplets.Count);
    afterFacingIndex = (byte)random.Next(brickTemplets[afterBrickIndex].Count);
    if (playing && recordMode && !reviewing)
    {
        StepInfos.Add(new StepInfo(0, 1, afterBrickIndex, afterFacingIndex));
        lastRecordTime = Environment.TickCount;
    }
    if (youxiNext != null && afterBrickIndex != brickIndex) youxiNext.Update(this);
    DrawCurrent(Graphics.FromImage(backBitmap), false);
    if (!CheckBrick(brickIndex, facingIndex, brickPoint))
    {
        if (playing && recordMode && !reviewing)
        {
            StepInfos.Add(new StepInfo(0, 3, 0, 0));
            lastRecordTime = Environment.TickCount;
        }
        GameOver();
    }
    Invalidate();
}

```

(5) 通过 Downfall 方法执行方块落底后的处理，如果某行都满，则执行行的删除方法程序。Downfall 方法的具体实现代码如下所示：

```

private void Downfall()
{
    foreach (Point vPoint in brickTemplets[brickIndex][facingIndex])

```

```

        points[vPoint.X + brickPoint.X, vPoint.Y + brickPoint.Y] =
            (byte) (brickIndex + 1);
        FreeLine();
        if (playing && !reviewing) NextBrick();
    }
    private void ImageListRecreateHandle(object sender, EventArgs e)
    {
        DoChange();
    }
    private void DetachImageList(object sender, EventArgs e)
    {
        ImageList = null;
    }
    public ImageList ImageList
    {
        get
        {
            return imageList;
        }
        set
        {
            if (value != imageList)
            {
                EventHandler handler = new EventHandler(ImageListRecreateHandle);
                EventHandler handler2 = new EventHandler(DetachImageList);
                if (imageList != null)
                {
                    imageList.RecreateHandle -= handler;
                    imageList.Disposed -= handler2;
                }
                imageList = value;
                if (value != null)
                {
                    brickWidth = ImageList.ImageSize.Width;
                    brickHeight = ImageList.ImageSize.Height;
                    DoChange();
                    if (!playing && !reviewing) GameOver();
                    if (youxiNext != null)
                    {
                        youxiNext.BackColor = BackColor;
                        youxiNext.SetSize(brickWidth * 4, brickHeight * 4);
                        youxiNext.Update(this);
                    }
                    value.RecreateHandle += handler;
                    value.Disposed += handler2;
                }
            }
        }
    }
}

```

(6) 定义 TetrisScore 属性，设置得到游戏的积分。具体实现代码如下所示：

```

private void DetachyouxiScore(object sender, EventArgs e)
{
    youxiScore = null;
}
public youxiScore TetrisScore
{
    get
    {
        return youxiScore;
    }
}

```



```

    }
    set
    {
        if (value != youxiScore)
        {
            EventHandler handler = new EventHandler(DetachyouxiScore);
            if (youxiScore != null) youxiScore.Disposed -= handler;
            youxiScore = value;
            if (value != null)
            {
                value.SetSize(4*brickWidth, 6*brickHeight);
                value.Update(this);
                value.Disposed += handler;
            }
        }
    }
}

```

(7) 定义 ProgressBar, 设置实现回放进度条的处理。具体实现代码如下所示:

```

private void DetachProgressBar(object sender, EventArgs e)
{
    progressBar = null;
}
/// <summary>
/// 回放进度条
/// </summary>
public ProgressBar ProgressBar
{
    get
    {
        return progressBar;
    }
    set
    {
        if (value != progressBar)
        {
            EventHandler handler = new EventHandler(DetachProgressBar);
            if (progressBar != null) progressBar.Disposed -= handler;
            progressBar = value;
            if (value != null)
            {
                progressBar.Minimum = 0;
                progressBar.Maximum = StepInfos.Count;
                progressBar.Value = stepIndex<0? 0 : stepIndex;
                value.Disposed += handler;
            }
        }
    }
}

```

当用 C# 实现 WinForm 项目时, 经常需要用到进度条(ProgressBar)来显示进度信息。在大型软件项目中, 建议使用多线程技术实现进度条效果。因为如果不采用多线程控制进度条, 很容易造成窗口假死(无法实时看到进度信息)的情况。

下面结合一个简单例子进行讲解。

首先引用多线程对象, 代码如下:

```
using System.Threading;
```

然后定义一个代理，用于更新 ProgressBar 的值(Value)，代码如下：

```
//更新进度列表
private delegate void SetPos(int ipos);
```

然后编写进度条值更新函数(参数必须跟声明的代理参数一样)，代码如下：

```
private void SetTextMessage(int ipos)
{
    if (this.InvokeRequired)
    {
        SetPos setpos = new SetPos(SetTextMessage);
        this.Invoke(setpos, new object[] { ipos });
    }
    else
    {
        this.label1.Text = ipos.ToString() + "/100";
        this.progressBar1.Value = Convert.ToInt32(ipos);
    }
}
```

然后编写按钮单击事件的响应，代码如下：

```
private void button1_Click(object sender, EventArgs e)
{
    Thread fThread = new Thread(new ThreadStart(SleepT));
    //开辟一个新的线程
    fThread.Start();
}
```

然后编写 SleepT 延时函数，代码如下：

```
private void SleepT()
{
    for (int i=0; i<500; i++)
    {
        System.Threading.Thread.Sleep(100);

        //单纯的执行延时
        SetTextMessage(100*i/500);
    }
}
```

到此为止，一个简单的多线程进度条程序就做好了。

(8) 定义各个热键事件处理方法，并通过 OnKeyDown 方法设置按键按下时所要发生的对方块的控制方式。例如左移、右移和变换等方式。具体实现代码如下所示：

```
protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    if (backBitmap != null) e.Graphics.DrawImage(backBitmap, 0, 0);
}
protected override bool IsInputKey(Keys keydata)
{
    return (keydata == Keys.Down) || (keydata == Keys.Up)
        || (keydata == Keys.Left) || (keydata == Keys.Right)
        || (keydata == Keys.Escape) || base.IsInputKey(keydata);
}
protected override void OnMouseDown(MouseEventArgs e)
{
    base.OnMouseDown(e);
}
```

```

        if (CanFocus) Focus();
    }
    protected override void Dispose(bool disposing)
    {
        if (threadReview != null) threadReview.Abort();
        base.Dispose(disposing);
    }
    protected override void OnKeyDown(KeyEventArgs e)
    {
        base.OnKeyDown(e);
        if (!playing || reviewing) return;
        switch (e.KeyCode)
        {
            case Keys.A: // 左移
            case Keys.Left:
                BrickOperate(BrickOperates.boMoveLeft);
                break;
            case Keys.D: // 右移
            case Keys.Right:
                BrickOperate(BrickOperates.boMoveRight);
                break;
            case Keys.W: // 左旋
            case Keys.Up:
                BrickOperate(BrickOperates.boTurnLeft);
                break;
            case Keys.Back: // 右旋
            case Keys.F:
                BrickOperate(BrickOperates.boTurnRight);
                break;
            case Keys.Down: // 向下
            case Keys.S:
                BrickOperate(BrickOperates.boMoveDown);
                break;
            case Keys.Enter: // 直下
            case Keys.Space:
            case Keys.End:
            case Keys.J:
                BrickOperate(BrickOperates.boMoveBottom);
                break;
        }
    }
}

```

(9) 通过 youxiNext 类来显示下一个出现的方块控件，分别通过方法 youxiNext、Clear、SetSize 和 OnPaint 来设置下一个方块。youxiNext 类的具体实现代码如下所示：

```

public class youxiNext : Control
{
    public youxiNext()
    {
        BackColor = Color.Black;
        base.SetStyle(ControlStyles.OptimizedDoubleBuffer, true);
    }
    private Bitmap backBitmap;
    public void Clear()
    {
        Graphics vGraphics = Graphics.FromImage(backBitmap);
        vGraphics.FillRectangle(new SolidBrush(BackColor), vGraphics.ClipBounds);
    }
    public void Update(youxiControl AyouxiControl)
    {

```



```

        if (AyouxiControl == null) return;
        Clear();
        AyouxiControl.DrawNext(Graphics.FromImage(backBitmap));
        Invalidate();
    }
    public void SetSize(int AWidth, int AHeight)
    {
        Width = AWidth;
        Height = AHeight;
        backBitmap = new Bitmap(AWidth, AHeight);
    }
    protected override void OnPaint(PaintEventArgs e)
    {
        base.OnPaint(e);
        if (backBitmap != null) e.Graphics.DrawImage(backBitmap, 0, 0);
    }
}

```

(10) 通过 `youxiScore` 类来显示游戏的当前积分，分别通过方法 `youxiScore`、`Clear`、`Update`、`SetSize` 和 `OnPaint` 来进行积分的处理设置和显示。`youxiScore` 类的具体实现代码如下所示：

```

/// 显示积分信息的控件
public class youxiScore : Control
{
    private Bitmap backBitmap;
    public youxiScore()
    {
        BackColor = Color.Black;
        base.SetStyle(ControlStyles.OptimizedDoubleBuffer, true);
    }
    public void Clear()
    {
        Graphics vGraphics = Graphics.FromImage(backBitmap);
        vGraphics.FillRectangle(new SolidBrush(BackColor), vGraphics.ClipBounds);
    }
    public void Update(youxiControl AyouxiControl)
    {
        if (AyouxiControl == null) return;
        Clear();
        AyouxiControl.DrawScore(Graphics.FromImage(backBitmap));
        Invalidate();
    }
    public void SetSize(int AWidth, int AHeight)
    {
        Width = AWidth;
        Height = AHeight;
        backBitmap = new Bitmap(AWidth, AHeight);
    }
    protected override void OnPaint(PaintEventArgs e)
    {
        base.OnPaint(e);
        if (backBitmap != null) e.Graphics.DrawImage(backBitmap, 0, 0);
    }
}


```

到此为止，完成了整个项目的开发工作。

很多人在具体编码之前，头脑中会有很多思路；但是，在具体编码时，却发现连一行代码也写不出来。这种情况在很多初学者身上都发生过。要想解决这种问题，那就只能学

好基本的语法知识，并且要学得扎实一点。

## 1.7 测试运行

视频讲解  光盘：视频\第 1 章\测试运行.avi

在 Visual Studio 2013 中开始调试项目，运行后的主界面如图 1-18 所示；运行中的界面如图 1-19 所示；游戏设置菜单界面如图 1-20 所示。



图 1-18 主界面的效果



图 1-19 运行中的界面



图 1-20 游戏设置菜单界面

## 第2章 多媒体学习社区

高帆项目开发

随着 Internet 的普及，网络已成为人们学习、工作、生活不可或缺的一部分。网络学习是信息化社会学习的主要途径之一，而网络学习受到广大学习者的青睐是因为在网络上学习资源丰富、学习时间灵活，还可以通过互动交流的方式进行学习。互动交流是非常有效的网络学习手段，对学习者的创新能力和认知能力的发展，有不可低估的作用。

本章我们将通过开发一个多媒体学习网站——互动媒体学习社区，介绍利用 ASP.NET + Access 快速开发互动网站的方法。



### 赠送的超值电子书

- 011 不变的常量
- 012 隐式转换
- 013 显式转换
- 014 装箱与拆箱
- 015 枚举
- 016 结构
- 017 数组
- 018 Console 类
- 019 Convert 类
- 020 常量和变量的区别



## 2.1 修炼自身

视频讲解 光盘：视频\第2章\修炼自身.avi

作为一名合格的程序员，需要具备基本的程序员修养水平。要想提高修养水平，就需要耐得住寂寞，经得住诱惑，懂得忘记，学会放弃，学会适应。在本节的内容中，我们将详细讲解提高程序员自身修养的基本知识。

### 2.1.1 “码农”和“高大上”

世界是丰富多彩的。程序员群体虽然都是高级脑力劳动者，都是辛勤耕耘的“码农”，但总会有那么一些人能够从众多的码农中脱颖而出，成为“高大上”的架构师或 CTO，获得令人羡慕的薪资待遇。与之相对应的是，普通码农可能总要不休止地绝对服从、拼命加班，而得到的却只是相对微薄的工资。

那么，高薪族为什么能成为“高大上”的架构师呢？这是因为他们具备了普通码农所没有的特质。在众多特质中，自身修养这一硬实力是实实在在的，是看得见、摸得着的。

程序员是一种创造性的工作，必须注重个人的修养。IT 界的程序员可以分为两类，具体说明如下。

(1) 一类是普通码农，也被称为见习程序员。这类人的特点是任劳任怨，因为还处于成长阶段，所以主要工作是整天加班修改代码。

(2) 一类是牛人大佬，也被称为程序员中的“高大上”。他们已经积累了丰富的编程经验，写几小时的代码可能胜过普通码农们写几个星期的代码。

在当前的 IT 界中，普通码农占大多数。能够成为码农，通过脑力劳动把自己的聪明才智贡献出来，是幸运的。码农只要不断提升自身的技术修养，勇攀事业高峰，就能成长为合格的程序员，进而有希望变成人人都羡慕的“高大上”。

### 2.1.2 赢在自身——快速提升自身修养

很明显，程序员都想成为金字塔顶端的“高大上”一族。但现实对“高大上”的架构师和 CTO 的要求更高，下面列出一些要点。

#### (1) 掌握基础

对于任何行业、任何工作来说，融会贯通才是获得成功的关键。一个人想要成为优秀程序员，就必须有坚实的基础。对核心理念的深刻理解会帮助我们用最好的方法设计和实施出最完美的方案。如果感觉到还没有掌握核心的计算机科学知识，或者感觉对某种编程语言方面的知识点的理解还欠缺，现在就可以回顾基础知识，这一点儿都不晚。

#### (2) 尽量写出简单易懂、有逻辑性的代码

编写的代码要保持短小而精悍的特点，尽量编写有逻辑性的代码，避免复杂化。有时，人们写复杂的代码的原因仅仅是为了展示他们有能力写出这样的代码。其实，简单而富有

逻辑的代码不但问题更少，而且更容易扩展。

### (3) 花更多的时间分析问题，将会花更少的时间去解决问题

花更多的时间理解和分析问题，然后再设计方案，就会发现剩下的事情已经很容易了。设计不是说要用建模语言和工具，可以是仅仅看看天空，在脑子里构思就行。那些一遇到问题就开始敲代码的人，往往会最终偏离需求。

### (4) 成为第一个检查你的代码的人

这一点虽然有一点难度，但是试着在其他人修改你的代码之前修改它，随着时间的推移，就会写出几乎没有 Bug(错误或缺陷)的代码。你需要对自己的代码做没有任何偏见的检查，也不要犹豫让其他人来检查你的代码。与其他优秀的程序员一起工作，虚心倾听别人的意见，能够使自己也成长为一名优秀的程序员。

### (5) 养成阅读文档的习惯

阅读很多文档是作为优秀程序员的必备习惯之一。文档可能是产品说明书、JSR、API 说明、教程等。阅读文档可以帮助我们获得必要的基础知识，写出更好的代码。

### (6) 及时把握技术风向标

IT 界的新技术层出不穷，新版本、新工具和新语言充斥我们耳边。一项新的技术往往能够彻底颠覆一个行业，例如 Android 彻底颠覆了智能手机世界。要想迅速成为一名优秀的程序员，并且能够脱颖而出，就需要具备随时学习新技术的能力。

学习新技术的好处多多，例如使我们的知识面更广，能够对各项开发技术进行横向比较，使自己的开发技术更加融会贯通。更为重要的是，学习新技术让我们能够站在技术前沿的最高点，不但提高了自己的生存技能，而且也能够做到与他人不同，能够更快地从整个团队中甚至在整個公司中脱颖而出。

### (7) 不要迷失在快速更迭的科技世界中

及时把握技术风向标并不是意味着每出现一门新技术，就马上投入到学习中去。在 IT 行业中，会经常遇见许多人，他们对现在的工作不满，甚至离开现在的工作去追寻新的工作，理由是因为他们想要学习“最新的科技”。

虽然我们每天听到的都是新工具、接口、框架，说能让程序更简单、速度更快(在科技界中这是司空见惯的，并且会一直如此)，但是，最基本的、最核心的科技变化其实比那些框架、工具和接口的变化小得多。

举个例子，在 Java 企业级应用中，每个星期都会出现新的框架，但是核心的技术是不变的，例如基于“客户端-服务器端”的请求、MVS 模式、数据源绑定、XML 解析等。

所以，要花工夫去学习核心概念，而不是去担忧日新月异的框架和新出现的工具。只要具备了核心技术的基础，就会发现需要学习新的框架、工具以及接口时，是很容易的。

### (8) 阅读别人的代码，研读大师的作品

如果你不能吸收 IT 界前辈大师的经验和知识，自己就永远都无法成为一位大师。


成为大师的方法之一是，找到一位大师，让其倾囊传授其所知。有这种可能么？当然有，虽然机会不大，而且你必须非常走运，才能拜导师。

其实，大师们的经验和精华就在他们所编写的代码中！我们要做的就是去阅读他们的代码。打个比方，要想成为一名卓越的木匠，就得观察大量结构优良的家具。大师们的源码少则几行，多则上万行。例如 Linux 是全世界大师们的呕心力作，而 Android 更是软件巨



头谷歌公司开发大师们的心血之作。通过阅读他们的代码，分析他们建立的架构，在吸收消化之后，我们就离大师不远了。

## 2.2 开发背景简介

视频讲解  光盘：视频\第 2 章\开发背景简介.avi

信息化是当今世界经济和社会发展的趋势。互联网的发展，不仅改变了人们的工作和生活方式，也改变了教育和学习方式。

计算机网络是计算机技术和通信技术紧密结合的产物，它涉及通信与计算机两个领域。它的诞生，使计算机体系结构发生了巨大的变化，在当今社会经济中起着非常重要的作用，对人类社会的进步做出了巨大的贡献。

网络技术的发展和国际互联网在全球的开通，改变了人们的生活和工作方式，以及学习方式，甚至也改变了思维方式。传统意义上的教学方式正由于因特网的飞速发展和广泛应用而产生着质的变化。基于计算机数字技术的多媒体教学，已经发展成为综合利用计算机网络通信和多媒体技术，以因特网为传播媒介，对远方的学生进行交互式教学的网络教育，它不但改变了传统的教学模式和手段，而且也从本质上改变了传统的教学思想和观念，促进了教育的终身化、现代化、社会化和国际化。

在网络上丰富多彩的学习资源中，传统的文档和图片已不能满足学习者的需求，多媒体资源教学网通过一个虚拟的网络学习环境，让学习者可以观看或发布视频教程和语音教程，进行相互学习和交流。在 Internet 上的学习网站不受时空的限制，自由性比较强，能够实现各取所需，使学习既方便又轻松。

## 2.3 系统设计分析

视频讲解  光盘：视频\第 2 章\系统设计分析.avi

软件项目开发的第一步，是进行系统设计分析和项目需求分析，在本节的内容中，将详细讲解本点播系统设计的具体分析工作，为步入后面的具体编码工作打下基础。

### 2.3.1 互动媒体学习社区的优势

互动媒体学习社区的最大优势不是最快，而是更多、更深，它为学习者提供了无限大的选择空间，通过多种媒体表现形式，使学习者可以选择最适合自己的学习方式。互动媒体学习社区是传统课堂的延伸和拓展，是课后学习的一个良好平台，与学校相比，有着更好的学习气氛，与平常的网络学习系统相比，则又多了些针对性的纵深辅导。

在现实应用中，互动媒体学习社区的魅力主要表现在如下 3 个方面。

#### (1) 内容选项丰富

凡是稍具规模的学科网站，一般都具有良好的网络学习界面，互动媒体学习社区向学习者提供的学习方法有 5 种以上，学习者可以通过选择适合自己学习方式。如时下流行的



视频媒体,因其图、文、声、像俱全,使学习过程变得活泼有趣,从而可以使学习者能够更加投入地学习。

### (2) 方便学习和交流

读者可在学科网站中的留言板等功能模块中,方便地与其他学友进行互动交流,而且这种交流有不受地域限制、友善和人性化的特点。

### (3) 功能齐全

在系统中,学习者可进行学科授课计划、学习课程简介等信息的查看,以及相关课件的下载,视频教程的观看和下载等操作。

## 2.3.2 系统的特点

本网站主要是在现有的正常的网上学习的基础上,增加了语音视频的学习内容,而且学习者还可以上传、下载视频,从而可以提高学习者的学习兴趣。

网站把大量信息的人工管理转变为计算机管理,简化了网站管理员的工作,提高了管理的效率,同时也更加方便用户学习。

本系统采用 Access 关系数据库对数字化信息资源进行组织和管理。项目使用 Microsoft Visual Studio 2013 进行开发。信息平台的前台和后台操作采用 B/S 结构,以增强系统的安全保密性、稳定性和易操作性。所使用的开发语言是 C#,这是目前最完备的面向对象语言。

Access 是一套简洁、快速的数据库管理系统,它提供了多人使用的管理模式。同时,Microsoft Visual Studio 2013 开发平台和 Access 同是微软公司的产品,因而具有良好的整合性。系统采用高度集成的模块结构,将所有的模块整合到一个通用的中央数据库中。

概括地说,本系统具有下列特点。

### (1) 完备的学习功能

系统设计的学习功能模块,可分为公告通知、语音,视频课程观看、用户注册、在线留言、语音、视频课程下载、论坛讨论等,操作功能健全而不冗余,简约而无遗漏。

### (2) 科学的学习模式

系统采用多种学习方式,学习者可结合自己的情况灵活多样地进行检索,能轻松地找到适合自己的学习方法;通过学习者意见反馈系统,管理员可以随时把握学习者的各种新的需求并及时与学习者进行网上互动交流;通过完善的信息发布系统,学习者可以及时得到网站最新动态和新闻,掌握行业动向;后台的结构化管理模式,涵盖了学科介绍、视频课程、语音视频、留言板、论坛等,为学校的电子化管理提供了一套良好的管理模式。

### (3) 人性化的操作界面

一个网站,要想吸引用户经常光顾,界面的美观性就显得非常重要。本互动媒体学习社区通过专业美工的精心打造,让学科网站的设计充分体现出简约、实用和美观的风格。

### (4) 完善的安全机制

拥有独立的密码校验功能,可以确保用户和网站的数据有较好的安全性。

## 2.3.3 系统目标

根据需求分析的描述以及实际考察,确定网站需要实现的功能如下:

- 操作简单方便、界面简洁美观。
- 注册功能。用户通过注册成为网站会员。
- 发布下载教程。对会员提供发布教程和下载教程的功能。
- 密码找回功能。当会员忘记密码时，可以通过此功能找回。
- 留言功能。通过留言功能进行互动交流。
- 查询功能。用户通过查询，可以快速找到需要的教程。
- 后台管理功能。管理员可以通过后台界面，进行网站的维护和管理。
- 系统运行稳定，安全可靠。

### 2.3.4 确定设计方案——B/S 体系结构

本系统所面向的对象是广大的互联网用户。因此，将要采用比较流行的 B/S 三层架构，如图 2-1 所示。

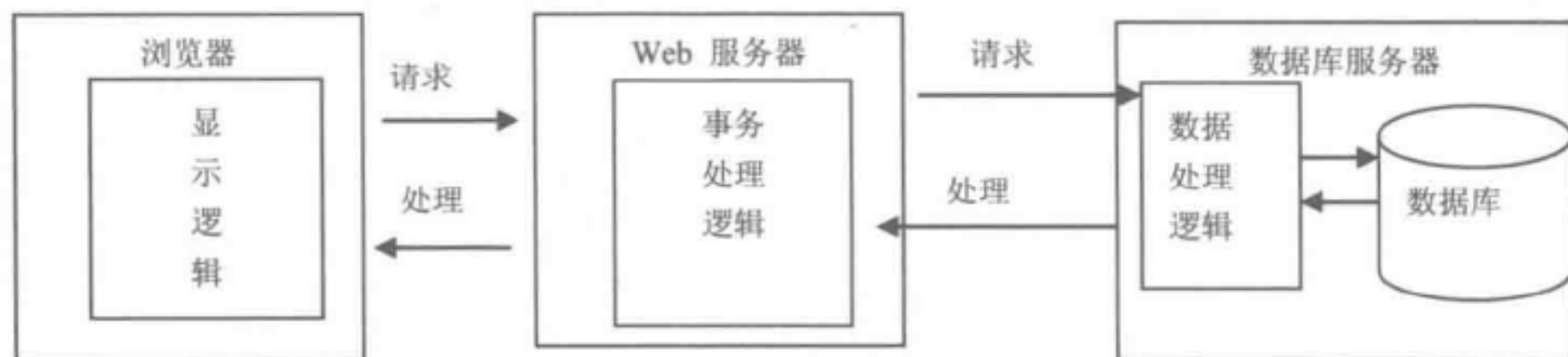


图 2-1 B/S 三层架构

B/S 体系结构与 C/S 体系结构相比，不仅具有其全部的优点，而且又有 C/S 体系结构所不具备的独特优势，具体说明如下。

- 开放的标准：B/S 所采用的标准都是开放的、非专用的，是经过标准化组织所确定而非单一厂商所制定的，保证了其应用的通用性和跨平台性。
- 较低的开发和维护成本：B/S 的应用只需在客户端安装通用的浏览器即可，维护和升级工作都在服务器端进行，不需对客户端进行任何改变，故而大大降低了开发和维护的成本。
- 用户使用简单，界面友好：B/S 的用户界面都在统一的浏览器上，浏览器易于使用、界面友好，又因为它不再负责数据的存取和复杂数据计算等任务，只需要进行显示，因而大大降低了对客户端的要求。

从以上的分析与比较中可以看出，B/S 模式具有 C/S 模式无法替代的优越性，它简化了系统的开发和维护，并且特别适用于网上信息发布。因此，我们所开发的互动媒体学习社区采用了基于 B/S 模式的体系结构。

## 2.4 需求分析

视频讲解 光盘：视频\第 2 章\需求分析.avi

“需求分析”是指对要解决的问题进行详细的分析，弄清楚问题的要求，包括需要输



入什么数据、要得到什么结果、最后应输出什么。可以这么说，在软件工程中的“需求分析”就是确定要计算机“做什么”，要达到什么样的效果。在开发软件项目的过程中，“需求分析”工作是在系统开发之前必须做的重要工作之一。在本节的内容中，将详细讲解本项目需求分析工作的具体过程。

## 2.4.1 可行性分析

### (1) 风险分析

① 系统软件和硬件的风险：尽管采用了性能较高的硬件设备和较稳定的系统软件，但网站仍存在因软硬件崩溃而带来的风险。解决办法之一是定期备份数据，以降低风险。

② 计划的拖延：网站的开发存在因计划拖延带来的风险。

### (2) 技术可行性

大学中的“数据库系统概论”、“计算机操作系统”、“计算机网络”等多门学科为整个开发提供了坚实的基础。C#是一种相对简单的语言，Access 也是一种简单的数据库，所以技术难度并不高。

### (3) 操作可行性

该网站如果投入使用，预期可以做到界面友好、管理方便、使用简单，管理人员经过培训后，也是完全能够使用本网站来管理教程的相关信息的。

### (4) 经济可行性

该系统可以运行于现在市场上出售的各种个人电脑中，系统成本主要集中在系统的开发上。当系统投入运行后，可以实现在网上学习的功能，所带来的效益远远大于系统软件的开发成本，在经济上是完全可行的。

## 2.4.2 功能分析

互动媒体学习社区系统从两种用户角度进行功能划分。

### (1) 会员部分。

- 学科信息展台：为会员提供学科操作平台。
- 下载视频：会员对视频进行下载操作。
- 下载语音：会员进行语音下载操作。
- 观看视频：会员观看学科的视频教程。
- 网站论坛：用于会员对网站的学科内容和服务提建议或交流的平台。

### (2) 管理员部分。

- 管理员账号管理：添加用户；编辑用户。
- 编辑视频：编辑视频简介；上传学科视频。
- 编辑语音：编辑语言简介；上传语音课程。
- 留言管理：查看留言信息；回复留言信息；删除留言。
- 公告管理：发布公告；删除公告。



## 2.4.3 业务流程

互动媒体学习社区网站的业务流程如图 2-2 所示。

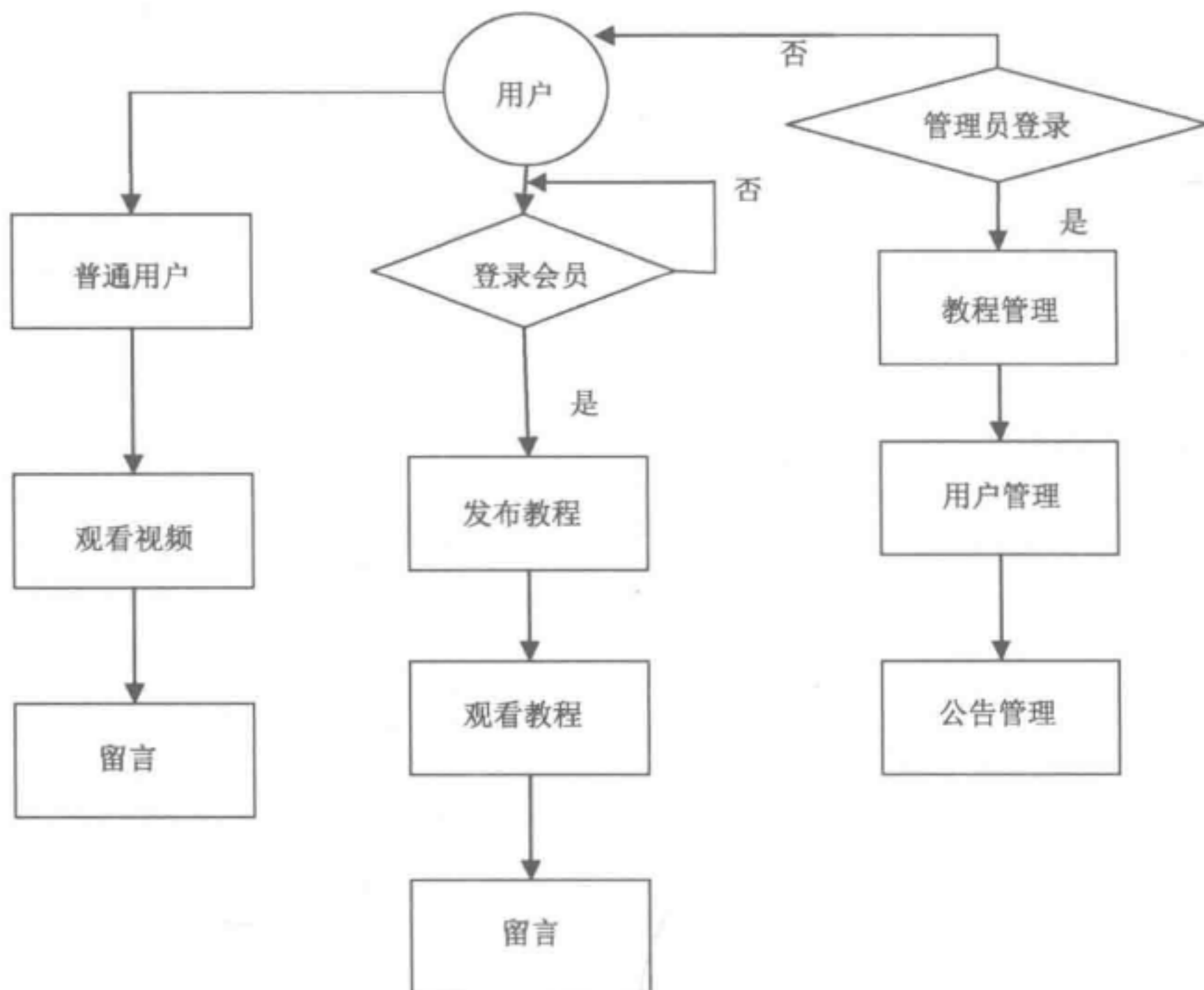


图 2-2 业务流程

## 2.5 总体设计

视频讲解 光盘：视频\第 2 章\总体设计.avi

本系统所面向的对象是广大的互联网用户。因此，将要采用比较流行的 B/S 三层架构。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统开发、维护和使用。

客户机上只须安装一个浏览器(Browser)，如 IE 浏览器，服务器上安装 Access 或 SQL Server 等数据库即可。浏览器通过 Web Server 与数据库进行数据交互。

根据互动媒体学习社区的特点，可以将其分为前台和后台两个部分设计。

前台主要实现发布教程(发布视频或语音教程、查看已发布的语音教程、查看已发布的视频教程)、浏览教程(浏览视频或语音教程、发布留言)、登录功能、查询功能。

后台主要实现公告管理(管理公告、发布公告)、教程管理(发布教程、管理视频教程、管理语音教程)、用户管理等功能。

互动媒体学习社区的前台系统的功能框架如图 2-3 所示。

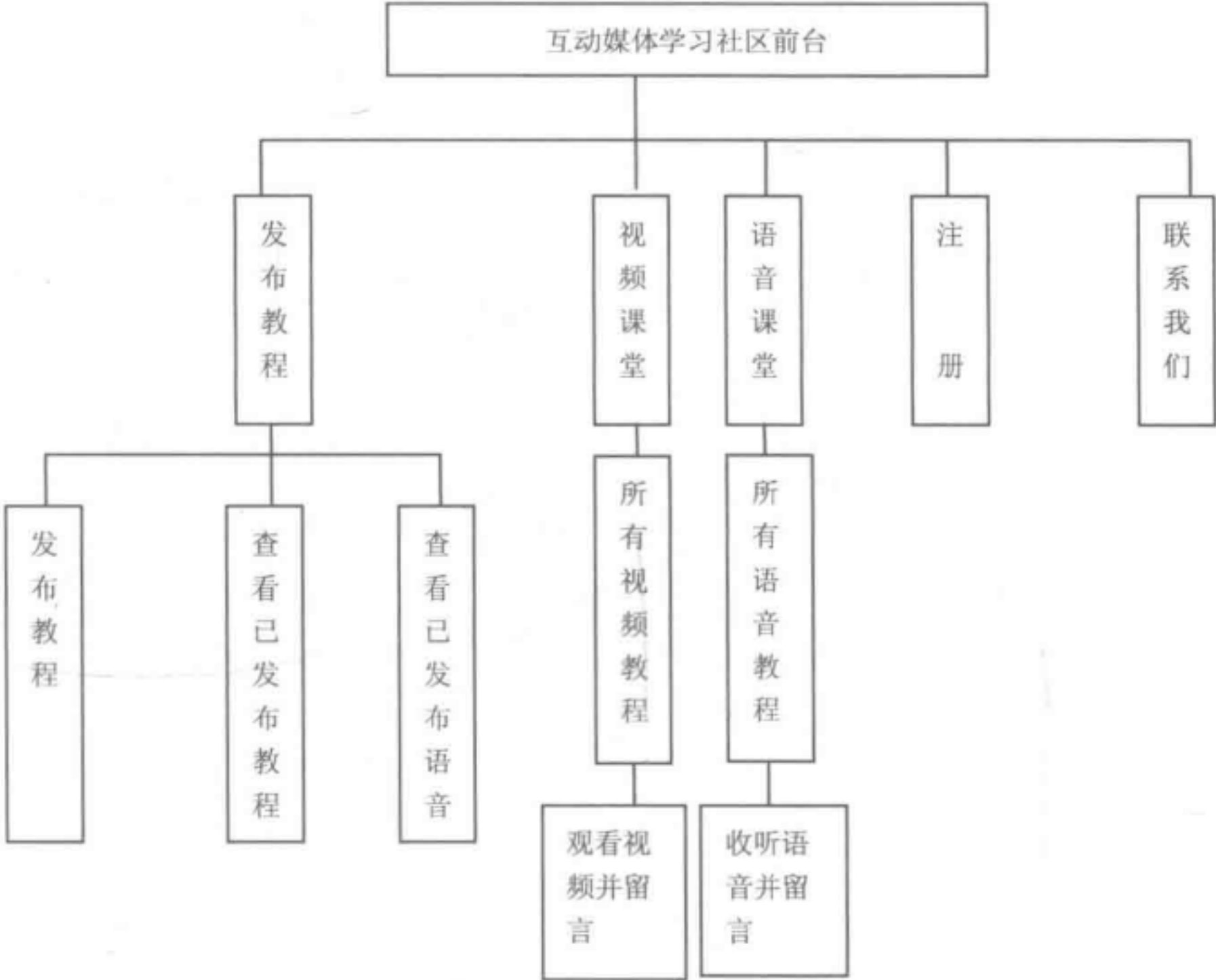


图 2-3 前台系统的功能框架

互动媒体学习社区的后台功能框架如图 2-4 所示。

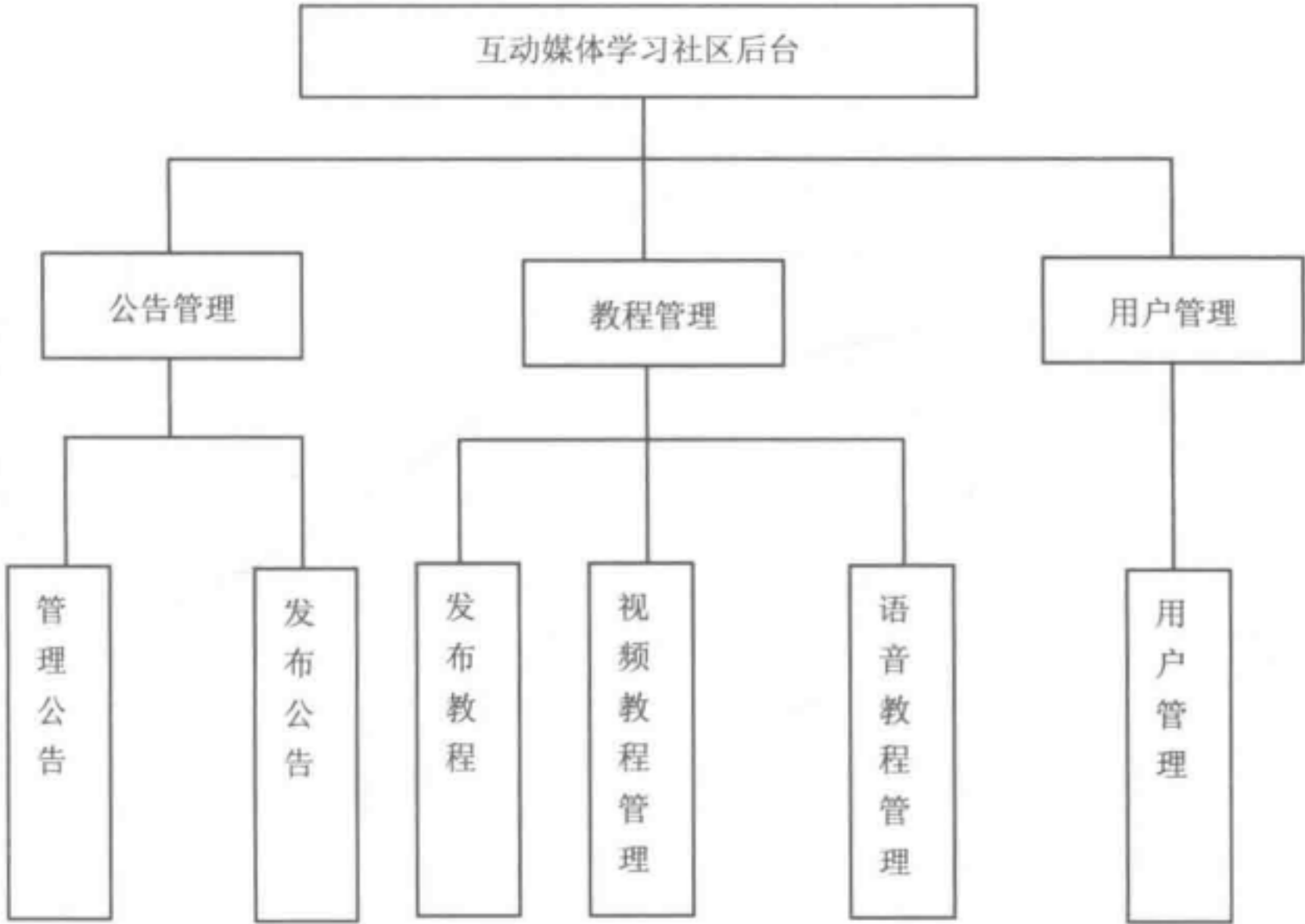


图 2-4 后台系统的功能框架

## 2.6 系统预览

视频讲解 光盘：视频\第 2 章\系统预览.avi

互动媒体学习社区网由多个页面组成，下面仅列出几个典型页面，其他页面参见光盘中的源程序。首页效果如图 2-5 所示，主要实现导航、最新教程、教程排行、公告信息、登录功能和搜索功能。



图 2-5 系统首页

查看教程页面的效果如图 2-6 所示，主要实现观看视频教程和发布留言的功能。

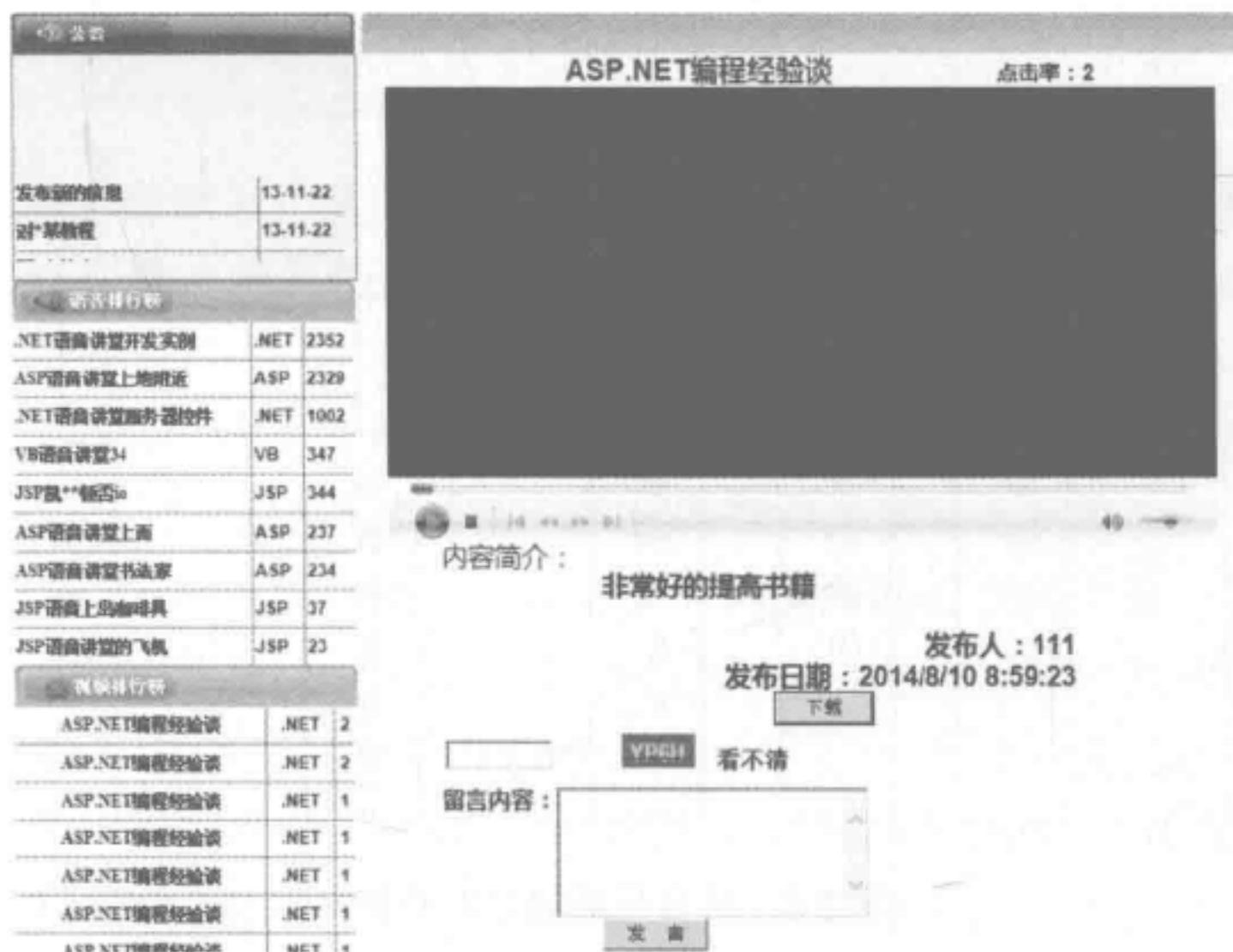


图 2-6 查看教程的页面



发布教程页面如图 2-7 所示，主要实现发布教程、视频教程管理和语音教程管理。

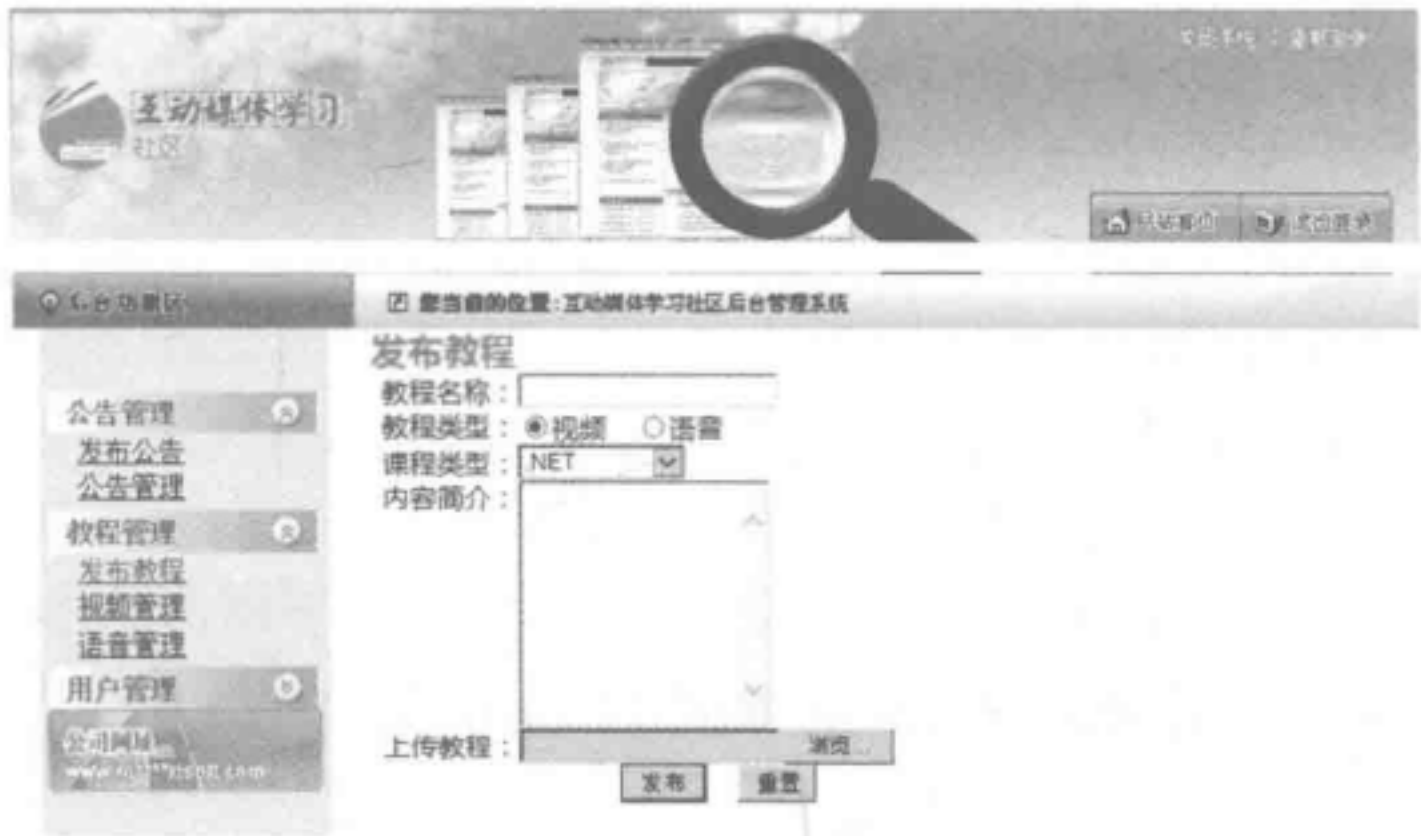


图 2-7 发布教程页面

后台视频管理页面如图 2-8 所示，主要实现查看视频详细信息、视频留言管理、删除视频的操作。

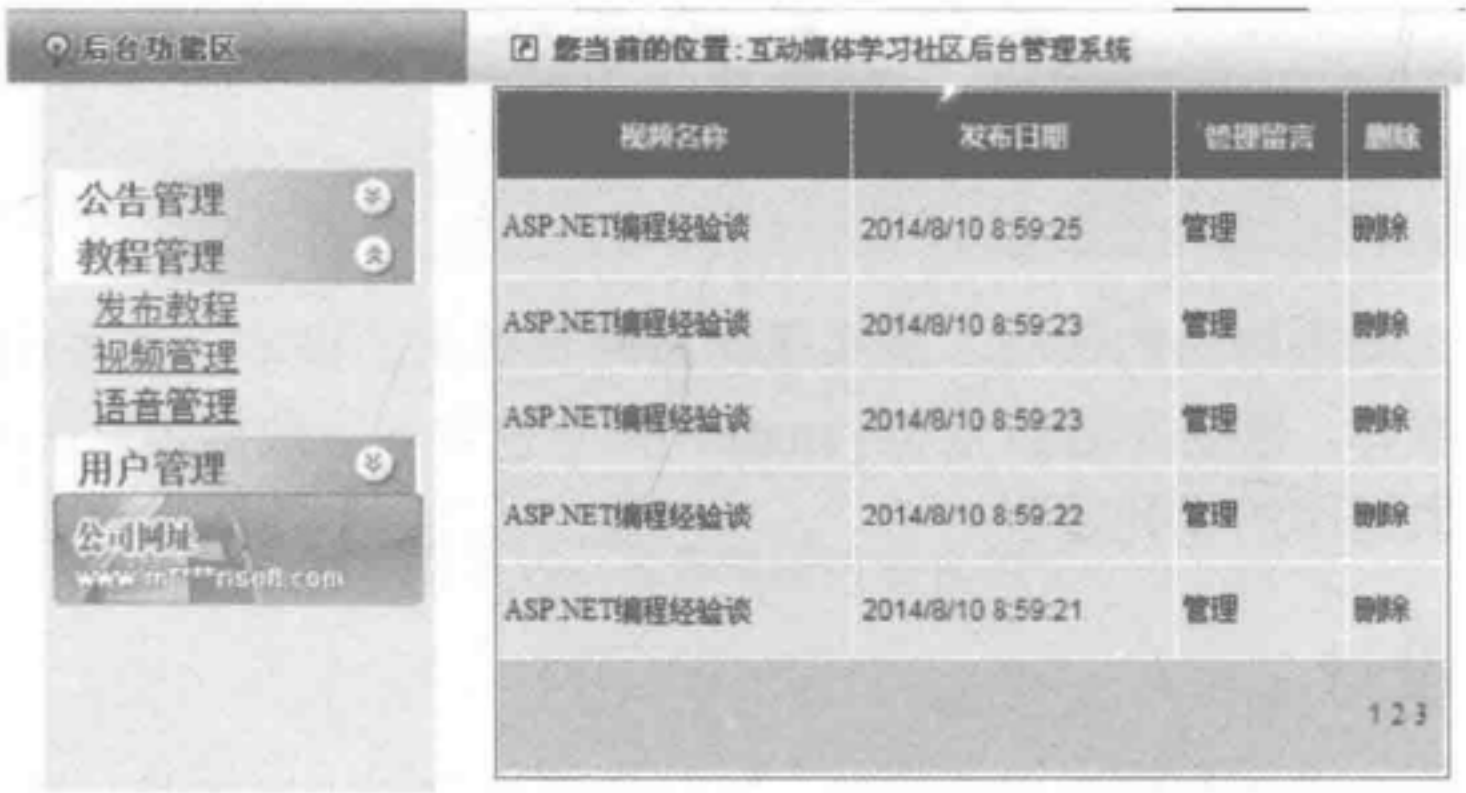



图 2-8 后台视频管理页面

作为在线 Web 系统，网页界面的美感十分重要。好的网页效果能给客户带来视觉冲击，留下好的印象。在具体美化时，我们通常会用一些漂亮的素材图片作为背景，并用 CSS 来配合控件和信息的显示。至于具体怎样搭配，就要根据客户和设计师的喜好了。

## 2.7 构建开发环境

视频讲解  光盘：视频\第 2 章\构建开发环境.avi

### 1. 网站开发环境

- (1) 网站开发环境：Microsoft Visual Studio 2013 集成开发环境。
- (2) 网站开发语言：ASP.NET + C#。
- (3) 网站后台数据库：Microsoft Access。

(4) 开发环境及运行平台: Windows XP(SP2) / Windows 2000(SP4) / Windows Server 2003(SP1) / Windows 7 / Windows 8。

💡 注意: SP(Service Pack)是指 Windows 操作系统的补丁。

## 2. 服务器端

- (1) 操作系统: Windows 7。
- (2) Web 服务器: Internet 信息服务(IIS)管理器。
- (3) 数据库服务器: Microsoft Access。
- (4) 浏览器: IE 6.0 及以上。
- (5) 网站服务器运行环境: Microsoft .NET Framework SDK 4.5.1。

## 3. 客户端

- (1) 浏览器: Internet Explorer 6.0。
- (2) 分辨率: 最佳效果为 1024×768 像素。

# 2.8 数据库设计

视频讲解 光盘: 视频\第 2 章\数据库设计.avi

数据库技术是动态网站的核心, 是实现动态网页效果的根本。本系统将采用 Access 数据库作为后台数据库, 数据库名称为 db\_study, 其中包含 6 个数据表。在本节的内容中, 将详细讲解本系统数据库的设计过程。

## 2.8.1 概念设计

数据库设计是在系统分析之后进行的, 这时的功能需求已经明确, 只需在 DBA 的参与下对数据库方案进行详细设计即可。在这一个阶段的设计工作过程中, 为了提高开发效率, 可以考虑使用第三方设计工具, 最佳推荐工具首推 PowerDesigner(以下简称 PD)。

PD 不仅能满足我们的设计需求, 还可以通过逆向工程从数据库对象生成设计模型, 并且, 它还具有相当的灵活性。PD 中的设计情况如图 2-9 所示。

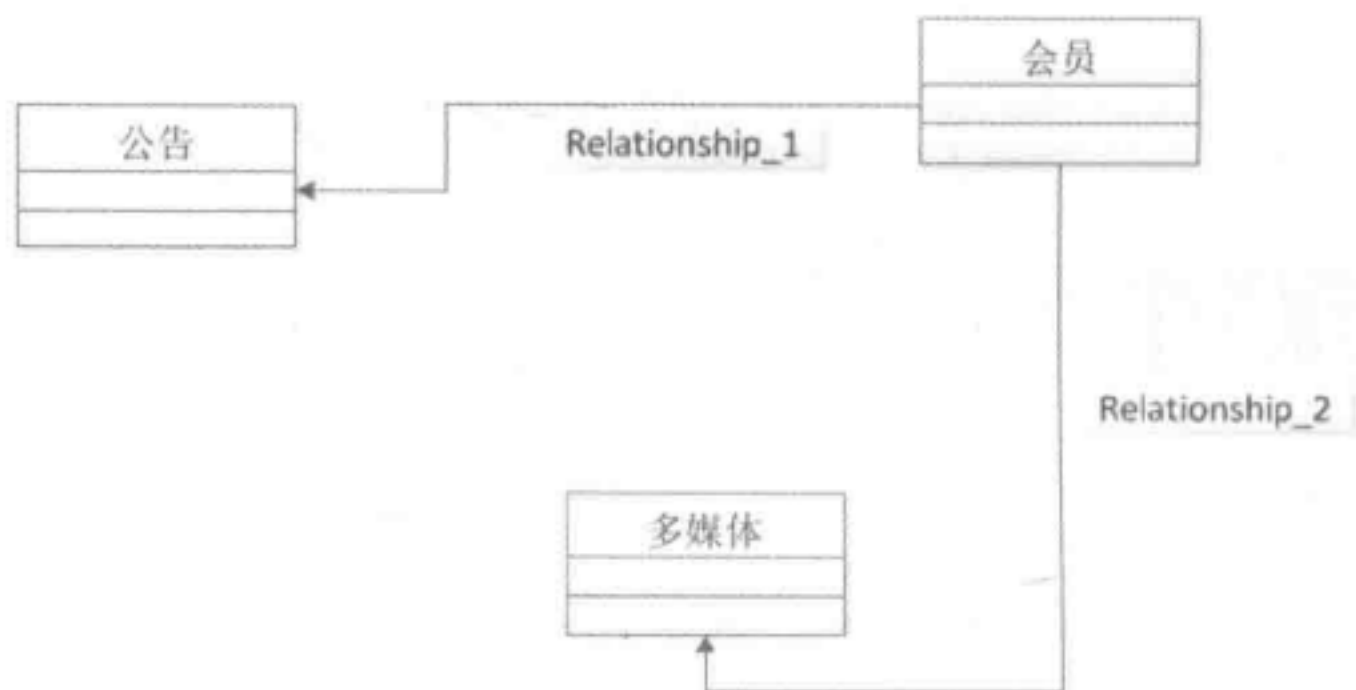


图 2-9 PD 中的设计

在概念设计阶段中，从用户的角度看待数据及处理要求和约束，产生一个反映用户观点的概念模式，然后把概念模式转换成逻辑模式。利用 ER 方法进行数据库的概念设计，可分成三步进行：首先设计局部 ER 模式，然后把各局部 ER 模式综合成一个全局模式，最后对全局 ER 模式进行优化，得到最终的模式，即概念模式。

## 2.8.2 实体 E-R 图

通过对网站进行需求分析和系统功能结构的确定，规划出系统中使用的数据库实体对象，分别为公告信息实体、留言信息实体、语言类型实体、留言信息实体、会员信息实体和教程信息实体(其中包括视频教程实体和语音教程信息实体)。

通过对网站进行的需求分析和系统功能结构的确定，规划出系统中使用的数据库实体对象，分别为公告信息实体、语言类型实体、留言信息实体、会员信息实体和视频教程信息实体(由于视频教程信息实体和语音教程信息实体类似，这里只给出视频教程信息实体)。本系统中，各个实体 E-R 图的具体说明如图 2-10 ~ 2-14 所示。

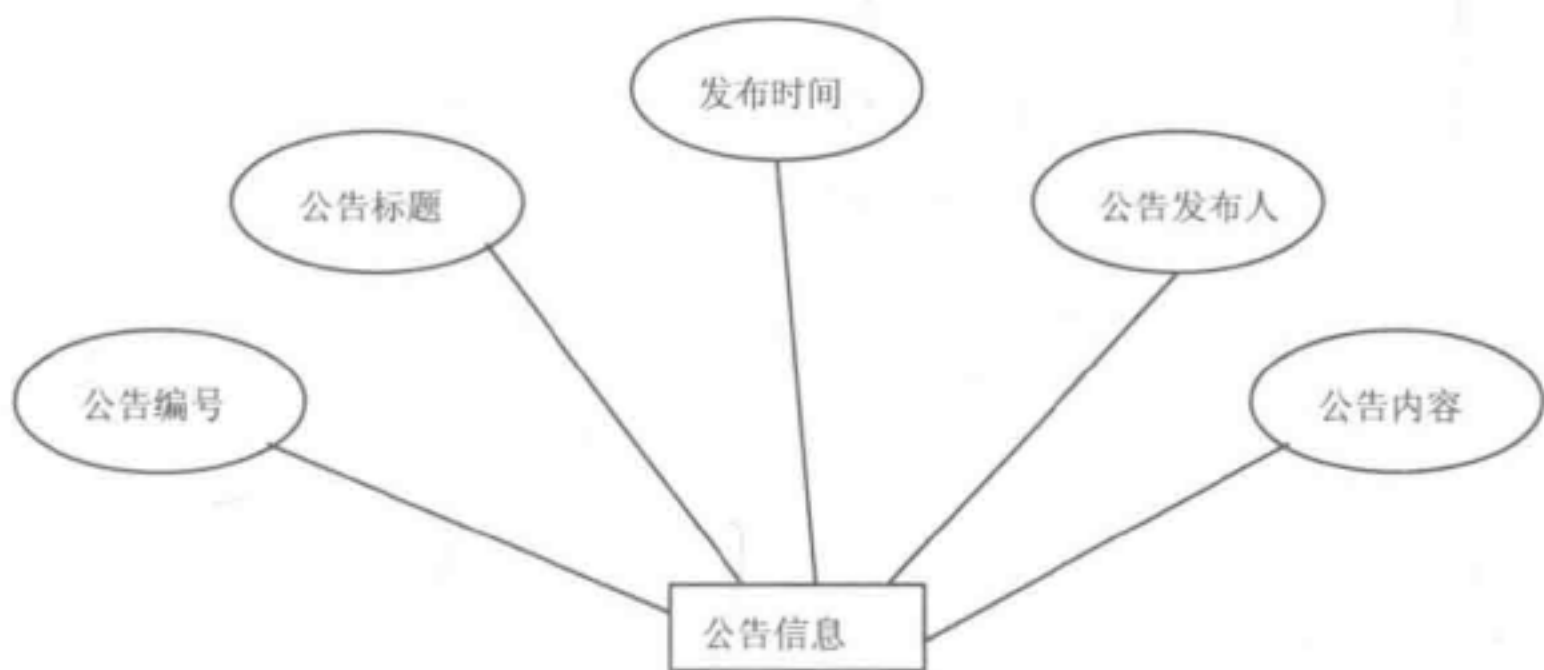


图 2-10 公告信息的 E-R 图



图 2-11 会员信息的 E-R 图





图 2-12 视频教程信息的 E-R 图

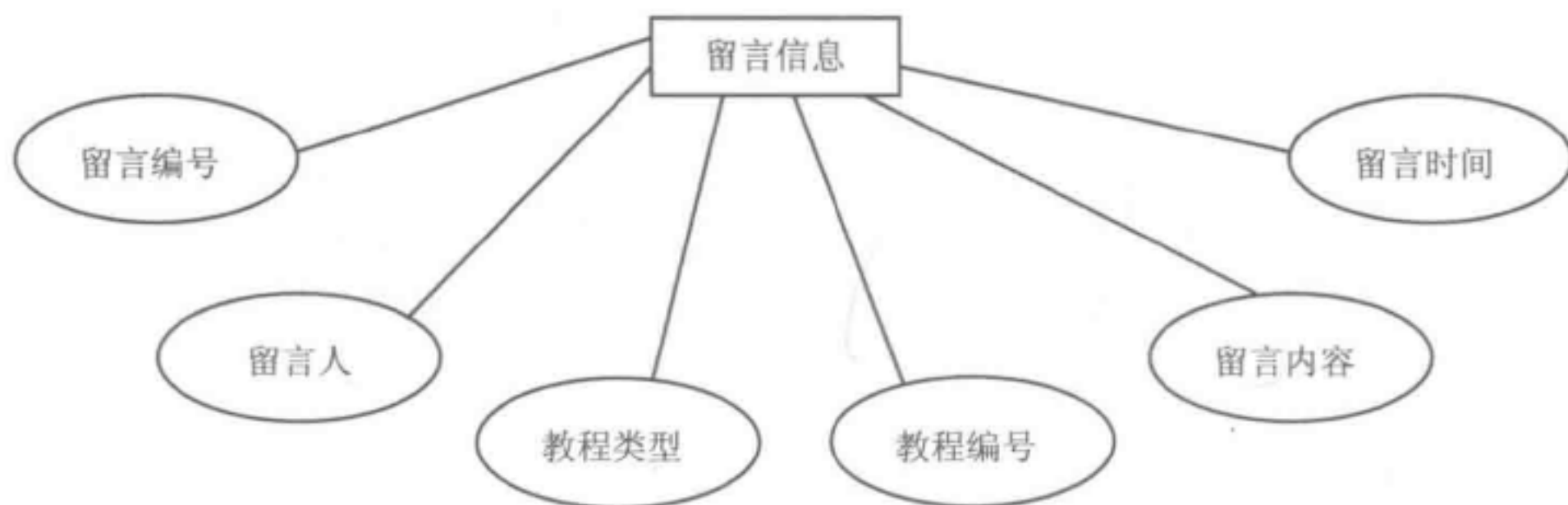


图 2-13 留言信息的 E-R 图

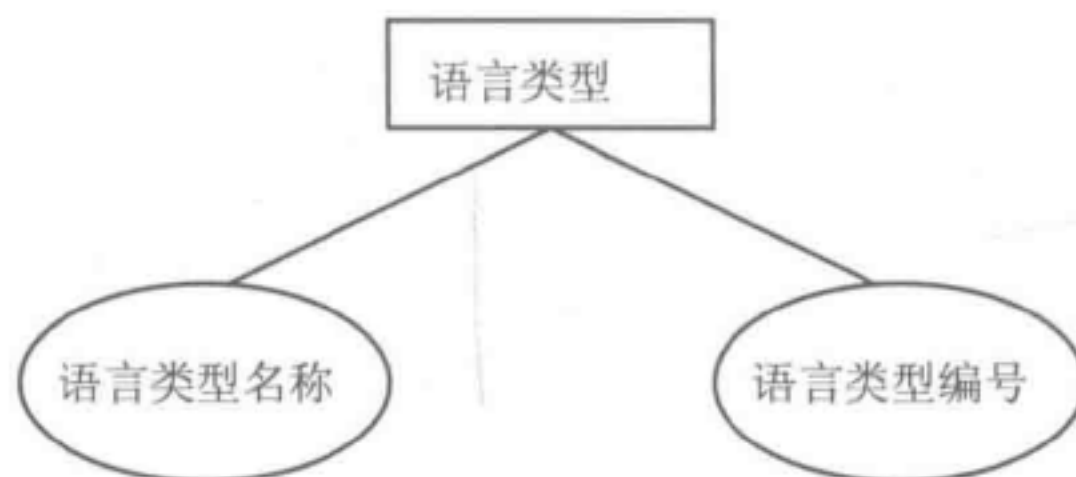


图 2-14 语言类型的 E-R 图

### 2.8.3 逻辑设计

根据系统功能设计的要求，以及功能模块的划分，对于信息系统用户信息数据库，可以列出以下数据表。

(1) 公告信息表，如表 2-1 所示。主要包括公告信息的相关消息，例如公告标题、公告发布时间、内容等。以自动增长 ID 为主键。包括一个外键 Name。

表 2-1 tb\_Bulletin(公告信息表)

| 字段名称    | 数据类型  | 字段大小 | 允许为空 |
|---------|-------|------|------|
| ID      | 自动编号  | 长整型  | 自增编号 |
| Title   | 文本    | 50   | 否    |
| Content | 备注    | 16   | 否    |
| Date    | 日期/时间 | 50   | 否    |
| Name    | 文本    | 50   | 否    |

(2) 会员信息表，如表 2-2 所示。主要存储会员注册的信息，以自动增长 ID 为主键。包括一个外键 Name。

表 2-2 tb\_login(会员信息表)

| 字段名称         | 字段类型  | 大 小 | 允许为空 | 说 明    |
|--------------|-------|-----|------|--------|
| ID           | 自动编号  | 长整型 | 自增编号 | 会员标号   |
| Name         | 文本    | 50  | 否    | 会员登录名  |
| Pass         | 文本    | 50  | 否    | 会员密码   |
| Email        | 文本    | 50  | 否    | 电子邮箱地址 |
| IDcard       | 文本    | 50  | 否    | 身份证号码  |
| Zname        | 文本    | 50  | 否    | 真实姓名   |
| Sex          | 文本    | 4   | 否    | 性别     |
| PassQuestion | 文本    | 50  | 否    | 密码提示问题 |
| PassSolution | 文本    | 50  | 否    | 密码提示答案 |
| LoginDate    | 日期/时间 | 50  | 否    | 注册时间   |
| Lock         | 数字    | 长整型 | 是    | 锁定状态   |

(3) 语音教程信息表，如表 2-3 所示。主要存储会员发布的语音教程信息，以自动增长 SoundID 为主键。包括一个外键 Name。

表 2-3 tb\_Sound(语音教程信息表)

| 字段名称      | 字段类型 | 大 小  | 允许为空 | 说 明    |
|-----------|------|------|------|--------|
| SoundID   | 数字   | 自动编号 | 自增编号 | 语音教程编号 |
| SoundType | 数字   | 4    | 否    | 教程语言类型 |

续表

| 字段名称         | 字段类型  | 大 小 | 允许为空 | 说 明      |
|--------------|-------|-----|------|----------|
| SoundName    | 文本    | 50  | 否    | 语音教程名称   |
| SoundUrl     | 文本    | 500 | 否    | 语音教程存储路径 |
| ClickSum     | 数字    | 4   | 是    | 语音教程点击率  |
| SoundContent | 文本    | 20  | 是    | 语音教程内容简介 |
| FBDate       | 日期/时间 | 10  | 否    | 语音教程发布日  |
| Name         | 文本    | 50  | 是    | 语音教程发布人  |

(4) 视频教程信息表，如表 2-4 所示。主要存储会员发布的视频教程信息，以自动增长 VideoID 为主键。包括一个外键 Name。

表 2-4 tb\_Video(视频教程信息表)

| 字段名称         | 字段类型  | 大 小  | 允许为空 | 说 明      |
|--------------|-------|------|------|----------|
| VideoID      | 数字    | 自动编号 | 自增编号 | 视频教程编号   |
| VideoType    | 数字    | 4    | 否    | 视频语言类型   |
| VideoName    | 文本    | 50   | 否    | 视频教程名称   |
| VideoUrl     | 文本    | 500  | 否    | 视频教程存储路径 |
| ClickSum     | 数字    | 4    | 是    | 视频教程点击率  |
| VideoContent | 文本    | 20   | 是    | 视频教程内容简介 |
| FBDate       | 日期/时间 | 10   | 否    | 视频教程发布日  |
| Name         | 文本    | 50   | 是    | 视频教程发布人  |

(5) 留言信息表，如表 2-5 所示。主要存储用户留言的详细信息。

表 2-5 tb\_Speak(留言信息表)

| 字段名称         | 字段类型  | 大 小 | 允许为空 | 说 明   |
|--------------|-------|-----|------|-------|
| SpeakID      | 自动编号  | 长整型 | 自增编号 | 留言人编号 |
| Speaksman    | 文本    | 50  | 否    | 留言人   |
| TutorialType | 文本    | 50  | 否    | 教程类型  |
| TutorialID   | 数字    | 4   | 否    | 教程编号  |
| SpeakContent | 备注    | 50  | 是    | 留言内容  |
| SpeakDate    | 日期/时间 | 8   | 否    | 留言日期  |



(6) 语言类型表，如表 2-6 所示。主要存储用户选择的编程语言。

表 2-6 tb\_Type(语言类型表)

| 字段名称     | 字段类型 | 大 小 | 允许为空 | 说 明    |
|----------|------|-----|------|--------|
| TypeID   | 自动编号 | 长整型 | 自增编号 | 语言类型   |
| TypeName | 文本   | 50  | 是    | 语言类型名称 |

(7) 讨论区留言信息表，如表 2-7 所示。主要存储讨论区的留言信息。

表 2-7 Messages 表(讨论区留言信息表)

| 字段名称         | 字段类型 | 大 小 | 允许为空 |
|--------------|------|-----|------|
| ID           | 自动编号 | 长整型 | 自增编号 |
| TopicID      | 数字   | 长整型 | 否    |
| MsgTitle     | 文本   | 50  | 是    |
| SpeakContent | 文本   | 50  | 否    |
| Speaksman    | 文本   | 50  | 否    |
| SpeakDate    | 文本   | 50  | 否    |

2.9 设计文件夹组织结构和功能模块

视频讲解  光盘：视频\第 2 章\设计文件夹组织结构和功能模块

为了便于读者对本网站的学习和理解，在此将网站文件的组织结构展示出来，这样，读者对本系统从整体结构上就可以有一个大致的了解了，有利于对后面各个文件具体实现过程的学习。

2.9.1 文件组织结构的设计

前台文件的组织结构如图 2-15 所示。  
后台文件的组织结构如图 2-16 所示。



图 2-15 前台文件的组织结构



图 2-16 后台管理文件的组织结构

## 2.9.2 用户功能模块设计

### (1) 识别参与者

参与者有游客、会员、管理员。

在用户功能模块中，对于普通用户，提供相关的用户登录、注册、注销、修改密码的功能；而对于后台的管理员，不仅提供相关的登录及验证机制，而且，管理员可以对所有普通用户的信息进行锁定、删除的操作。

管理员用例图如图 2-17 所示。

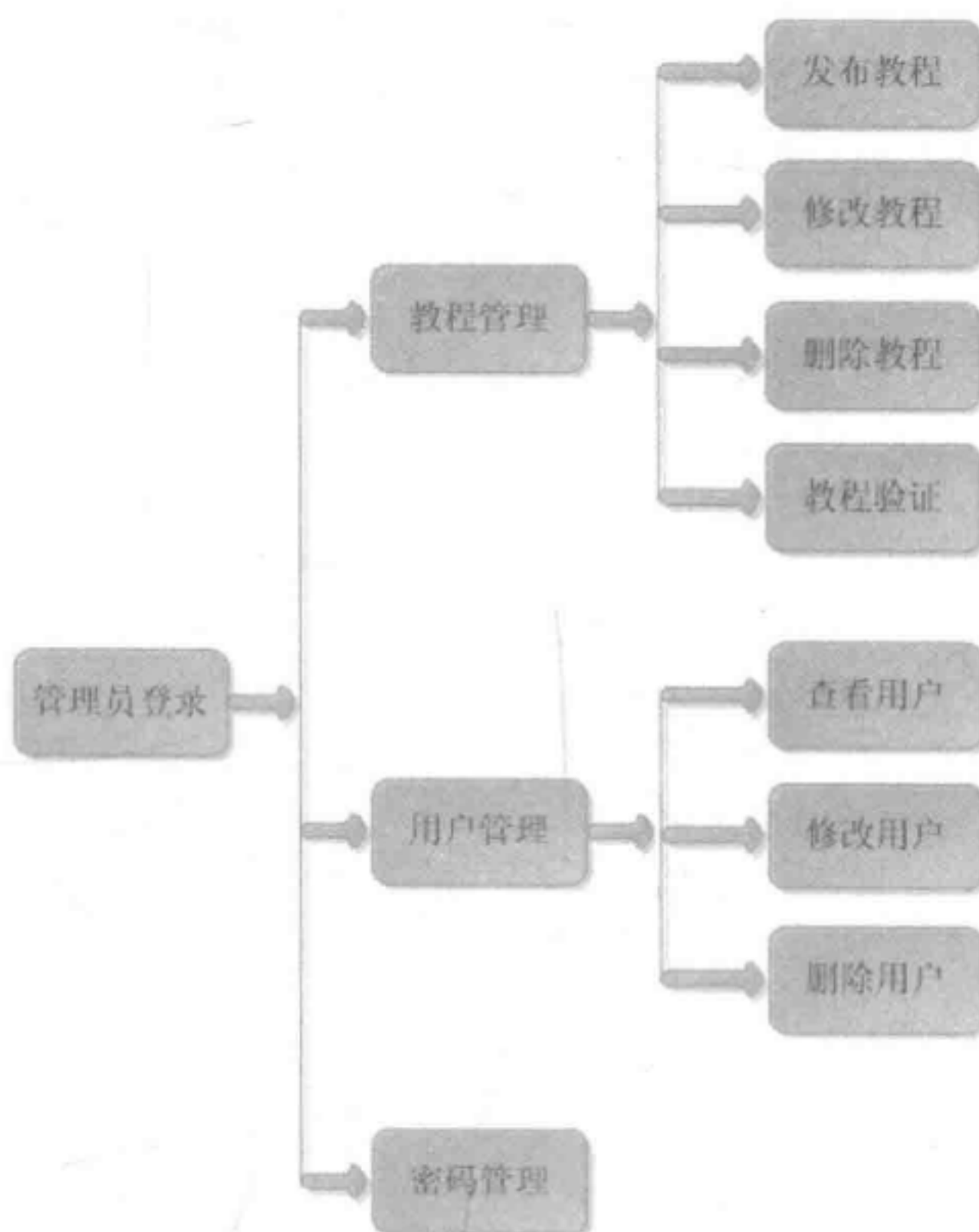


图 2-17 管理员用例图

会员用例图如图 2-18 所示。

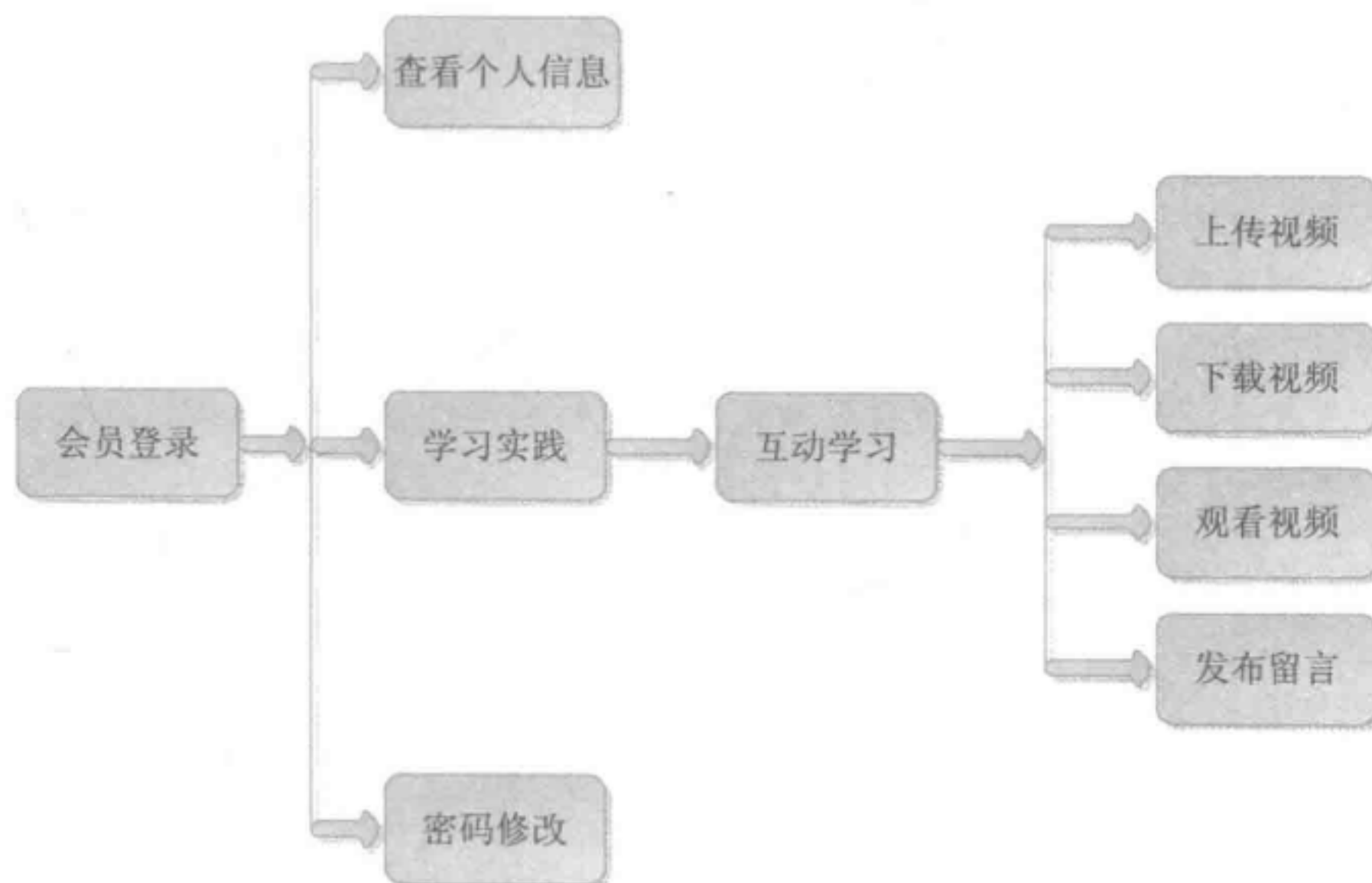


图 2-18 会员用例图



## (2) 教程模块的设计

在教程模块中，普通用户与管理员都可以浏览教程信息，所有用户可以通过教程简介、教程排行来查看信息，也可以通过关键字来查询有关教程的信息。所有用户在观看教程后可以留言。对于注册用户除了浏览教程，还可以发布自己的教程。管理员查看并管理教程，对所有教程进行验证。游客与教程模块，注册用户与教程模块，管理员教程模块的功能分别如图 2-19、2-20 和 2-21 所示。

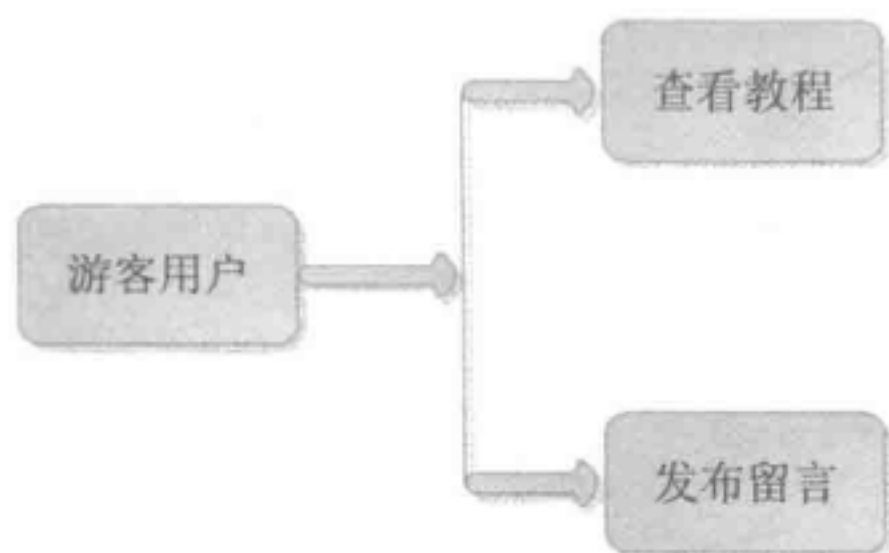


图 2-19 游客与教程模块的功能

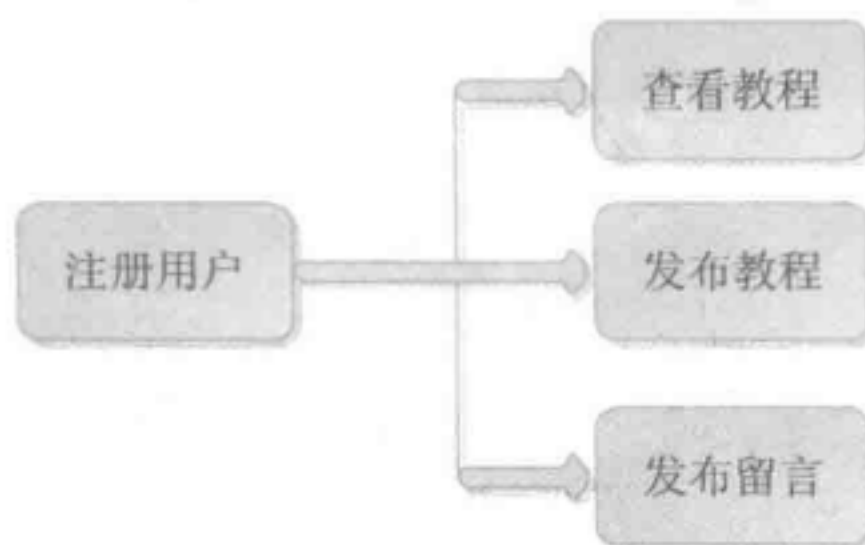


图 2-20 注册用户与教程模块的功能

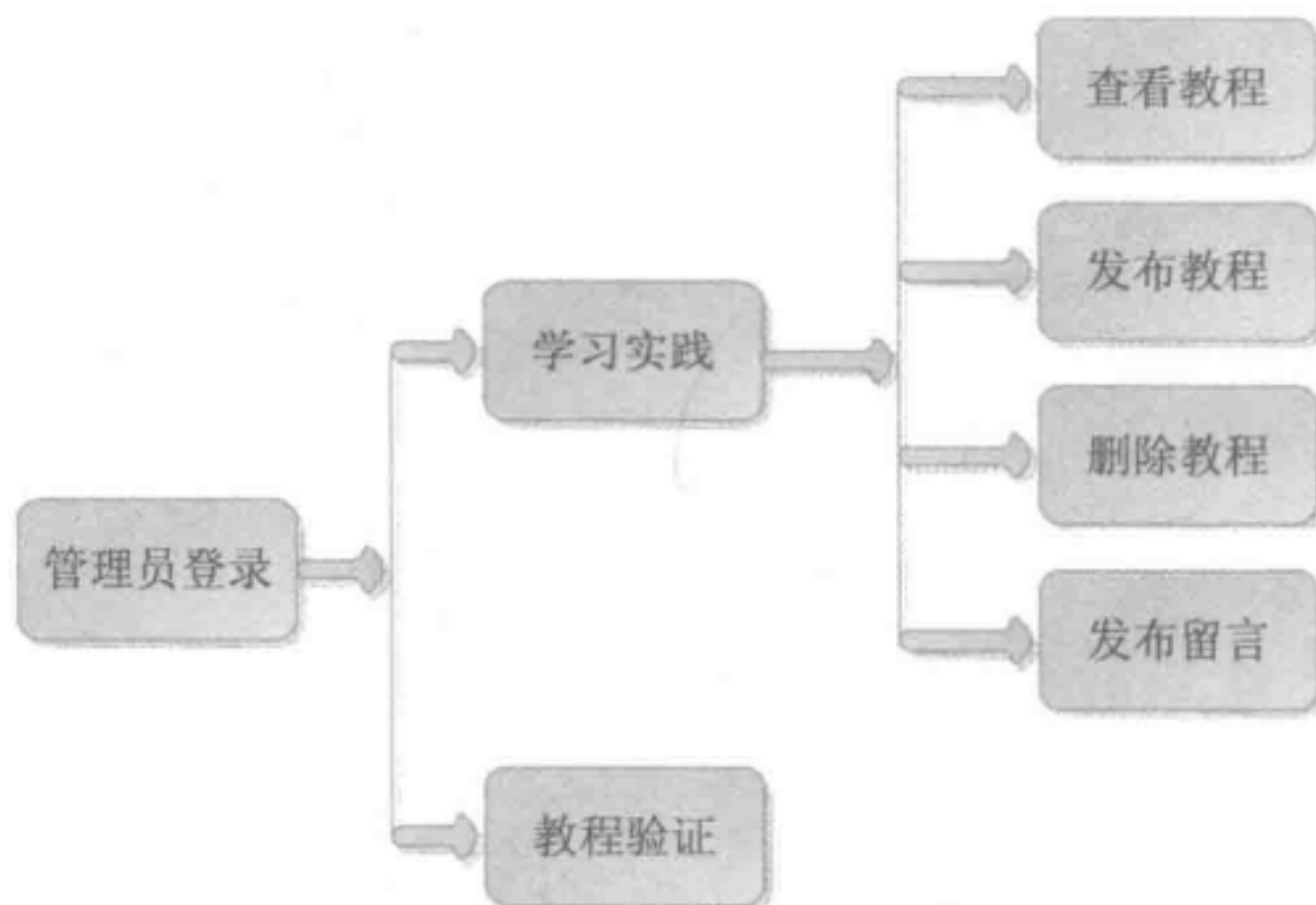


图 2-21 管理员与教程模块的功能

## (3) 论坛留言功能模块的设计

在论坛留言功能模块中，普通用户或者管理员都可以在网站上进行留言，而管理员则能够在后台读取并浏览相关留言或者对留言信息进行相应的操作处理。

图 2-22 和 2-23 分别给出了普通用户留言功能模块及管理员管理留言功能模块。

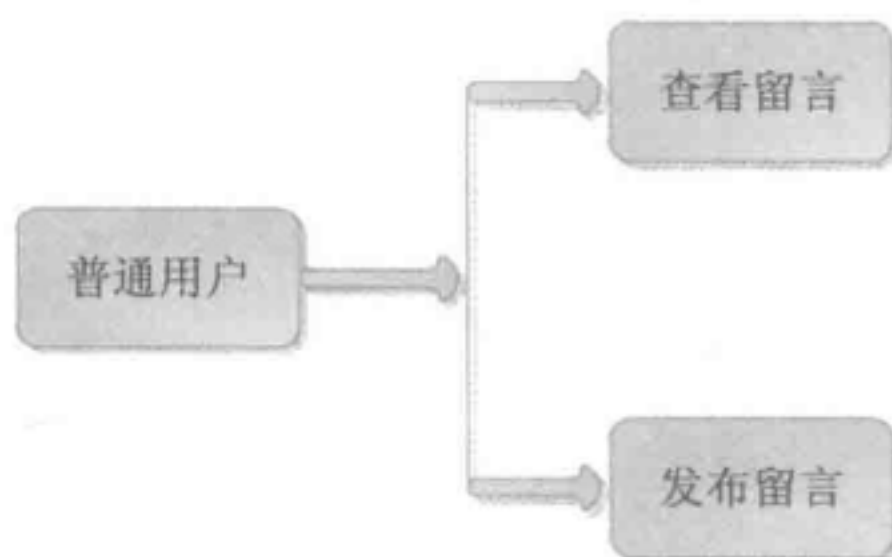


图 2-22 普通用户留言功能模块

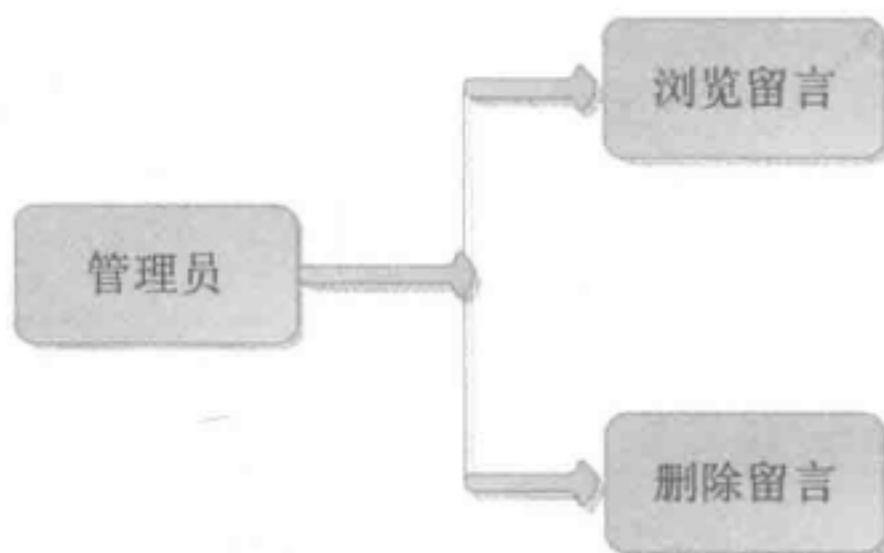


图 2-23 管理员管理留言功能模块

## 2.10 公共类的设计

视频讲解 光盘: 视频\第2章\公共类设计.avi

在软件项目中, 公共类的设计遵循了面向对象的模块化设计思想, 主要目的是将在项目中被多次用到的功能编写成独立的类, 在使用时直接调用这些类即可。这样做的好处是便于维护, 提高了开发效率, 减少了代码编写量。

有关模块化设计的好处, 读者可以从古龙先生的武侠小说片段中得到启迪:

一座高山, 一处低岩, 一道新泉, 一株古松, 一炉红火, 一壶绿茶, 一位老人, 一个少年。

“天下最可怕的武器是什么?” 少年问老人: “是不是例不虚发的小李飞刀?”

“以前也许是, 现在却不是了。”

“为什么?”

“因为自从小李探花仙去后, 这种武器已成绝响。”老人黯然叹息: “从今以后, 世上再也不会有小李探花这种人; 也不会再有小李飞刀这种武器了。”

少年仰望高山, 山巅白云悠悠。

“现在世上最可怕的武器是什么?” 少年又问老人: “是不是蓝大先生的蓝山古剑?”

.....

“不是。”老人道: “你说的这些武器虽然都很可怕, 却不是最可怕的一种。”

“最可怕的一种是什么?”

“是一口箱子。”

“一口箱子?” 少年惊奇极了: “当今天下最可怕的武器是一口箱子?”

“是的。”

这是出自古龙的武侠名著《英雄无泪》的一段对白, 没错, 最厉害的武器是一口箱子。

等看整本小说之后才明白, 这不是一口简单的箱子, 箱子里有很多个零部件, 能够根据不同的对手而迅速组成一个战胜对手的武器。

我们将箱子中的武器引申到 C# 程序, 会发现, 这个箱子与程序中的类是十分相似的。我们要实现某个功能时, 可以编写一个类来实现它。如果有多个问题, 则编写多个类就可以实现, 类就是我们编程中的那个神秘的箱子。

例如, 数据库操作类可用来完成数据库的连接操作以及数据库的查询、添加、删除和修改操作, 将这几种操作编写到一个公共类里, 就可以减少重复代码的编写, 有利于代码的维护。在本节的内容中, 将详细讲解本系统公共类的具体设计过程。

### 2.10.1 数据库操作类的设计

创建数据库操作类的方法为: 从 Microsoft Visual Studio 2013 的菜单栏中选择“网站”→“添加新项”命令, 在弹出的“添加新项”对话框中选择“类”, 将文件命名为 dataOperate.cs,

如图 2-24 所示。

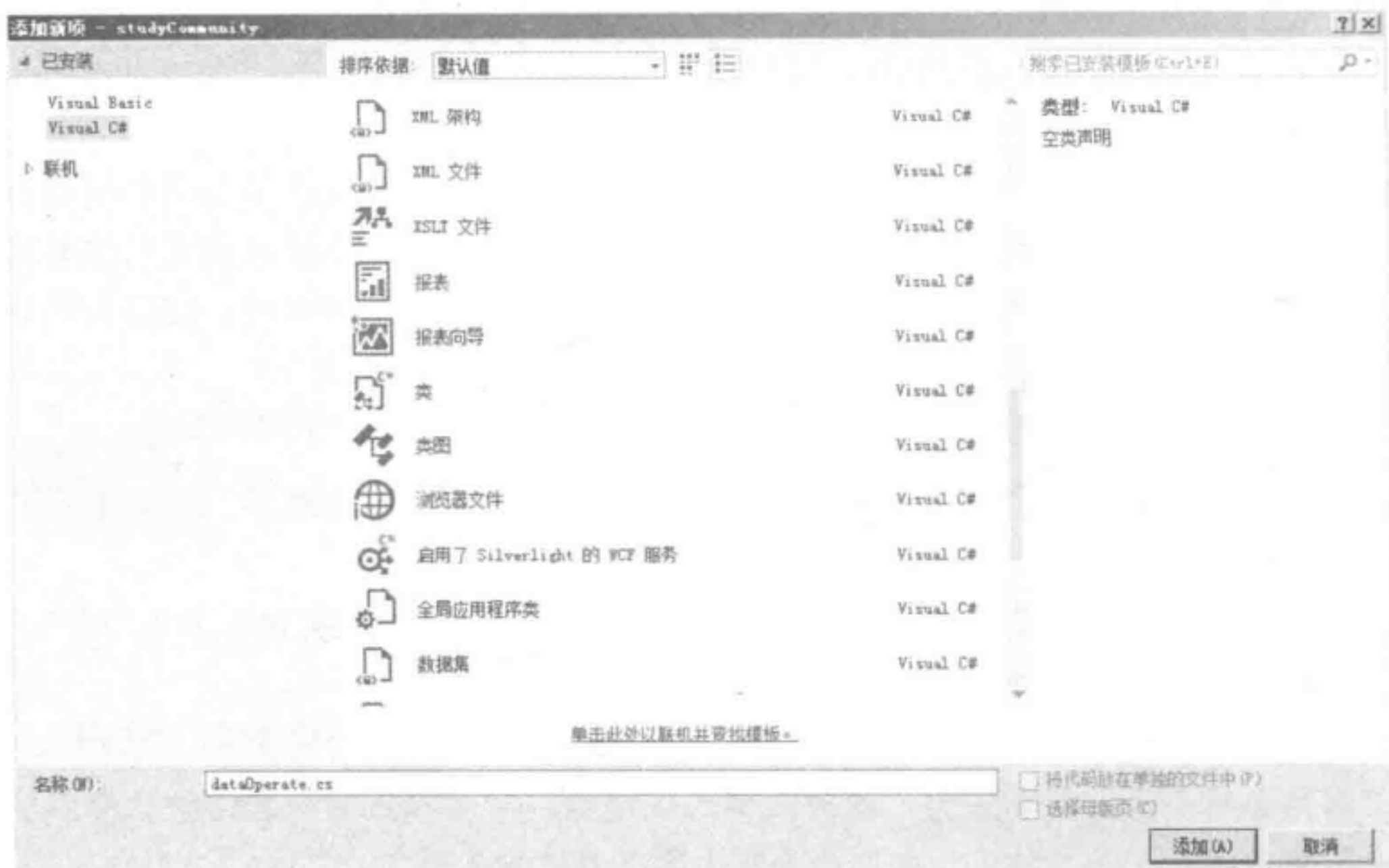


图 2-24 添加数据库操作类

单击“添加”按钮，将弹出一个提示对话框，如图 2-25 所示，此对话框询问是否将刚才创建的类存放在 App\_Code 文件夹中，单击“是”按钮，完成数据库操作类的创建。

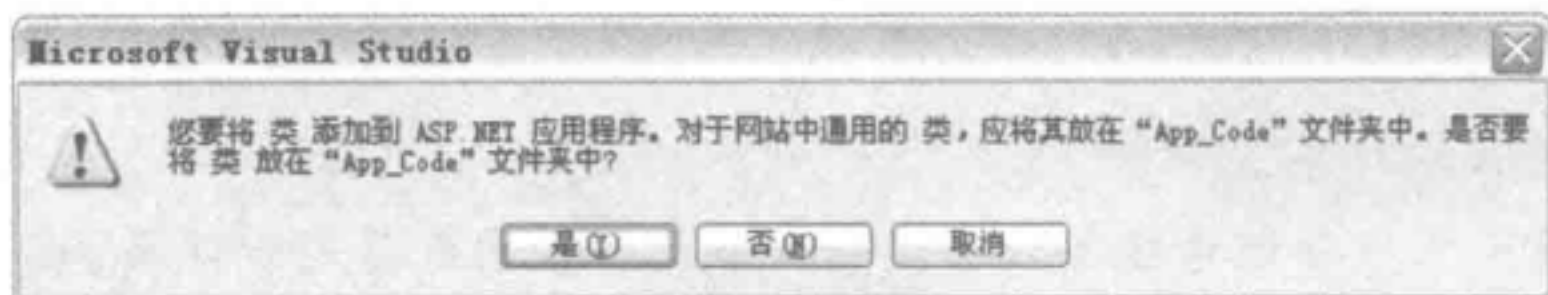


图 2-25 提示对话框

在解决方案资源管理器里的 App\_Code 文件夹中，可以看到新创建的数据库操作类。双击数据库操作类，进行此类的编写。在此类里，可以看到系统自动添加的命名空间、公共类和构造函数。

由于此类需要对数据库进行操作，所以需要引用命名空间 System.Data.OleDb。dataOperate.cs 文件的代码如下：

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;
/// <summary>
/// dataOperate 的摘要说明
```



```

/// </summary>
public class dataOperate
{
    public static DataSet ds;
    public dataOperate()
    {
        //
        // TODO: 在此处添加构造函数逻辑
        //
    }
}

```

在 dataOperate 类中，一共需要定义 6 个方法，下面分别对这几个方法进行讲解。

### (1) createCon 方法

createCon()方法用来连接数据库，此方法返回的类型为 OleDbConnection，主要用来构造数据库的连接，代码如下：

```

public OleDbConnection createCon()
{
    //生成 OleDbConnection 的一个对象，用于连接数据库
    OleDbConnection odbc = new OleDbConnection(
        "Provider=Microsoft.Jet.OLEDB.4.0;Data source=|DataDirectory|db_study.mdb;");
    return odbc;
}

```

### (2) adlData 方法

adlData(string sql)方法用来添加或删除数据。此方法返回一个布尔值，用来表示添加或删除数据是否成功，执行成功返回 true，否则返回 false。调用此方法时，应传入一个 string 类型的参数，此参数表示所要执行的 SQL 语句。代码如下：

```

public bool adlData(string sql)
{
    OleDbConnection Odbc = createCon(); //调用 createCon 方法连接数据库
    Odbc.Open(); //打开数据库连接
    OleDbCommand com = new OleDbCommand(sql, Odbc); //对 Access 数据库执行一个 SQL 语句
    int i = Convert.ToInt32(com.ExecuteNonQuery()); //返回所影响的行，并转换成 int 类型
    Odbc.Close();
    if (i > 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

### (3) isData 方法

isData(string sql)方法用来查找数据是否存在。此方法返回一个整型值，用来表示是否查找到数据，查找到数据则返回一个大于 0 的值，否则返回 0。调用此方法时，应传入一个 string 类型的参数，此参数表示所要执行的 SQL 语句。代码如下：

```

public int isData(string sql)
{
    OleDbConnection Odbc = createCon(); //调用 createCon 方法连接数据库
    Odbc.Open(); //打开数据库连接
}

```

```
OleDbCommand com = new OleDbCommand(sql, Odbc); //对 Access 数据库执行一个 SQL 语句
int i = Convert.ToInt32(com.ExecuteScalar()); //返回首行首列
Odbc.Close();
return i;
}
```

#### (4) updateData 方法

updateData(string sql)方法用来更新数据。此方法没有返回值。在调用此方法时,应传入一个 string 类型的参数,此参数表示所要执行的 SQL 语句。代码如下:

```
public void updateData(string sql)
{
    OleDbConnection Odbc = createCon(); //调用 createCon 方法连接数据库
    Odbc.Open(); //打开数据库连接
    OleDbCommand com = new OleDbCommand(sql, Odbc); //对 Access 数据库执行一个 SQL 语句
    com.ExecuteScalar(); //返回首行首列
    Odbc.Close();
}
```

#### (5) row 方法

row(string sql)方法用来查找并返回一行数据。此方法返回一个 OleDbDataReader 对象。在调用此方法时,应传入一个 string 类型的参数,此参数表示所要执行的 SQL 语句。

代码如下:

```
public OleDbDataReader row(string sql)
{
    OleDbConnection Odbc = createCon(); //创建 OleDbConnection 对象
    Odbc.Open(); //打开数据库连接
    OleDbCommand com = new OleDbCommand(sql, Odbc);
    return com.ExecuteReader(); //返回 OleDbDataReader 对象
}
```

#### (6) rows 方法

方法 rows(string sql, string table)用来查找并返回多行数据。此方法返回一个 DataTable 对象。在调用此方法时,应传入两个 string 类型的参数,第一个参数表示要执行的 SQL 语句,第二个参数表示表名。代码如下:

```
public DataTable rows(string sql, string table)
{
    DataSet ds;
    OleDbConnection Odbc = createCon();
    Odbc.Open();
    OleDbDataAdapter oda = new OleDbDataAdapter(sql, Odbc);
    ds = new DataSet(); //创建数据集
    oda.Fill(ds, table); //填充数据集
    Odbc.Close();
    return ds.Tables[table]; //返回数据表
}
```

## 2.10.2 业务功能类设计

网站业务功能类用来存放开发中常用的方法,可以减少重复代码的编写,有利于代码的维护。在此类中有两个自定义方法,即 Encrypting 方法和 Decrypting 方法,这两个方法主要用来完成重要信息的加密和解密。

### (1) Encrypting 方法

Encrypting(string Source)方法用来对比较重要的信息进行加密操作。此方法返回一个 string 类型的值, 该值表示已经加密的信息。在调用此方法时, 应传入一个 string 类型的参数, 此参数表示需要加密的信息。

代码如下:

```
public static string Encrypting(string strSource)
{
    //把字符串放到byte 数组中
    byte[] bytIn = System.Text.Encoding.Default.GetBytes(strSource);
    //建立加密对象的密钥和偏移量
    byte[] iv = { 102, 16, 93, 156, 78, 4, 218, 32 }; //定义偏移量
    byte[] key = { 55, 103, 246, 79, 36, 99, 167, 3 }; //定义密钥
    //实例 DES 加密类
    DESCryptoServiceProvider mobjCryptoService = new DESCryptoServiceProvider();
    mobjCryptoService.Key = iv;
    mobjCryptoService.IV = key;
    ICryptoTransform encrypto = mobjCryptoService.CreateEncryptor();
    //实例 MemoryStream 流加密文件
    System.IO.MemoryStream ms = new System.IO.MemoryStream();
    CryptoStream cs = new CryptoStream(ms, encrypto, CryptoStreamMode.Write);
    cs.Write(bytIn, 0, bytIn.Length);
    cs.FlushFinalBlock();
    return System.Convert.ToBase64String(ms.ToArray());
}
```

### (2) Decrypting 方法

Decrypting(string Source)方法将已加密的信息进行解密。此方法返回一个 string 类型的值, 该值表示解密后的信息。在调用此方法时, 应传入一个 string 类型的参数, 此参数表示需要解密的信息。

代码如下:

```
public static string Decrypting(string Source)
{
    try
    {
        //将解密字符串转换成字节数组
        byte[] bytIn = System.Convert.FromBase64String(Source);
        //给出解密的密钥和偏移量, 密钥和偏移量必须与加密时的密钥和偏移量相同
        byte[] iv = { 102, 16, 93, 156, 78, 4, 218, 32 }; //定义偏移量
        byte[] key = { 55, 103, 246, 79, 36, 99, 167, 3 }; //定义密钥
        DESCryptoServiceProvider mobjCryptoService = new DESCryptoServiceProvider();
        mobjCryptoService.Key = iv;
        mobjCryptoService.IV = key;
        //实例流解密
        System.IO.MemoryStream ms = new System.IO.MemoryStream(bytIn, 0, bytIn.Length);
        ICryptoTransform encrypto = mobjCryptoService.CreateDecryptor();
        CryptoStream cs = new CryptoStream(ms, encrypto, CryptoStreamMode.Read);
        StreamReader strd = new StreamReader(cs, Encoding.Default);
        return strd.ReadToEnd();
    }
    catch (Exception ex)
    {
        throw new Exception("在文件解密的时候出现错误! 错误提示: \n" + ex.Message);
    }
}
```



⚠ 注意：实现 Encrypting 方法和 Decrypting 方法时，需引用 System.Security.Cryptography 命名空间、System.IO 命名空间、System.Text 命名空间。

## 2.11 网站首页设计

视频讲解 光盘：视频\第 2 章\网站首页设计.avi

网站首页中包含如下所示的模块。

- 网站导航：包括发布教程、视频课堂、语音课堂、注册、联系我们。
- 教程搜索：可以根据教程类型和教程语言进行搜索。
- 网站公告：网站近期的动态，或是一些通告事项。
- 用户登录：只有登录，用户才有权限发布教程和下载教程。
- 最新发布教程：包括最新发布的视频教程和语音教程。
- 教程排行榜：包括点击率最高的视频教程和语音教程。

首页设计得好坏，直接影响到浏览者及用户对网站的印象，因此页面整体布局要合理，简洁美观，网站首页的运行效果如图 2-26 所示。



图 2-26 网站的首页

在主页 Web 窗体的加载事件中，调用各个功能绑定到 DataList 控件上的方法。实现代码如下：


```
protected void creatVideo()
{
    try
```

```

{
    string Sql = "SELECT top 10 * from tb_Video as a inner join tb_Type as b
                on a.VideoType=b.TypeID ORDER BY VideoID DESC";
    gvNewVideo.DataSource = mydo.rows(Sql, "tb_Video").DefaultView;
    gvNewVideo.DataBind();
}
catch (Exception error)
{ Response.Redirect(error.Message.ToString()); }
}
//最新语音
protected void creatSound()
{
    try
    {
        string Sql = "select top 10 * from tb_sound as a inner join tb_Type as b
                    on a.SoundType=b.TypeID order by SoundID DESC";
        gvNewSound.DataSource = mydo.rows(Sql, "tb_sound").DefaultView;
        gvNewSound.DataBind();
    }
    catch (Exception error)
    {
        Response.Write(error.ToString());
        Response.Write("<script language=javascript>alert('数据库失败')</script>");
    }
}
}

```

## 2.12 实现用户注册模块

视频讲解  光盘：视频\第2章\实现用户注册模块.avi

浏览者可以通过用户注册功能注册成为本网站的会员，用户注册页面如图 2-27 所示。



图 2-27 用户注册页面

用户注册并登录后，可以发布自己制作的视频教程和语音教程，也可以下载自己喜欢的教程。当用户在首页单击导航栏中的“注册”链接按钮或在登录模块中单击“新用户注

册”按钮时，将进入用户注册页面。

实现用户注册信息时应注意：用户名不能为空、密码必须填写、两次密码输入必须一致、电子邮件地址格式是否正确、身份证号格式是否正确。这些信息的验证都是通过服务器验证控件来实现的。

## 2.12.1 login.aspx 页面部分代码分析

该页面首先通过调用 add 方法，将用户添加的注册信息添加到数据库中，具体的实现代码如下所示：

```
protected bool add()
{
    dataOperate mydo = new dataOperate(); //创建数据库操作类的对象
    string name = txtName.Text;
    //调用业务功能类中 Encrypting 方法，对用户密码进行加密
    string pass = Operate.Encrypting(txtPass.Text);
    string sex; //获取性别
    if (RadioButtonMan.Checked)
    {
        sex = "男";
    }
    else
    {
        sex = "女";
    }
    string trueName = txtTrueName.Text; //获取真实姓名
    string idCard = this.txtIDCard.Text; //获取电话
    string passQuestion = this.txtPassQuestion.Text;
    //根据指定的密码和哈希算法，生成一个适合于存储在配置文件中的哈希密码
    string passSolution = FormsAuthentication.HashPasswordForStoringInConfigFile(
        this.txtPassSolution.Text, "MD5");
    string email = txtEmail.Text; //获取电子邮件
    string sql = "insert into tb_login(
        Name,Pass,ZName,Sex,Email,IDCard,PassQuestion, PassSolution) values('"
        + name + "','" + pass + "','" + trueName + "','" + sex + "','" + email + "','"
        + idCard + "','" + passQuestion + "','" + passSolution + "')";
    return mydo.adlData(sql);
}
```

## 2.12.2 用户登录设计

在登录框输入对应的用户名、密码后，确定登录，如图 2-28 所示。提交后，所需参数将传送到后台的 entry.aspx 中。



图 2-28 用户登录界面的效果

首先通过用户名对数据库中的 tb\_login 表进行检索，若检索到的记录集为空，提示用户名不正确，若记录不为空，再进行密码判断，若密码与数据表中的密码不一致，则同样会给出错误提示信息，当用户名和密码都正确后，则成功登录后台管理页面，并将用户以 session 对象保存起来。用户登录的流程如图 2-29 所示。



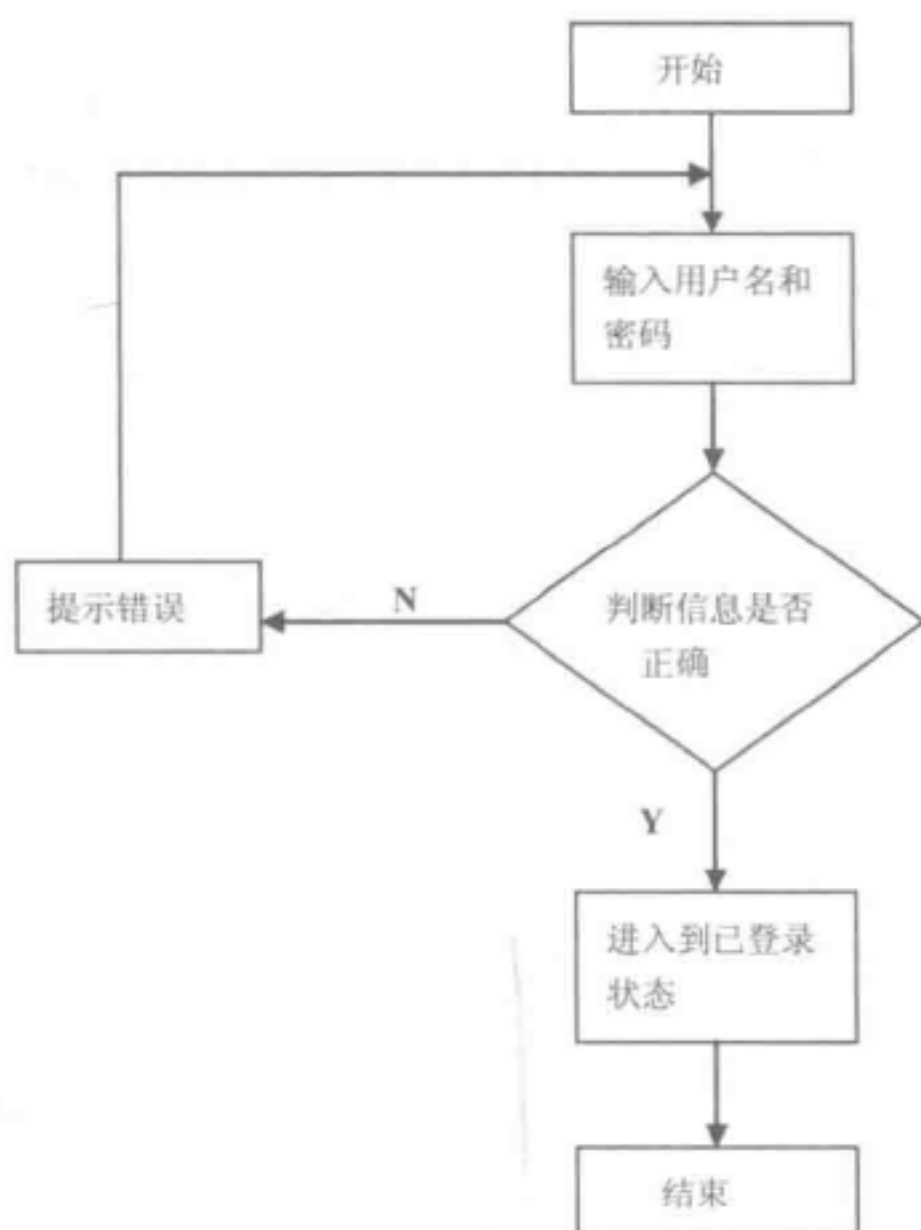


图 2-29 用户登录的流程

当用户注册为本网站的会员后，必须进行登录后，才能使用上传或下载教程和管理教程的功能。当用户登录成功后，进行留言时，会将发言人保存为用户的登录名，如果是普通用户发言人，将保存为游客。这里主要应用了 Session 对象，利用 Session 对象来保存用户登录名。当用户发布教程时，或在发布留言时，可以将 Session 对象保存的用户名添加到数据库中。Session 对象是 HttpSessionState 类的一个实例，其功能是用来存储跨网页程序的变量或者对象，Session 对象只针对单一访问者使用，也就是说，各个连接的机器都有各自的 Session 对象，不同的客户端无法互相存取。Session 对象终止于联机机器离线时，也就是当使用者关掉浏览器或超过设定的 Session 变量的有效时间时，Session 对象就会消失。Session 对象的常用属性及说明如下。

(1) Contents 属性：获取对当前会话状态对象的引用。

语法：public HttpSessionState Contents{get;}

属性值：当前的 HttpSessionState

(2) Item 属性：获取或设置会话值，该属性可重载，它有两种格式。

① 按数字索引获取或设置会话值。

语法：public Object this[int index]{get;set;}

属性值：存储在指定索引中的会话状态值。

② 按名称获取或设置会话值。

语法：public Object this[string name]{get;set;}

属性值：带指定名称的会话状态值。

(3) Timeout 属性：获取并设置在会话状态提供程序终止会话之前各请求之间所能维持的时间(以分钟为单位)。

语法：public int Timeout{get;set;}

属性值：超时期限(以分钟为单位)。

登录功能的实现需要先判断验证码是否正确，验证码正确才需要使用数据库操作类中的 row 方法来判断用户输入的用户名和密码是否正确。实现代码如下：

```
protected void imgbtnLanding_Click(object sender, ImageClickEventArgs e)
{
    string name = txtName.Text; //将登录名存储到变量中
    string pass = Operate.Encrypting(txtPass.Text); //将密码加密并存储到变量中
    string yzm = txtYzm.Text; //将验证码存储到变量中
    if (Session["CheckCode"].ToString().Equals(yzm)) //判断验证码是否正确
        //Session["CheckCode"]用来存储自动生成的验证码
    {
        try {
            //调用数据库操作类中的 isData 方法判断用户是否存在
            string sql = "select count(*) from tb_login where Name='"
                + name + "' and Pass='" + pass + "'";
            int i = mydo.isData(sql);
            if (i > 0)
            {
                sql = "select * from tb_login where Name='" + name + "'";
                OleDbDataReader odr = mydo.row(sql); //返回一条记录
                odr.Read();
                if (odr["lock"].ToString() == "0") //判断用户是否锁定
                {
                    Session["UserName"] = name; //将登录名保存到 Session 中
                    Response.Redirect("index.aspx");
                }
                else
                {
                    Page.RegisterStartupScript(
                        "false", "<script>alert('此用户已被锁定!')</script>");
                }
            }
            else
            {
                Response.Write("<script>alert('密码或用户名错误!')</script>");
            }
        }
        catch (Exception ex) {
            Response.Write(ex.Message.ToString());
        }
    }
    else
    {
        Page.RegisterStartupScript("false", "<script>alert('验证码错误!')</script>");
    }
}
```

## 2.13 发布并管理教程

视频讲解 光盘：视频\第2章\发布并管理教程.avi

网站的会员用户可以将自己制作的或从其他途径收集到的教程发布到网站上，供其他用户共享，还可以对自己发布的教程进行管理，以及查看其他用户的留言。效果如图 2-30 所示。

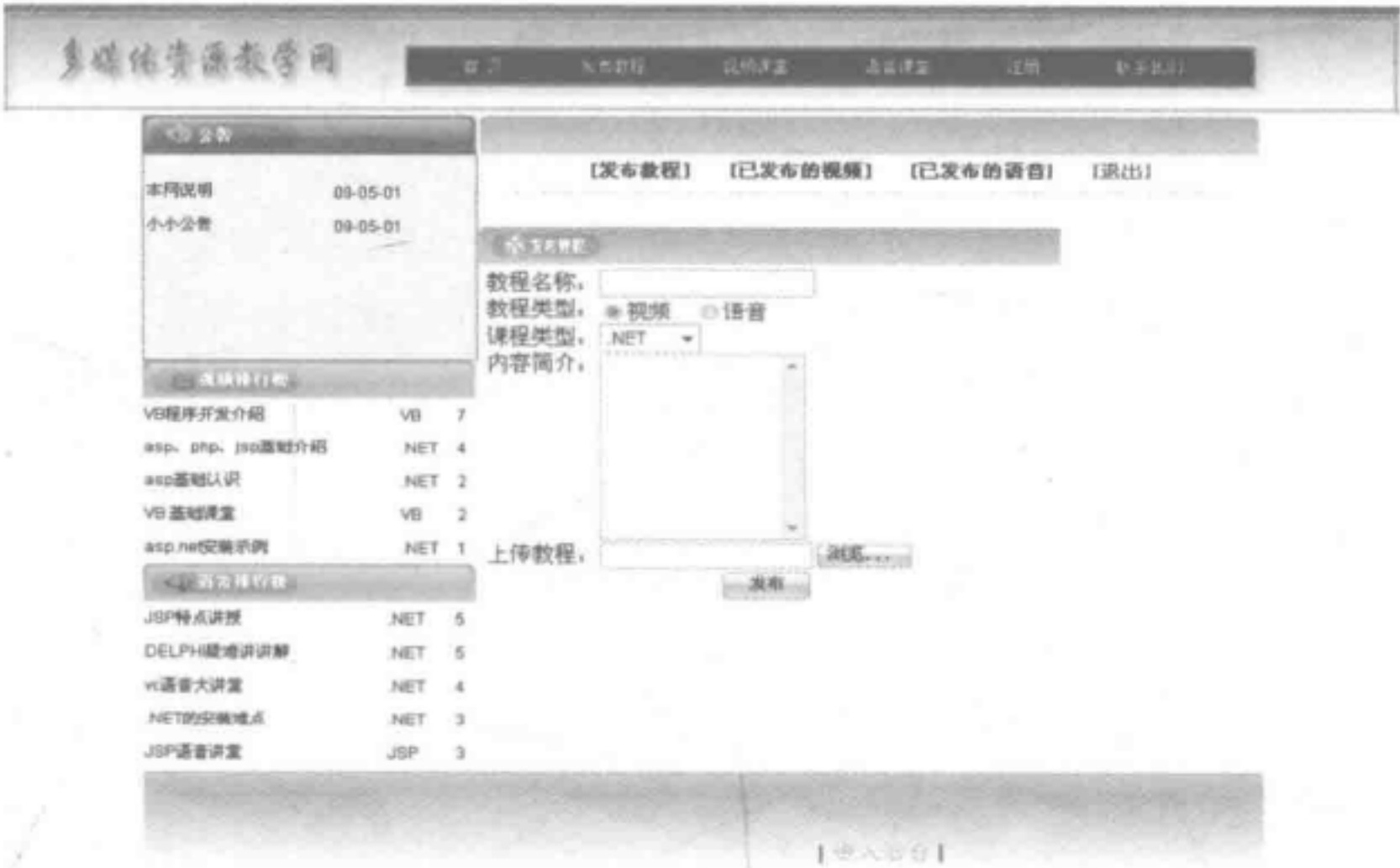


图 2-30 发布教程页面

当用户登录成功后单击“发布教程”按钮，将进入到发布教程页面，发布并管理教程效果图与流程如图 2-31 所示。

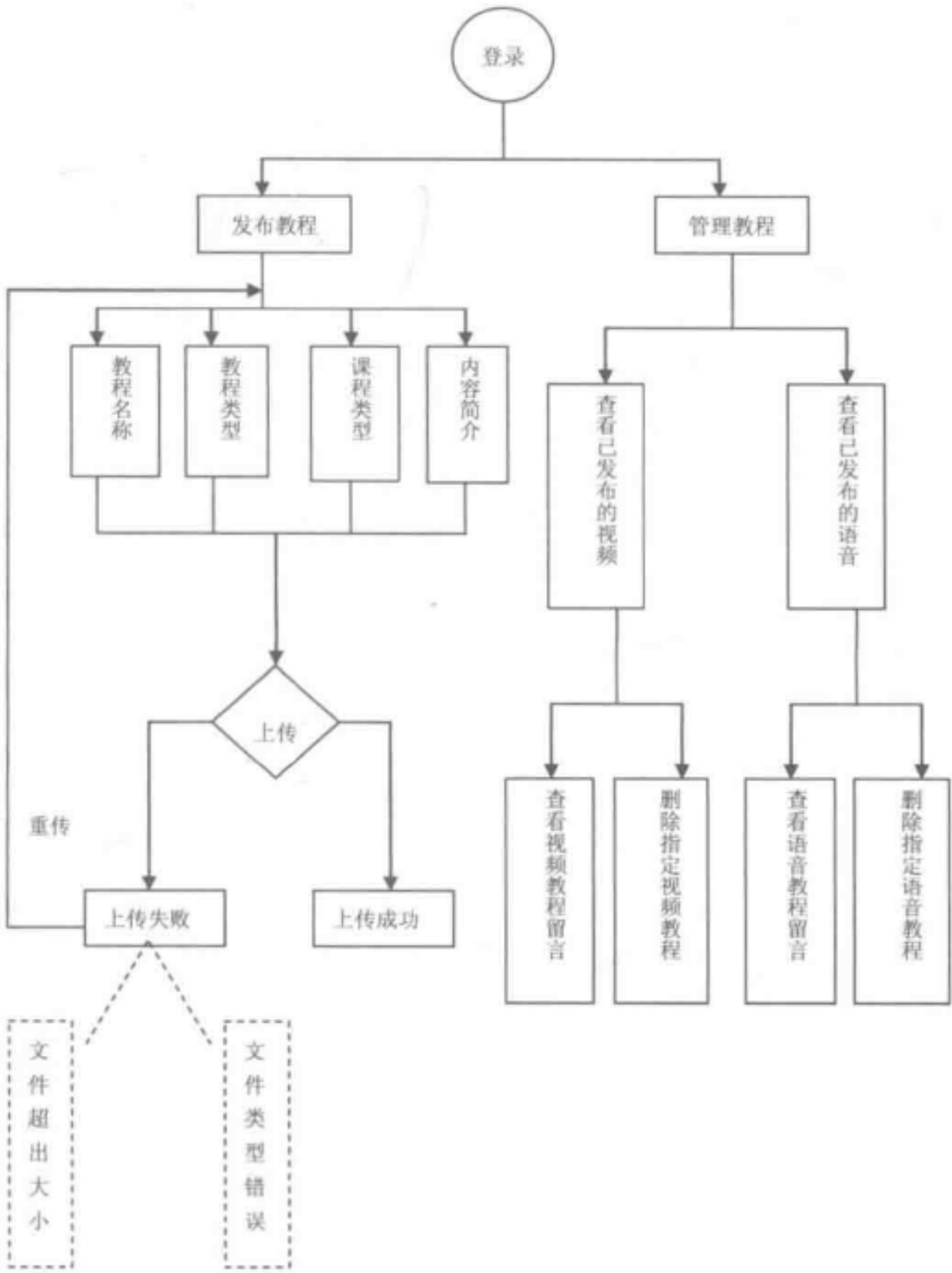


图 2-31 发布并管理教程的流程



## 2.13.1 发布教程

在主页 Web 窗体的加载事件中，需要先判断用户是否登录，如果登录，才可以发布教程或管理自己发布的教程，如果没有登录，将弹出对话框并跳转到首页。而在发布教程时，当用户将教程信息添加正确后，单击“发布”按钮，将用户所发布的教程信息保存到数据库中，此方法的代码如下：

```
protected void btnFB_Click(object sender, EventArgs e)
{
    string vsname = txtName.Text;           //获取教程的名称
    string isVS;
    if (rdibtnSound.Checked)                //获取教程的类型
    {
        isVS = "tb_Sound";
    }
    else
    {
        isVS = "tb_Video";
    }

    string typ = this.ddlLanguage.SelectedValue; //获取语言类型
    string content = txtContent.Text;           //获取内容简介
    string name = Session["UserName"].ToString(); //获取用户登录名
    int clicksum = 0;                          //初始化点击率
    string Path = "";

    try
    {
        string sql = "";
        if (isVS == "tb_Sound")                //判断教程是否是语音类型
        {
            //判断用户上传的文件类型
            if (FileUpload1.FileName.Substring(FileUpload1.FileName.LastIndexOf(".")
                + 1) == "mp3")
            {
                //判断用户上传的语音教程是否存在
                if (!File.Exists(Server.MapPath(".") + "\\Sound\\"
                    + this.FileUpload1.FileName))
                {
                    sql = "insert into tb_Sound(SoundType, SoundName, SoundUrl,
                        ClickSum, SoundContent, Name) values ('"
                        + typ + "', '" + vsname + "', '" + this.FileUpload1.FileName
                        + "', '" + clicksum + "', '" + content + "', '" + name + "')";
                    //设置路径，用于保存上传的语音文件
                    Path = Server.MapPath("./") + "Sound" + "\\"
                        + this.FileUpload1.FileName;
                    if (mydo.adlData(sql))
                    {
                        //将文件保存到指定位置
                        FileUpload1.PostedFile.SaveAs(Path);
                        Page.RegisterStartupScript(
                            "true", "<script>alert('上传成功!')</script>");
                        txtName.Text = "";
                        txtContent.Text = "";
                    }
                }
            }
            else
        }
    }
}
```

```

        {
            Page.RegisterStartupScript(
                "true", "<script>alert('上传失败!')</script>");
        }
    }
    else
        Page.RegisterStartupScript(
            "false", "<script>alert('教程名称已经存在!')</script>");
    }
    else
        RegisterStartupScript(
            "false", "<script>alert('只能上传 mp3 类型!')</script>");
    }
    else
    {
        //判断用户上传的文件类型
        if(FileUpload1.FileName.Substring(
            FileUpload1.FileName.LastIndexOf(".") + 1) == "wmv")
        {
            //判断用户上传的视频教程是否存在
            if (!File.Exists(Server.MapPath(".") + "\\Video\\"
                + this.FileUpload1.FileName))
            {
                sql = "insert into tb_Video (VideoType, VideoName, VideoUrl, ClickSum,
                    VideoContent, Name) values (" + typ + ", '" + vsname + "', '"
                    + this.FileUpload1.FileName + "', " + clicksum + ", '"
                    + content + "', '" + name + "')";
                //设置路径, 用于保存上传的视频
                Path = Server.MapPath("./") + "Video" + "\\"
                    + this.FileUpload1.FileName;
                if (mydo.adlData(sql))
                {
                    FileUpload1.PostedFile.SaveAs(Path);
                    Page.RegisterStartupScript(
                        "true", "<script>alert('上传成功!')</script>");
                    txtName.Text = "";
                    txtContent.Text = "";
                }
                else
                {
                    Page.RegisterStartupScript(
                        "true", "<script>alert('上传失败!')</script>");
                }
            }
            else
            {
                Page.RegisterStartupScript(
                    "false", "<script>alert('教程名称已经存在!')</script>");
            }
        }
        else
        {
            RegisterStartupScript(
                "false", "<script>alert('只能上传 wmv 类型!')</script>");
        }
    }
}
catch (Exception ex)
{
    Page.RegisterStartupScript(
        "false", "<script>alert('上传教程不能为空!')</script>");
}
}

```

## 2.13.2 查看教程页设计

在视频教程详细信息页面中(因视频详细信息和语音详细信息页面相似, 这里主要介绍视频详细信息页面), 用户可以查看教程的发布日期、发布人、该视频的点击率和视频教程的内容简介。当用户观看完视频教程时, 可以通过留言功能发表自己的看法, 或进行技术交流。视频教程的详细信息页面如图 2-32 所示。

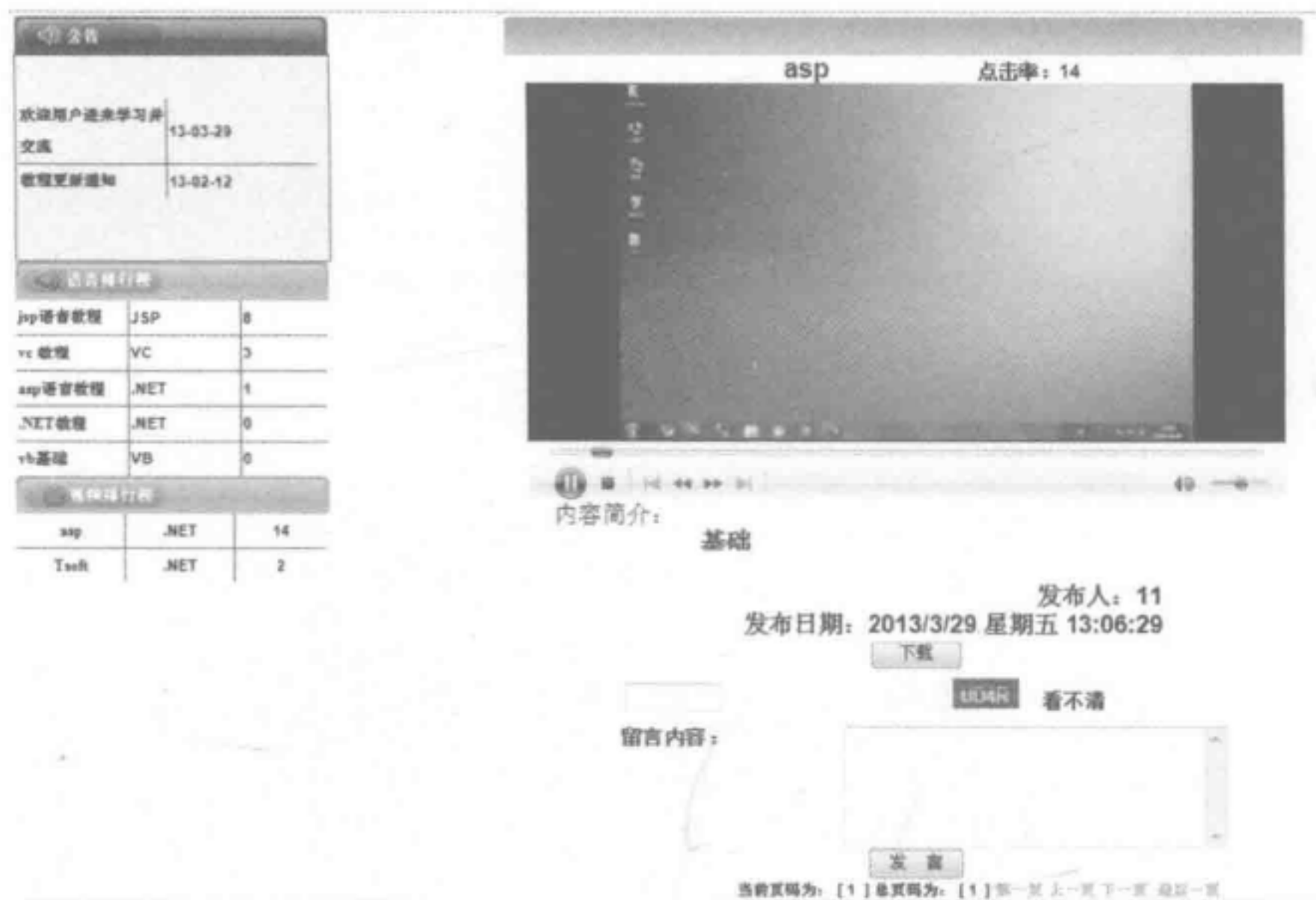


图 2-32 视频教程的详细信息页面

在 Web 窗体的加载事件中, 需要判断是否为首次加载, 如果是首次加载, 需要绑定验证码、调用增加点击率的自定义方法和显示留言的自定义方法。而视频教程需要在每次加载时绑定。在页面中还需要定义几个全局变量, 用来存储视频教程的详细信息。

实现代码如下:

```
public string VUrl; //存储视频路径
public string VideoTitle; //存储视频名称
public string Content; //存储视频内容简介
public string Name; //存储发布人
public string FBDate; //存储发布时间
public string ClickSum; //存储点击率
dataOperate mydo = new dataOperate(); //创建数据库操作类对象
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        this.Image1.ImageUrl = "ValidateCode.aspx";
        addClickSum(); //自定义方法, 增加点击率
        seekSpeak(); //自定义方法, 显示留言信息
    }
    seeVi(); //自定义方法, 绑定视频教程
}
```



在视频教程的下面有一个留言框，发布留言可以使用户相互之间进行交流和讨论，通过“发言”按钮的 Click 事件来实现。在 Click 事件中，需要判断用户是否登录，如果登录，将发言人保存为用户的登录名，如果未登录，将发言人保存为游客。为了防止用户使用非法程序连续发言，使数据库中的数据量过大，这里也使用了验证码功能。

留言框模块的实现代码如下：

```
protected void btnSpeak_Click(object sender, EventArgs e)
{
    string spokesman;
    //判断用户是否登录。如登录，发言人存储登录名。未登录则存储为游客
    if (Session["UserName"] != null)
    {
        spokesman = Session["UserName"].ToString();
    }
    else
    {
        spokesman = "游客";
    }
    string speakContent = this.txtContent.Text; //存储用户发言内容
    string tutorialType = "Video"; //教程类型
    string tutorialID = Request.QueryString["VideoID"]; //视频编号
    string insertSql = "insert into tb_Speak([Spokesman],[TutorialType],
        [TutorialID],[SpeakContent]) values('" + spokesman + "','"
        + tutorialType + "','" + tutorialID
        + "','" + speakContent + "')";
    if (Session["CheckCode"].ToString().Equals(this.txtYzm.Text.ToString()))
        //判断验证码是否正确
    {
        bool bo = mydo.adlData(insertSql);
        if (bo)
        {
            seekSpeak(); //重新绑定留言信息
            Page.RegisterStartupScript("true", "<script>alert('发言成功!');</script>");
        }
        else
        {
            Page.RegisterStartupScript("false", "<script>alert('发言失败!')</script>");
        }
    }
    else
    {
        Page.RegisterStartupScript("false", "<script>alert('验证码错误')</script>");
    }
}
```

## 2.14 后台管理页面的设计

视频讲解 光盘：视频\第2章\后台管理页面的设计.avi

在用户管理页面中，管理员可以对注册用户进行锁定和删除操作。操作主要用于当用户发布了不当的视频或留言时将其锁定，使此用户不能登录。后台的用户管理页面的运行效果如图 2-33 所示。



图 2-33 后台的用户管理页面

### 2.14.1 实现用户管理的页面

首先看用户管理页面的实现，前台 GridView 控件是个关键，GridView 控件用于显示用户并管理用户。将所有的注册用户信息绑定到 GridView 控件时，如果此用户为锁定状态，那么在 GridView 控件中用户的“状态”栏中显示为锁定图标，“操作”栏的提示信息显示为“解锁”。此功能主要使用数据绑定方法来实现，在后台创建两个自定义方法：ImgUrl 方法和 BtnText 方法。ImageUrl 方法主要用来绑定用户的状态图标，BtnText 方法主要用来绑定“操作栏”的值。而在前台 GridView 控件中绑定以上两方法，并传入用户的编号，显示相应的值。GridView 控件的主要代码如下：

```
<Columns>
    <asp:BoundField DataField="Name" HeaderText="用户名" />
    <asp:TemplateField HeaderText="状态">
        <ItemTemplate>
            <asp:Image ID="Image1"
                ImageUrl='<%#ImageUrl(Convert.ToString(Eval("ID")))%>'
                runat="server" />
        </ItemTemplate>
    </asp:TemplateField>
    <asp:BoundField DataField="LoginDate" HeaderText="注册时间" />
    <asp:TemplateField HeaderText="操作">
        <ItemTemplate>
            <a href="change.aspx?id=<%# Convert.ToString(Eval("ID"))%>"
                <%#BtnText(Convert.ToString(Eval("ID")))%></a>
        </ItemTemplate>
    </asp:TemplateField>
    <asp:CommandField HeaderText="删除" ShowDeleteButton="True">
        <ControlStyle Font-Underline="False" />
    </asp:CommandField>
</Columns>
```

在 Web 窗体页面的加载事件中，通过调用自定义 createUser() 方法，将用户的详细信息显示出来。createUser 自定义方法将用户的详细信息绑定到 GridView 控件上。管理员可以将注册用户删除，这里主要还是使用了 GridView 控件中的 RowDeleting 事件，此事件在删

除用户前引发。在此事件中，通过用户的编号，使用 SQL 语句将此用户删除，如果删除成功，将给出相应的提示。实现代码如下：

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    string sql = "delete from tb_login where ID="
        + GridView1.DataKeys[e.RowIndex].Value.ToString();
    if (mydo.adlData(sql))
    {
        Page.RegisterStartupScript("true", "<script>alert('删除成功!')</script>");
    }
    else
    {
        Page.RegisterStartupScript("false", "<script>alert('删除失败!')</script>");
    }
    createUser();
}
```

2.14.2 视频管理模块设计

管理员可以通过视频管理和语音管理模块对所有的教程进行管理，由于视频管理和语音管理基本类似，这里主要介绍视频管理模块。在视频管理页面中，可以看到所有视频，并对其留言进行管理，如图 2-34 所示。



图 2-34 视频管理页面

由于数据信息量不断增大，表格的高度将会增加，这样一来，不但影响界面的美观，而且对界面的性能也有一定的影响。在这种情况下，可以使用 GridView 控件的功能进行分页，每页显示指定的行数即可。GridView 控件自带分页功能，只要设置相关属性，即可实现分页功能。

在此，将 GridView 控件的 AllowPaging 属性设置为 true，表示允许分页；将 pageSize 属性设置为某个正整数，用来控制每个页面中显示的行数，在此设为 5。设置完这两个属性后，在 GridView 控件的 PageIndexChanging 事件中编写代码来实现翻页功能。实现翻页功能的代码如下：



```
protected void grvVideo_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    grvVideo.PageIndex = e.NewPageIndex; //设置当前页的索引值
    grvVideo.DataBind(); //重新绑定 GridView 控件
}
```

## 2.15 系统测试

视频讲解 光盘：视频\第 2 章\系统测试.avi

测试是为了发现程序中的错误而执行程序的过程。好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案。

在对程序进行调试时，可能出现如图 2-35 所示的错误。

“/” 应用程序中的服务器错误。

操作必须使用一个可更新的查询。

说明：执行当前 Web 请求期间，出现未处理的异常。请检查堆栈跟踪信息，以了解有关该错误以及代码中导致错误的出处的详细信息。

异常详细信息: System.Data.OleDb.OleDbException: 操作必须使用一个可更新的查询。

源错误:

```
行 28:      Odbc.Open();
行 29:      OleDbCommand com = new OleDbCommand(sql, Odbc);
行 30:      int i = Convert.ToInt32(com.ExecuteNonQuery());
行 31:      Odbc.Close();
行 32:      if (i > 0)
```

源文件: f:\苏宁\开发项目\实例精通\实例互动学习社区\studyCommunity\App\_Code\dataOperate.cs 行: 30

图 2-35 错误信息

当程序试图更新数据库中的数据或类似操作时，会出现此错误。产生此错误的一般原因是由于没有给数据库写的权限。解决方法如下。

右击 Access 数据库文件，从弹出的快捷菜单中选择“属性”命令，弹出数据库属性对话框。

(1) 选择“安全”选项卡，单击“编辑”按钮，如图 2-36 所示。

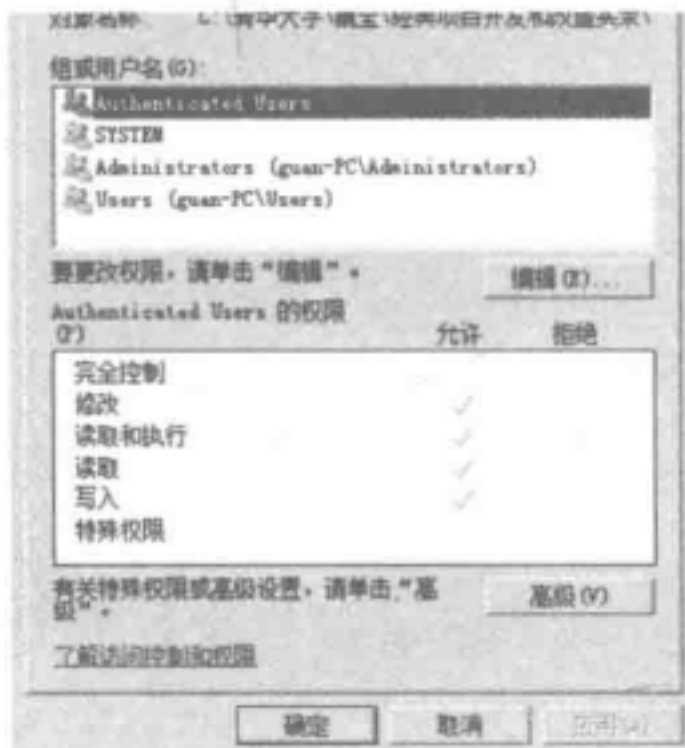


图 2-36 “安全”选项卡

(2) 弹出“db\_study 的权限”对话框，选择“添加”按钮。

(3) 弹出“选择用户或组”对话框，在其中的“输入对象名称来选择”文本框中输入字母 e 后，单击“检查名称”按钮，在“输入对象名称来选择”文本框中将显示 Everyone 用户组，单击“确定”按钮。

此时，Everyone 用户组已经添加到“安全”选项卡中的“组或用户名称”文本框里。选择 Everyone 用户组，下面将显示出 Everyone 的权限，选中“Everyone 的权限”列表框中的“完全控制”的“允许”复选框，最后单击“确定”按钮。

### 1. 错误信息

本系统有如下 3 类错误信息。

(1) 用户输入时的输入错误提示：对所有的用户输入做合法性检查。

(2) 运行中的错误：主要是数据库连接错误或动态生成的 SQL 脚本错误。

(3) 页面错误：一般是网络中断或程序中的 Bug 引起的。浏览器将会提示相应的错误信息。

### 2. 补救措施

系统故障或瘫痪后，可能采取的措施如下。

#### (1) 后备技术

该系统在页面设计上尽量做到相对独立，即在局部页面的错误或损坏不会影响其他模块的运行，这就避免了整个网站不能访问。

#### (2) 恢复和再启动技术

系统管理员定期在本地备份数据库，当原始系统数据万一丢失时，重新上传页面文件和数据库，则整个系统即可恢复正常运行。





## 第3章 大东科技人事管理系统

高帆项目开发

在本章的内容中，我们将详细讲解使用 Visual Studio 和 SQL Server 开发一个企业人事管理系统的过程。

该企业人事管理系统采用 C/S 结构，主要对企业员工的信息以及人事相关的工作流程进行集中管理，方便企业建立一个完善的、强大的员工信息数据库。

使用 C# 设计操作控件和编写操作程序，完成数据输入、修改、存储、调用、查询等功能；并使用 SQL Server 数据库形成数据表，进行数据存储。



### 赠送的超值电子书

- 021 表达式
- 022 基本运算符
- 023 数学运算符
- 024 赋值运算符
- 025 比较运算符
- 026 逻辑运算符
- 027 移位运算符
- 028 三元运算符
- 029 运算符的优先级
- 030 语句和语句块

## 3.1 程序员职场生存必杀技

视频讲解 光盘：视频\第3章\程序员职场生存必杀技.avi

程序员的程序人生是由很多个多姿多彩的阶段构成的，在这些众多的阶段中，因为职场生涯阶段占据了程序人生中的绝大部分历程，所以总有人说职场就是程序员的人生。

那么，究竟怎样做，才能使自己的职场生涯变得多姿多彩呢？在本节的内容中，我们将与读者探讨程序员在职场中如鱼得水的秘籍。

### 3.1.1 程序员的生存现状

IT 在中国的大发展不过是近 30 年的事，但却是风起云涌、豪杰四起的 30 年。不可否认，在中国的 IT 人中诞生了许多天才和富豪，但是这里比较关心的还是人数最多的大众 IT 人的生存现状，毕竟是千千万万的“他们”盖起了中国的 IT 大厦。

#### (1) 高薪，个体差异巨大

IT 从业者的确高薪，但是个体贫富差距很大，往往刚刚入门的程序员和高级顾问的收入差距会有几十倍。在中国的 IT 业中，拿低薪和高薪的人少，大部分都在中间徘徊。

#### (2) 低龄化

中国 IT 专业人员的年龄主要集中在 21~35 岁，其中 26~30 岁的比例最高，占到四成；其次是 21~25 岁的人群，略少于前者；31~35 岁的居第三位，不足两成；剩下的不足半成的是其他年龄段的，大都是 35 岁以上的开发人员。

#### (3) 理科天才的汇聚地

IT 行业是一个理科天才的聚集地，是个对个人能力要求比较高的行业，没有真本事会立刻被踢出队伍。程序员研究的可能是毫无趣味的代码和飘渺的算法，但是创造出来的却是极大方便人们解决问题的各种软件，这些软件功能强大，却简单易用。

#### (4) 改变了人们的生活

IT 正以前所未有的速度改变着人们的生活，用“神奇”来说一点也不为过。阿里巴巴和京东已经改变了人们的购物习惯，谷歌的技术创新产品已经彻底颠覆了人们的生活形态。

### 3.1.2 中外 IT 领域的企业文化

除了高校和科研院所，IT 人才基本上都是集中在各个公司的。而各 IT 公司因为文化背景、行业领域、公司规模等方面的差异，也存在着不同的企业文化。

#### (1) 欧美企业文化

计算机技术起源于美国，当然欧洲也贡献了大量的优秀数学家和计算机学者，所以欧美(当然也包括日本)的 IT 技术是全球领先的。欧美公司的技术含量是最高的，而且欧美公司是每个 IT 精英都最希望去的理想企业。欧美公司比较人性化，工作环境十分自由开放，而且上下级各个阶层的交流都非常直接，有些地方还会实行弹性工作制，就是尽最大限度让员工感到安逸，进而很自愿、很自觉、很自发地努力工作，证明自我。



欧美公司(主要是美国的公司)比较根本的原则就是以人为本。欧美公司认为一个企业最重要的实力就是人才,而人才是需要被尊重、保护的,所以那些企业鼓励员工学习,努力让员工感到适应,从而心无旁骛。

欧美公司对人才的重视,也体现在其招聘人才、构建最适合工作的环境、员工福利方面。虽然欧美公司招聘人才时不拘一格,但是想要进去,恐怕还是需要自己能头顶“精英”的光环才行。

### (2) 日韩企业文化

日本和韩国的IT企业在中国也有不小的规模,尤其是日本和韩国(主要是日本)的软件外包业务占中国的软件外包业务总量的比重很大,所以那些日韩外包的接包企业也或多或少地受到了日韩企业文化的影响。日韩企业的管理方式就是吸血鬼式的管理,难听点说,就是挖空心思地榨取员工的每一滴血。在那里,不加班是不合群的表现。另外,日韩的IT企业十分讲究团结,跟欧美公司最大的不同就是日韩企业讲究团体实力,而欧美公司更加侧重于天才战术。

韩国的IT公司和日本的差不多,不过韩国的公司对员工的企业文化培养比较重视,真正工作之前,会接受比较全面的培训。并且韩国IT公司的纪律也是非常严格的,这一点与日本公司差不多,所以有时会感觉很受限制。其次,韩国的升职制度也比较成问题,所以在韩企升到高层比较困难,不过韩国公司的坚强干劲还是很值得学习的。

### (3) 中资企业文化

国内的IT公司在IT大发展的30年间逐步改进,受到了欧美和日韩等国家的企业文化的影响,再结合国内传统企业文化的积淀,逐步形成了比较中庸的企业文化。

我国的IT发展时间不长,而IT在短短的历史中诞生了大量的技术。因此大量技术更类似于“涌入”的,再加上欧美等其他国家公司的进入,对我国的IT企业造成了多方面的影响,所以显得国内的IT企业看起来就像是欧美企业和本国传统企业的混合体。

国内的中小型IT企业远多于大型IT企业,这也就使得国内IT行业中的团队模式一般规模不会很大,分工也不是很细致,比较类似于欧美的自由模式,但又很像日韩的官僚等级,其实官僚主义并不是什么特征,任何比较大型、称得上“帝国”的公司都会滋生这种东西,比如IBM。

国内的IT企业文化还在不断地探索自己的道路,既接受着外国企业的文化冲击,又摸索着自己的道路。国外的文化不一定全有用,不可单纯地搞“拿来主义”。国内的企业需要自己的特色。比如,国内的员工私下关系一般都很好,而外国的企业中,员工在很大程度上都是为了工作才交际。

## 3.1.3 赢在职场——修炼程序员职场秘籍

### 1. 与客户的相处之道

在职场中,我们不可避免地需要与客户沟通、相处。作为一名程序员,当然也不可能例外。当面对性格各异客户时,怎样才能与它们好好相处,从而顺利地完成任务呢?在此,提出如下4条建议。

(1) 都说客随主便,在对方的“地盘”上,我们要矜持。在客户单位中时,我们要少



谈论业务之外的问题，并且在讲话时要正儿八经，声音响亮，不能给外人一个偷偷摸摸的印象。并且要谨记，绝不允许在客户所在单位里谈任何“潜规则”之类的东西，否则，只要出现过一次这种事情，客户便会对你非常反感的。

(2) 一定要保证我们作品的质量，无论是 Web 项目还是桌面项目。不能让客户承担任何可能的风险，如果客户会因为接受你的设计方案而感受到风险的话，肯定会退缩。

(3) 要跟客户说实话。不要自以为是地认为自己够聪明，其实，客户并不“傻”。客户既然能代表公司谈合作，他肯定是其公司里最了解计算机的一名代表，并且已经接触了不只你一家软件公司。他甚至会了解市面主流成本是多少钱。大多数客户都非常反感在谈价格时说“低了多少钱，我连本都赚不回来”之类的话。一般负责项目洽谈的代表都是由对计算机这一行业非常了解的人承担的，如果想蒙骗客户，客户会直接把你拒绝掉。

(4) 在客户单位中时，只要见到与客户在同一个办公室的任何人员，我们务必谦逊。不管他是负责这个项目，还是不负责这个项目。你要对客户代表周围的人表现出极大的尊敬和热情。当然也要有一个度，只要让他身边的人认为你是值得信赖的人即可，不要过度地认为你是在巴结他们。

## 2. 注意沟通

说起职业程序员，可能大家脑子里马上出现的是西服、领带、公文包的精干白领形象，其实，作为程序员来说，还应具备很多内在的职业素养。究竟如何才能提高自己的职业素养呢？以下分享几个小技巧。

(1) 尽量提高自己的表达能力和沟通能力，做到表达明确。

如果客户不能清楚表达需求，我们可以通过良好的表达和沟通能力来融入到客户组织内部，了解客户的工作流程，与客户共同更好地、更准确地定义和分析需求。程序员不仅要具备技术能力，表达能力也是必备的基本能力之一。人们都说与客户进行沟通很重要，但是，准确表达自己的观点和意见，才是成功沟通的基础。

(2) 使用多种方式了解需求，多一个途径就多一分成功率。

需求很重要，只有需求明确了，我们才能有目的地设计程序。了解需求的常用方法有问题分析法和建模分析法，当然，也可以是上述两种方法的结合，例如，在问题分析法中应用建模分析法。在与客户的员工谈话时，要遵循面向工作流程、面向任务、面向角色，也就是用“面向对象”的思想，帮助客户理清思路。

(3) 不要臆测客户的需求。

一般正规的开发公司都设有专门的需求工程师。当技术人员在编码过程中遇到需求不明确时，必须与项目经理或需求工程师及时沟通，程序员不能自作主张地猜测客户的需求。

(4) 不过度承诺，避免搬起石头砸自己的脚。

都说“搬起石头砸自己的脚”这种情况在软件开发中非常常见。的确，有很多程序员向客户大包大揽，承诺了不能实现的功能。所以，在定义需求阶段，一定要向客户说明“什么是我们能做的，什么是我们应该做的，什么是我们不能做的”。如果因为过度承诺而无法完成相应的功能，不但会给开发人员造成严重的挫折感，并且也会给客户带来不好的印象，甚至可能会造成违约，而需要负法律责任。

### 3. 同事交往经验谈

同事之间很容易形成利益关系，如果我们一味地对一些小事不能正确对待，会很容易形成隔阂。

在日常的同事交往中，只要把握好以下几个方面，就可以建立起融洽的同事关系。

#### (1) 以大局为重，务必少拆台

对于同事的缺点，因为大家都是由于工作关系而走在一起的，所以就要有集体意识，以大局为重，形成利益共同体。特别是在与外部人员接触时，要形成“团队形象”的观念，多补台、少拆台，不要为自身小利而破坏集体大利，最好“家丑不外扬”。

#### (2) 对待分歧，要求大同、存小异

同事之间由于经历、立场等方面的差异，对同一个问题，往往会产生不同的看法，引起一些争论，一不小心就容易伤和气。因此，与同事有意见分歧时，不要过分争论。当面对问题时，特别是在发生分歧时，要努力寻找共同点，争取求大同、存小异。实在不能一致时，不妨冷处理，表明“我不能接受你们的观点，我保留我的意见”，让争论淡化，又不失自己的立场。

#### (3) 对待升迁、功利，要保持平常心，不要嫉妒

许多同事平时一团和气，然而，遇到利益之争时，就当“利”不让。经常在背后互相谗言，或嫉妒心发作，说风凉话。建议时刻保持一颗平常心。

#### (4) 与同事、上司交往时，保持适当距离

在一个单位中，如果几个人交往过于频繁，容易形成表面上的小圈子，容易让别的同事产生猜疑心理，让人产生“是不是他们又在谈论别人是非”的想法。因此，在与上司、同事交往时，要保持适当的距离，避免形成小圈子。

#### (5) 在发生矛盾时，要宽容忍让，学会道歉

同事之间，经常会出现一些磕磕碰碰，如果不及时妥善处理，就会形成大矛盾。所以，在与同事发生矛盾时，要主动忍让，从自身找原因，换位为他人多想想，避免矛盾激化。当你勇于向对方道歉后，可能更会加深你们之间的感情。

## 3.2 系统介绍

视频讲解  光盘：视频\第3章\系统介绍.avi

在21世纪，什么是最重要的呢？葛优说过“人才最重要！”。

人才作为企业发展的核心竞争力，在企业的发展中发挥着不可比拟的作用。人事管理是人力资源管理发展的第一阶段(有时也作为广义的“人力资源管理”的代称)，是有关人事方面的计划、组织、指挥、协调、信息和控制等一系列管理工作的总称。应该通过科学的方法、正确的用人原则和合理的管理制度，调整人与人、人与事、人与组织的关系，谋求对工作人员的体力、心力和智力做最适当的利用与最高的发挥，并保护其合法的利益。

所以，良好的人才管理系统也便成为企业管理的一部分。一个现代化的企业人事管理系统有助于企业节约成本、提高效率，而且还可以使领导者更清楚地了解到企业员工的相



关资料，从而更合理地制定相关的人事信息。

### 3.2.1 系统背景介绍

信息化的迅速发展，互联网的高速蔓延，使企业的信息化管理出现了新的方向。一个现代化的企业要想生存和发展，必须跟上信息化的步伐，用先进的信息化技术来为企业的管理节约成本、制定规划。而人才，作为企业生存和发展的根本，在企业的管理中始终占有重要的地位。对企业的人才进行良好的人事管理，既有助于企业高层和人事管理人员动态、及时地掌握企业的人事信息，制定人才招聘和发展规划，也有利于企业优化改革，精简机构，最终实现人事管理的信息化建设。在此形势下，我们开发了这套人事管理系统，可应用于大部分企事业单位，管理人员可查询员工考勤、薪资、档案等相关信息，并可对其进行维护，普通员工可在管理人员授权后进行相应的查询等操作。

国外专家学者对人事管理系统的研究起步比较早，发达国家的企业非常注重自身人事管理系统的开发。特别是一些跨国公司，更是不惜花费大量的人力和物力来开发相应的人事管理系统，通过建立一个业务流的开发性系统，实现真正意义上的人事管理目标，挑选和留住最佳人才，同时不断提高这些人才的工作效益。

我国的信息管理系统是 20 世纪 90 年代初开始快速发展的。经过 20 余年的发展，我国的数据库管理技术也广泛地应用于各个领域，并且形成了产业化。但是，我们的工厂、企业对信息管理系统的的应用比起世界先进水平还相当落后。主要表现在：人事管理系统的使用范围相对狭窄、人事管理系统的功能相对有欠缺、稳定性较差、功能相对单一。

### 3.2.2 应用的目的与意义

人员的管理是一切管理工作的核心。员工代表一个企业的形象，因而人事管理机制设计得好坏，直接影响一个企业的成败。

在人事管理机制中，员工的档案管理是企业人事管理的基础，在企业员工流动性普遍增强的今天，拥有一个准确而及时的人事管理系统，有利于人事部门对员工的流动进行分析、管理，为企业提供人力资源保障。

通过把人力资源部的那些重复的、事务性的工作交给 HRP(Human Resource Planning, 人力资源管理系统)来解决，就可以免去用户以往人力资源管理工作繁琐和枯燥；用领先的人力资源管理理念，把人力资源管理的作业流程控制和战略规划巧妙地集合于一体。

系统重点涉及到人力资源管理工作中的薪资、考勤、绩效、调动、基本信息、用户管理以及用户切换等方面，并有综合的系统安全设置、报表综合管理模块，可以很好地为用户的人力资源管理部门在对员工的成本管理、知识管理、绩效管理等综合管理方面给予帮助。以每个月的“发工资”为例，考勤、人事信息变动、奖惩、迟到和旷工对本月的薪资计算都有影响；为了及时地计算和发放工资，往往要提前一个星期花费大量的时间、加班加点才能及时完成，而这样做，无论从工作效率还是准确度方面，都不是理想的，会浪费不少的人力和财力。

如果改用 HRP 管理，就能做到高效、高精度，还可以减少管理时带来的一些繁琐的工作，可以节约管理上的支出。



### 3.2.3 人事管理系统的发展趋势

人力资源管理系统将主导 21 世纪,无论是发达国家还是发展中国家,对人力资源的战略意义都有深刻的认识,并开始付诸行动。这种状况的变化起因于竞争压力。目前,世界经济趋向全球化。世界经济的全球化过程和国家的开放过程,要求组织的管理部门降低管理成本,以减少竞争压力和增强竞争力。

随着社会政治和经济的发展,人们的工作目标和价值观也都发生了重要的变化。这就对人事管理部门和管理人员提出了新的要求和新的问题,不得不考虑诸如工作类型设计、岗位分析、充分尊重员工以及为他们提供良好的个人发展空间和自我价值实现的环境与条件等问题。这样,人事管理系统就派上了用场。

## 3.3 系统需求分析

视频讲解 光盘: 视频\第 3 章\系统需求分析.avi

软件的开发是一个系统的工程,需要开发人员对软件工程有一个深层次的了解。软件工程是一门研究用工程化方法构建和维护有效的、实用的和高质量的软件的学科。它涉及到程序设计语言、数据库、软件开发工具、系统平台、标准、设计模式等方面。由此可见,软件工程在软件开发的过程中始终贯穿于整个工程中。所以,作为开发人员,要从始至终都要遵循软件工程的要求来进行具体的开发。软件工程的目标是:在给定成本、进度的前提下,开发出具有适用性、有效性、可修改性、可靠性、可理解性、可维护性、可重用性、可移植性、可追踪性可互操作性和满足用户需求的软件产品。追求这些目标有助于提高软件产品的质量和开发效率,减少维护的困难。

软件工程过程主要包括开发过程、运作过程、维护过程。它们覆盖了需求、设计、实现、确认以及维护等活动。

需求活动包括问题分析和需求分析。问题分析获取需求定义,又称软件需求规约。需求分析生成功能规约。

设计活动一般包括概要设计和详细设计。概要设计建立整个软件系统的结构,包括子系统、模块以及相关层次的说明、每一模块的接口定义。详细设计产生程序员可用的模块说明,包括每一模块中数据结构的说明及处理描述。

实现活动把设计结果转换为可执行的程序代码。

确认活动贯穿于整个开发过程,实现完成后的确认,保证最终产品满足用户的要求。

维护活动包括使用过程中的扩充、修改与完善。伴随以上过程,还有管理过程、支持过程、培训过程等。

人事管理系统分析需要对当今社会的人事管理方面的需求进行认真而全面的调查。

根据企业对人事管理系统的功能需求、业务操作规程及数据结构等的具体要求,调查了企业人事管理部门对员工基本信息、员工调动、员工奖罚、员工培训、员工考评、员工调薪、员工职称评定的各种管理要求,确定了系统的性能要求、系统运行支持环境要求,以及数据项的名称、数据类型、数据规格。以上这一切,为下一步的开发工作奠定了良好

的基础。

软件需求说明必须全面、概括性地描述人事管理系统所要完成的工作，使软件开发人员和用户对本系统中的业务流程及功能达成共识。开发人员通过需求说明，可以全面了解人事管理系统所要完成的任务和所能实现的功能。

管理员登录用例如图 3-1 所示。

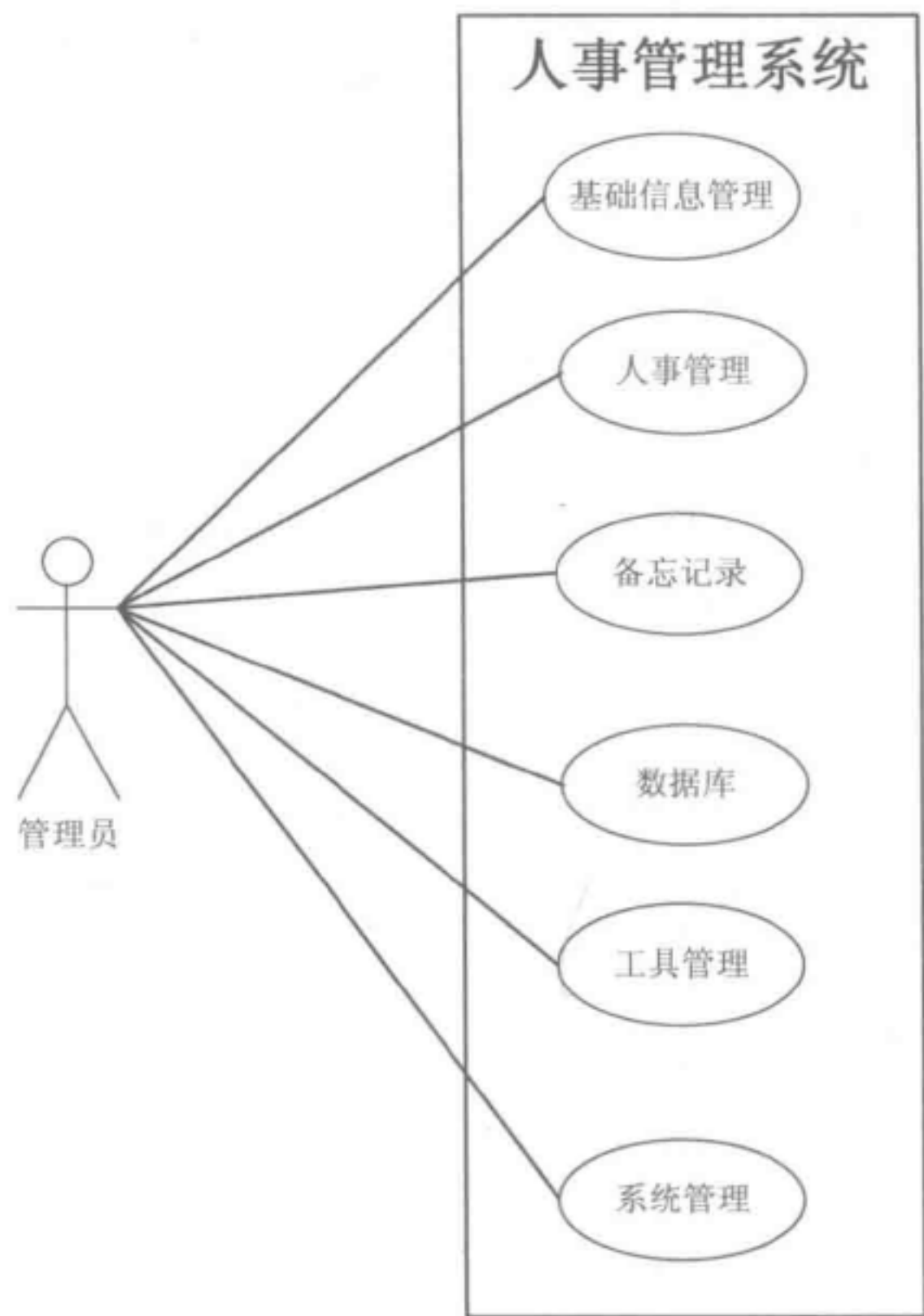


图 3-1 管理员登录用例

在表 3-1 中，给出了用户登录描述信息。

表 3-1 用户登录描述

| 用例名称 | 登录描述                                                |
|------|-----------------------------------------------------|
| 功能简述 | 管理员、员工需提供正确的用户名和密码，才能进入本系统                          |
| 前置条件 | 无                                                   |
| 后置条件 | 用户登录成功                                              |
| 基本流  | 用户在登录页面中输入用户名和密码，并提交系统，判断用户名和密码是否合法，根据用户的类型显示不同的主页面 |
| 扩展流  | 如果用户名或密码不合法，则返回登录页面，并给出错误信息                         |
| 备注   | (无备注，则删除本行)                                         |

在表 3-2 中，给出了用户权限信息。

表 3-2 权限用例分析

| 用例名称 | 权限分类                         |
|------|------------------------------|
| 功能简述 | 用户需提供正确的用户名和密码，进入系统后，拥有不同的权限 |
| 前置条件 | 无                            |
| 后置条件 | 用户注册、登录成功                    |
| 备注   | 用户注册时，必须输入正确的格式              |

在表 3-3 中，介绍了系统人事管理信息。

表 3-3 人事管理用例分析

| 用例名称 | 人事管理                                                                       |
|------|----------------------------------------------------------------------------|
| 功能简述 | 登录后，可根据需求，查看相关种类的信息，并进行修改                                                  |
| 前置条件 | 无                                                                          |
| 后置条件 | 必须是管理员登录                                                                   |
| 基本流  | 1. 管理员输入正确的用户名和密码<br>2. 进入主页面<br>3. 点击“人事管理”菜单<br>4. 进入人事管理界面<br>5. 进行信息修改 |
| 备注   | 可直接点击查看详细信息                                                                |

## 3.4 系统设计

视频讲解  光盘：视频\第3章\系统设计.avi

软件项目开发的第一步，是系统设计分析和项目需求分析。

在本节的内容中，将详细讲解该人事管理系统设计的具体分析工作，为步入后面的具体编码工作打下基础。

### 3.4.1 系统设计目标

根据企业对人事管理的要求，制定企业人事管理系统的目标，如下所示：

- 操作简单方便、界面简洁美观。
- 在查看员工信息时，可以对当前员工的家庭情况和培训情况进行添加、修改、删除操作。
- 方便快捷的全方位数据查询。
- 按照指定的条件对员工进行统计。
- 可以将员工信息以表格的形式插入到 Word 文档中。



- 实现数据库的备份、还原及清空操作。
- 由于该系统的使用对象较多，要有较好的权限管理。
- 能够在当前运行的系统中重新进行登录。
- 系统运行稳定、安全可靠。

### 3.4.2 系统功能设计

在整体设计中，我们将企业人事管理系统分为 6 个部分，即基础信息管理、人事管理、备忘录、数据库、管理工具、系统管理。

系统功能的结构如图 3-2 所示。



图 3-2 系统功能的结构

下面将具体介绍每个功能。

#### (1) 基础信息管理

主要功能包括：数据基础和员工提示信息两个部分。基础信息管理的数据库流程如图 3-3 所示。

#### (2) 人事管理

人事管理功能包括：人事档案浏览、人事资料查询、人事资料统计三大的部分。本章人事管理的数据库流程如图 3-4 所示。

#### (3) 备忘录

对日常记事信息进行添加、修改、删除及查询操作，对通信信息进行添加、修改、删除及查询操作。

备忘录管理的数据库流程如图 3-5 所示。

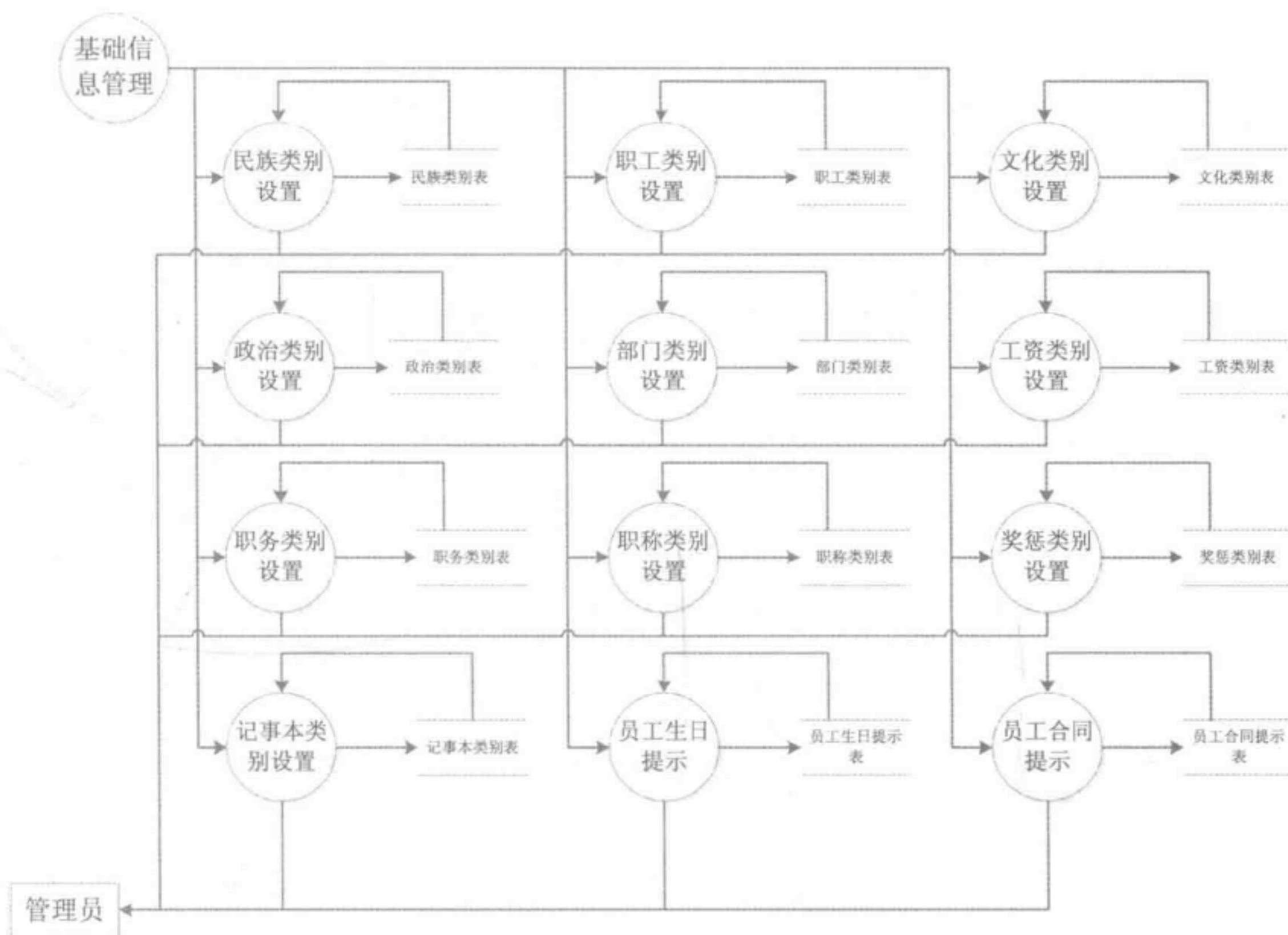


图 3-3 基础信息管理的数据流程

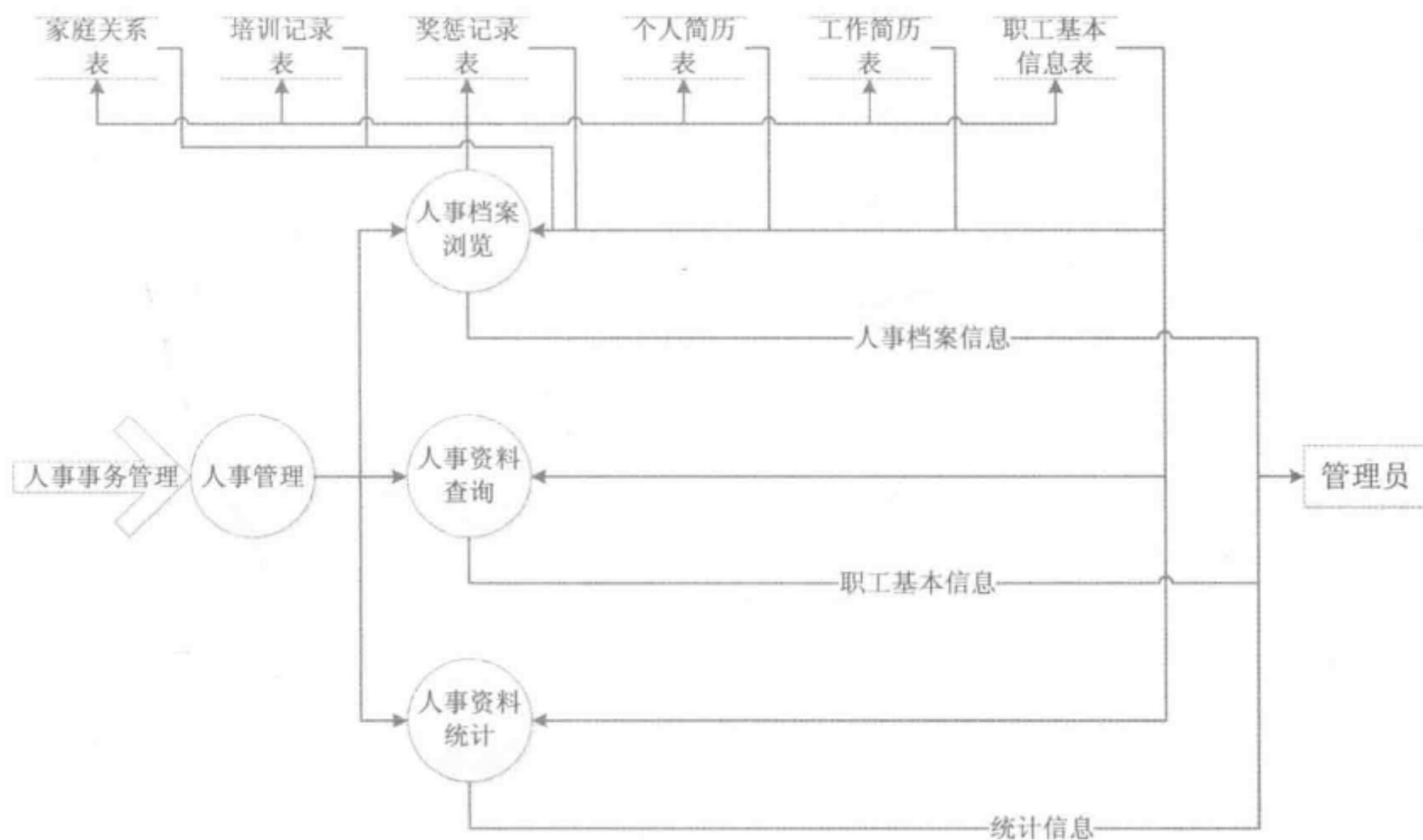


图 3-4 人事管理的数据流程

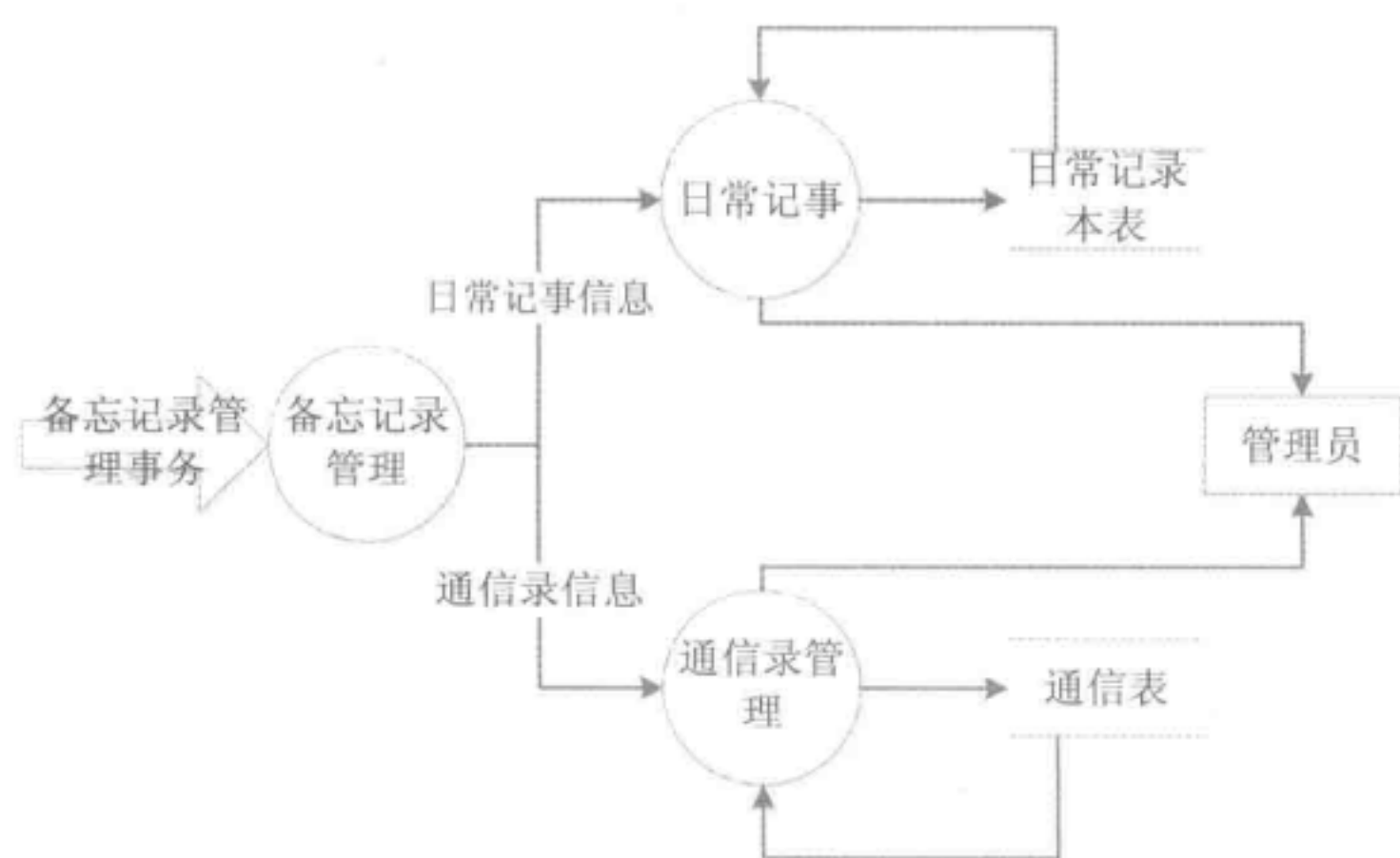


图 3-5 备忘录管理的数据流程

#### (4) 数据库

可对数据库进行备份、恢复及清空数据库操作。

#### (5) 管理工具

可直接调用计算器和记事本的快捷方式。

#### (6) 系统管理

可对本系统进行重新登录、用户设置，及系统退出的操作。

系统管理的数据流程如图 3-6 所示。

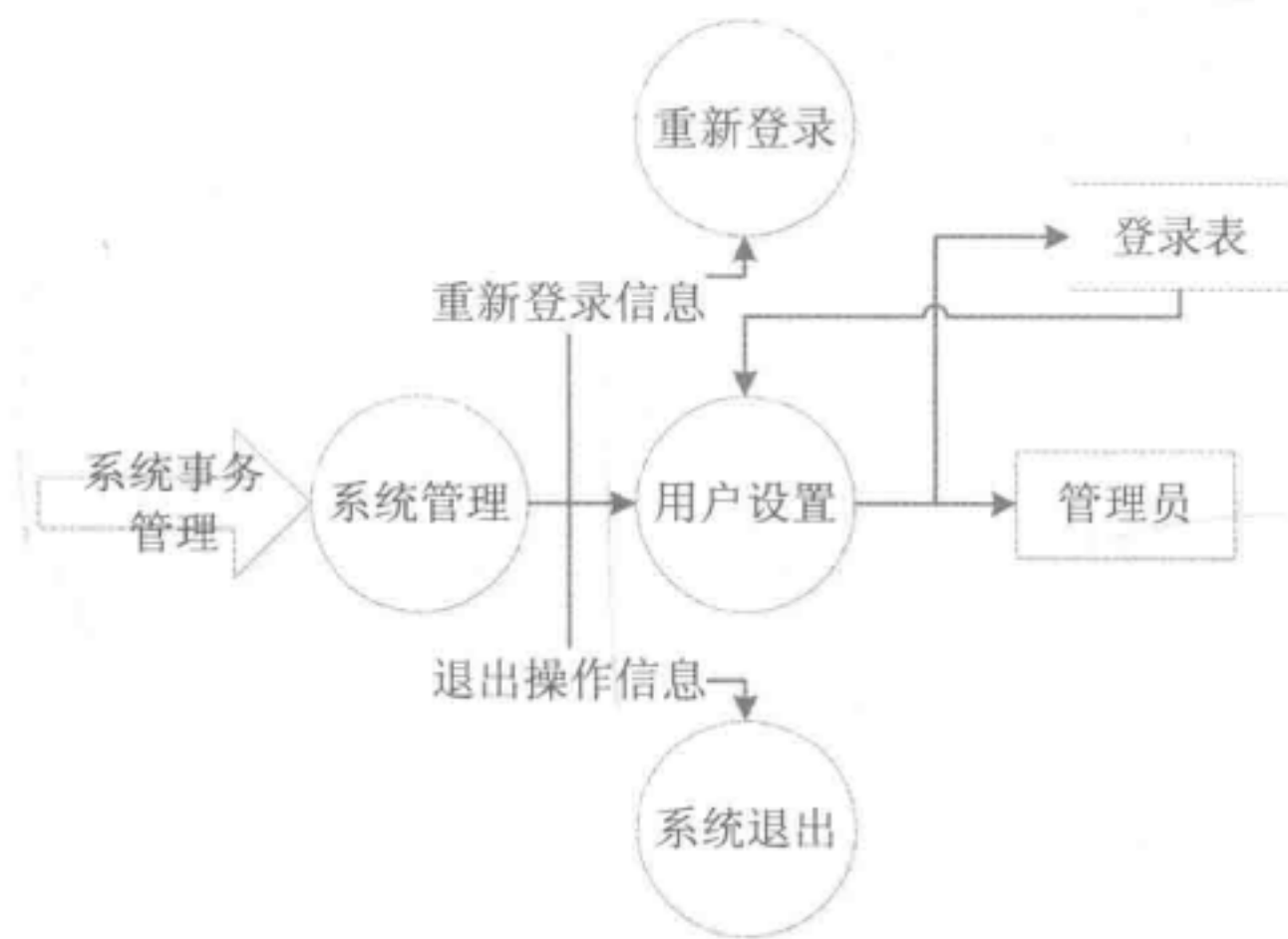


图 3-6 系统管理的数据流程

其实，作为一个更为完善的企业人事管理系统，应提供更为便捷与强大的信息查询功能，例如相应的网络操作及服务。由于开发时间和计算机数量有限，该系统并未提供这一功能。对信息的保护手段仅限于设置用户级别，以及提供数据文件的备份，比较简单，不能防止恶意破坏，安全性能有待进一步完善。



## 3.5 数据库设计

视频讲解 光盘: 视频\第3章\数据库设计.avi

在开发应用程序时,对数据库的操作是必不可少的,数据库设计是根据程序的需求及其实现的功能来制定的,数据库设计的合理性将直接影响到程序的开发工程。

### 3.5.1 数据库描述

数据库是数据管理的最新技术,是计算机科学的重要分支。近几年来,数据库管理系统已从专用的应用程序包发展成为通用系统软件。由于数据库具有数据结构化、最低冗余度、较高的程序与数据独立性、易于扩充、易于编制应用程序等优点,较大的信息系统都是建立在数据库设计之上的。

由于用到的数据表格多,另外考虑到实际情况,企业人事基本信息的变动,还有员工信息的多少的变化,我们选用 SQL Server 作为数据库进行开发,而不用 Access,主要是因为 Access 在实际运用中不适合此系统;而 SQL Server 是一种常用的关系数据库,能存放和读取大量的数据,管理众多并发的用户。

### 3.5.2 数据库分析

企业人事管理系统主要用来记录一个企业中所有员工的基本信息,以及每个员工的工作简历、家庭成员、奖惩记录等,数据量是根据企业员工的多少来决定的,本系统使用 Microsoft SQL Server 2008 作为后台数据库,数据库命名为 db\_PWMS,其中包含了 23 张数据表,用于存储不同的信息。

### 3.5.3 数据库概念设计

数据库设计是系统开发过程中的重要部分,它是通过分析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做详细的说明。

#### (1) 用户登录数据设计

在本系统中,为了提高系统的安全性,每个用户都要使用正确的用户名和密码,才能进入主窗体,为了能够记录正确的用户名和密码,应在数据库中创建登录表。登录表的实体 E-R 图如图 3-7 所示。

为了避免登录用户随意修改数据库中的信息,本系统应创建一个用户权限表,用于记录用户对程序中各窗体的操作权限,由于用户权限表与登录表是密切相关的,所以在权限表中必须有用户编号,以方便登录后在权限表中调用相关的权限。用户权限表的实体 E-R 图如图 3-8 所示。

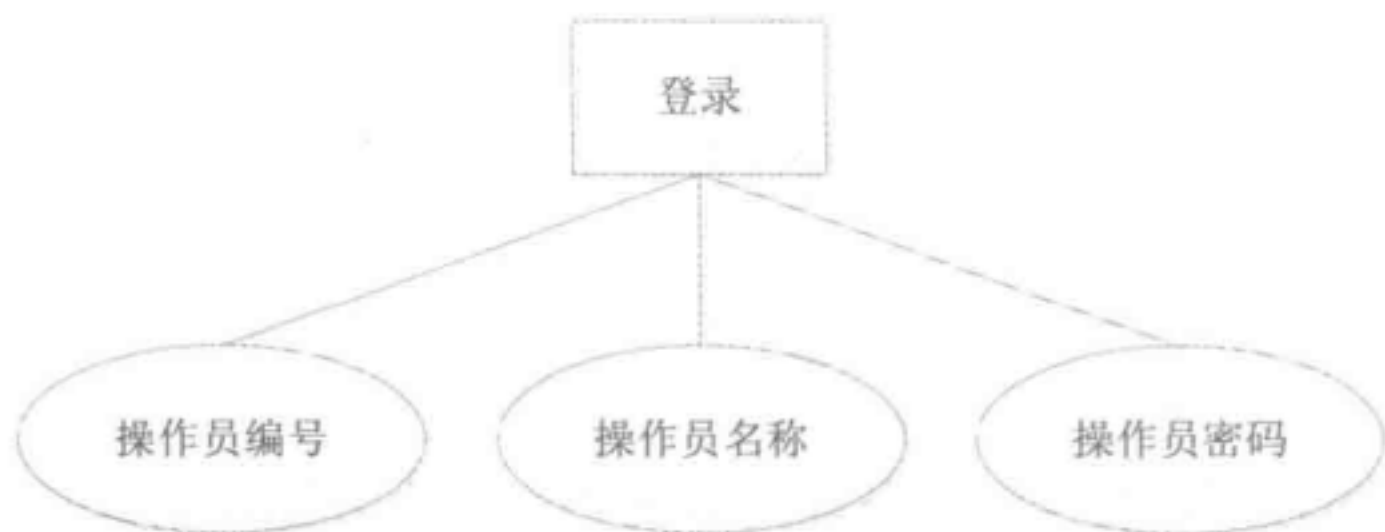


图 3-7 登录表的实体 E-R 图



图 3-8 用户权限表的实体 E-R 图

为了可以在用户权限表中更方便地添加用户权限信息，可以在数据库中创建一个权限模块，该模块中记录了系统中所有涉及的权限名(也就是权限所对应的窗体名称)，可以在添加用户权限时，将用户编号和权限模块中的全部信息添加到用户权限表中。权限模块表的实体 E-R 图如图 3-9 所示。

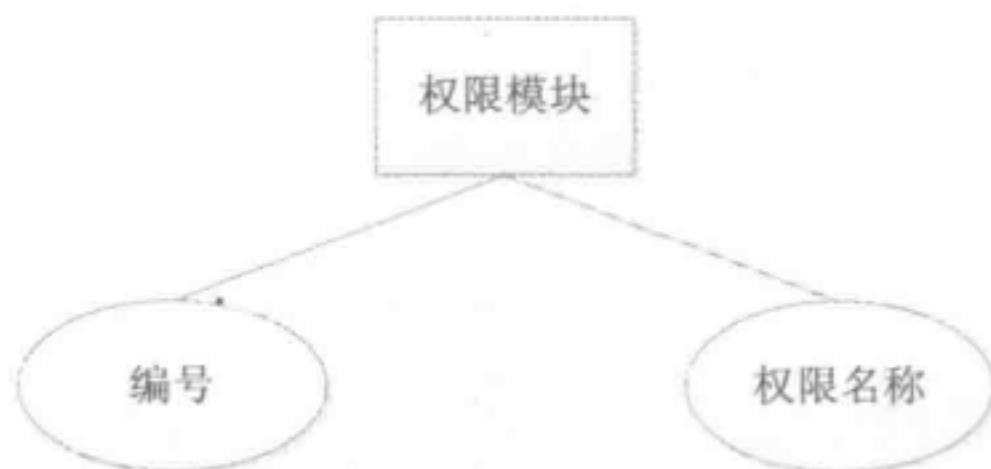


图 3-9 权限模块表的实体 E-R 图

## (2) 用户基础信息数据库的设计

在开发企业人事管理系统时，最重要的数据表是职工基本信息表，它记录了企业中所有职工的基本信息，因为该表中所涉及的字段信息很多，为了使前台在录入信息时更加简单、快捷，可以将基本表中的一些特定字段值在数据库中以表的形式进行记录。

例如，职工基本信息表中的职工类别、部门类别、文化程度等，它们的表结构都是编号+名称。首先，介绍职工基本信息中的部门类别信息，现代企业中，有很多部门分门别类，为了方便对各类信息的管理，部门的分类是必不可少的。这里简略地对部门进行管理。部门类别实体 E-R 图如图 3-10 所示。

文化程度能够在一定程度上反映个体的素质，是给他人的第一印象，文化程度是表示一个国家、一个民族人口素质的重要指标，它标志着一个国家的文化教育普及和发展程度。文化程度实体 E-R 图如图 3-11 所示。

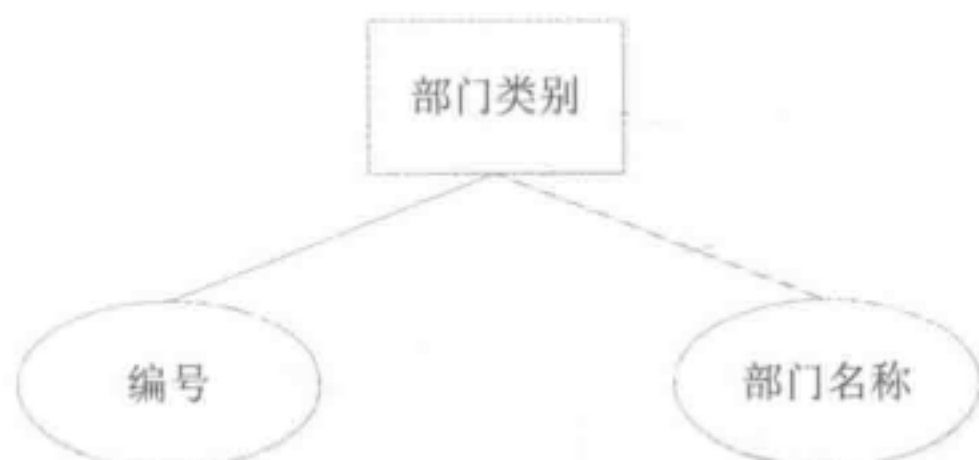


图 3-10 部门类别的实体 E-R 图

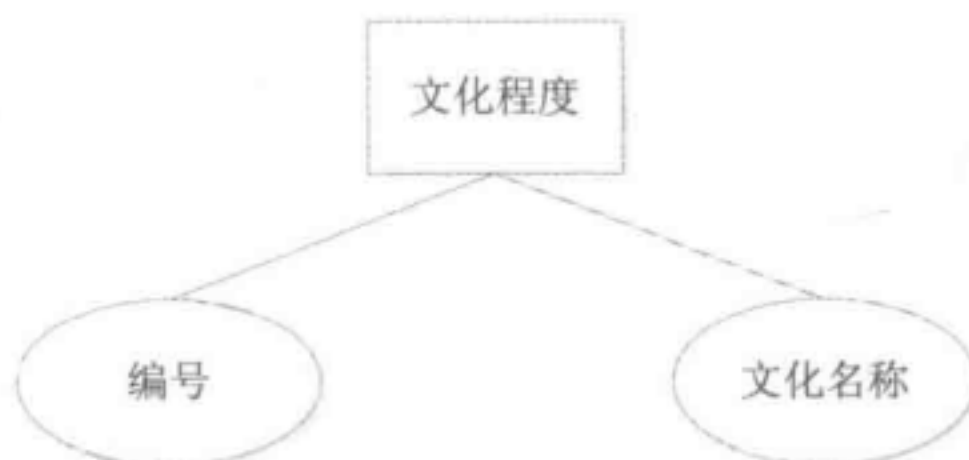


图 3-11 文化程度的实体 E-R 图

### (3) 人事管理模块数据库的设计

为了能够更好地了解职工基本信息表与其他表之间的关系，这里特地给出数据表关系图，如图 3-12 所示，通过该图可以看出，职工基本信息表的一些字段，可以在相关联的表中获取指定的值，并通过职工基本信息表的 ID 值，与家庭关系表、培训记录表、奖惩表建立相应的关系。



图 3-12 职工基本信息的实体 E-R 图

为了更具体地记录职工信息，可以创建一个家庭关系表，来记录每个职工的家庭成员以及工作单位，联系方式等。家庭关系表的实体图如图 3-13 所示。



图 3-13 家庭关系表的实体 E-R 图



应当给新员工或现有员工传授其完成本职工作所必需的正确思维认知、基本知识和技能的过程，通过提高员工工作绩效而提高企业效率，促进企业员工个人全面发展与企业可持续发展。创建一个培训记录表，其实体图如图 3-14 所示。



图 3-14 培训记录表的实体 E-R 图

#### (4) 备忘记录模块数据库设计

现代企业中，需要开各种各样的会议，为了能够详细地记录企业各种事务，所以创建一个日常记事表。日常记事表的实体 E-R 图如图 3-15 所示。

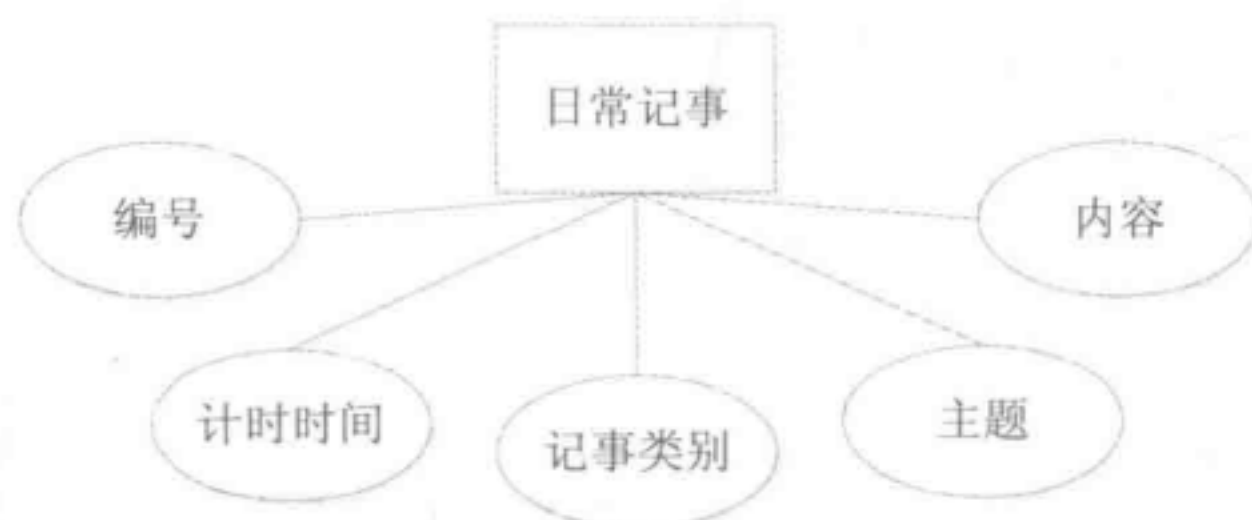


图 3-15 日常记事表的实体 E-R 图

人类的通信史依旧在不断地进化。从两个罐头加一根绳子开始，人类就在探索如何利用工具进行远端通信，直至发展到电报、电话、拨号盘电话、按键电话、手机、短信通信。通信实体 E-R 图如图 3-16 所示。



图 3-16 通信的实体 E-R 图

(5) 企业人事管理系统的数据表关系

为了更好地理解登录表与用户权限表、权限模块表之间的关系，下面给出其关系图，如图 3-17 所示。可以看出，在用户登录时，可以根据用户 ID 在用户权限表中调用相关的权限，当添加用户时，可以通过权限模块表中信息，将权限名称自动添加到用户权限表中，以方便在前台中对用户的添加操作。

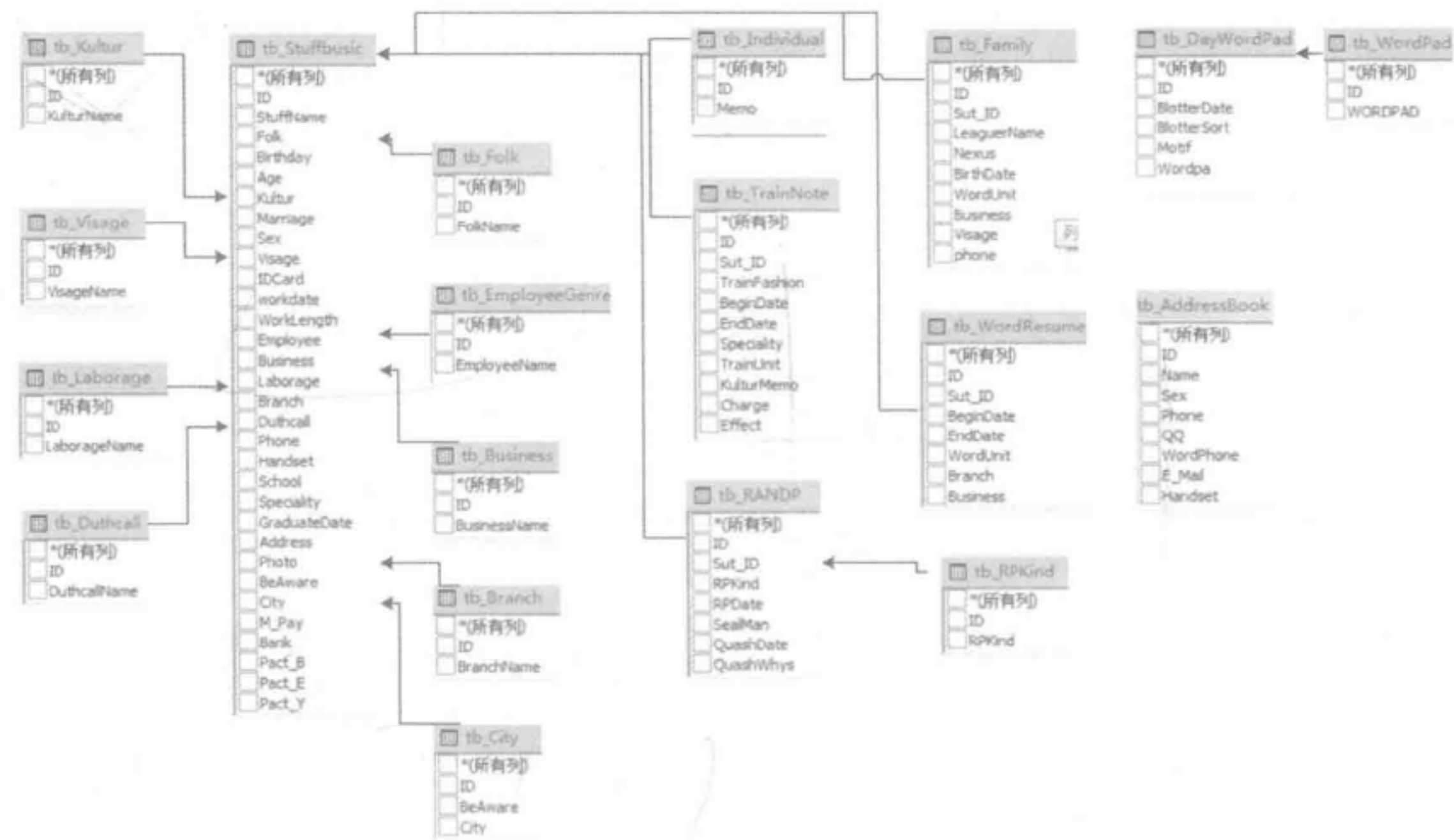


图 3-17 人事管理系统数据表的关系图

根据上面的 E-R 图，可以在数据库中创建相应的数据表，企业人事管理系统中各数据表的结构如下。

家庭关系表如表 3-4 所示。

表 3-4 家庭关系表

| 字段名         | 数据类型        | 长度 | 允许为空 |
|-------------|-------------|----|------|
| ID          | varchar(5)  | 5  | 否    |
| Sut_ID      | varchar(5)  | 5  | 是    |
| LeaguerName | varchar(20) | 20 | 是    |
| Nexus       | varchar(10) | 10 | 是    |
| BirthDate   | datetime    |    | 是    |
| WordUnit    | varchar(24) | 24 | 是    |
| Business    | varchar(10) | 10 | 是    |
| Visage      | varchar(10) | 10 | 是    |
| phone       | varchar(14) | 14 | 是    |

通信表用于存储职工的通信信息，如表 3-5 所示。

表 3-5 通信表

| 字段名       | 数据类型        | 长度 | 允许为空 |
|-----------|-------------|----|------|
| ID        | varchar(5)  | 5  | 否    |
| Name      | varchar(20) | 20 | 是    |
| Sex       | varchar(4)  | 4  | 是    |
| Phone     | varchar(13) | 13 | 是    |
| QQ        | varchar(15) | 15 | 是    |
| WordPhone | varchar(13) | 13 | 是    |
| E_Mail    | varchar(32) | 32 | 是    |
| Handset   | varchar(11) | 11 | 是    |

## 3.6 实现公共类

视频讲解  光盘：视频\第 3 章\实现公共类.avi

在开发应用程序时，可以将数据库相关操作以及对一些控件的设置、遍历等封装在自定义类中，以便于在开发程序时调用，这样，也可以提高代码的重用性。本系统创建了 MyMeans 和 MyModule 两个公共类，分别存放在 DataClass 和 ModuleClass 文件夹中，下面对这两个公共类中比较重要的自定义方法进行说明。

### 3.6.1 实现 MyMeans 公共类

类 MyMeans 封装了本系统中所有与数据库链接的方法，可以通过该类的方法与数据库建立连接，并对数据信息进行添加、修改、删除以及读取操作。在命名空间区域引用 using System.Data.SqlClient 命名空间。代码如下：

```
class MyMeans
{
    public static string Login_ID = "";
    public static string Login_Name = "";
    public static string Mean_SQL = "", Mean_Table = "", Mean_Field = "";
    public static SqlConnection My_con;
    public static string M_str_sqlcon = "Data Source=7IV5CGYJSVK2YCU;
    Database=db_PWMS;User id=sa;PWD=";
    public static int Login_n = 0;
    public static string AllSql = "Select * from tb_Stuffbusic";
}
public static SqlConnection getcon()
{
    My_con = new SqlConnection(M_str_sqlcon);
    My_con.Open();
    return My_con;
}
```

方法 getcon()是用 static 定义的静态方法，其功能就是建立于数据库的连接，用



sqlconnection 对象与指定的数据库相连接,通过 sqlconnection 对象的 open()方法打开与数据库的连接,并返回 sqlconnection 对象的信息。

方法 con\_close()的主要功能是对数据库操作后,通过该方法判断是否与数据库连接,如果连接,则关闭数据库连接具体是这样的,利用 if 语句先判断是否打开了与数据库的连接,如果是,就利用 con\_close()方法关闭连接,并释放所有的空间。代码如下:

```
public void con_close()
{
    if (My_con.State == ConnectionState.Open)
    {
        My_con.Close();
        My_con.Dispose();
    }
}
```

方法 getcom()的主要功能是用 SqlDataReader 对象以只读的方式读取数据库中的信息,并以 SqlDataReader 对象进行返回,其中 SQLstr 参数表示传递的 SQL 语句。具体是这样的,打开与数据库的连接后,创建 SqlCommand 对象,获取指定的 SQL 语句,执行 SQL 语句,生成一个 SqlDataReader 对象。代码如下:

```
public SqlDataReader getcom(string SQLstr)
{
    getcon();
    SqlCommand My_com = My_con.CreateCommand();
    My_com.CommandText = SQLstr;
    SqlDataReader My_read = My_com.ExecuteReader();
    return My_read;
}
```

方法 getaqlcom()是通过 SqlCommand 对象执行数据库中的添加、修改和删除操作,并在执行完后,关闭与数据库的连接,其中 SQLstr 参数表示传递的 SQL 语句:

```
public void getsqlcom(string SQLstr)
{
    getcon();
    SqlCommand SQLcom = new SqlCommand(SQLstr, My_con);
    SQLcom.ExecuteNonQuery();
    SQLcom.Dispose();
    con_close();
}
```

方法 getDataSet()的主要功能是创建 DataSet 对象后执行数据库中的添加、修改和删除的操作,并在执行完后,关闭与数据库的连接。代码如下:

```
public DataSet getDataSet(string SQLstr, string tableName)
{
    getcon();
    SqlDataAdapter SQLda = new SqlDataAdapter(SQLstr, My_con);
    DataSet My_DataSet = new DataSet();
    SQLda.Fill(My_DataSet, tableName);
    con_close();
    return My_DataSet;
}
```

在本系统的数据处理功能中,使用数据集提高了查询效率。在 C#程序中,用 ADO.NET 的任何软件解决方案的中心都是数据集。数据集是内存中的数据库数据的副本。数据集存

在于内存中，没有到包含相应表格或视图的数据库的活动的连接。这种断开的体系结构在读写数据库时，只使用数据库服务器资源，从而具有更大的可伸缩性。

运行时，数据从数据库传递给中间层商务对象，然后将其继续传递给用户界面。为了将数据从一层传送给另一层，ADO.NET 解决方案以 XML 格式表示内存数据(数据集)，然后将 XML 发送给另一个组件。

### 3.6.2 实现 MyModule 公共类

类 MyModule 将系统中所有窗体的动态调用，以及动态生成添加、修改、删除和查询的 SQL 语句等功能全部封装到了指定的自定义方法中，以便在开发程序时，进行重复调用。这样，就可以大大简化程序的开发过程。

因为该类中应用了可视化组件的基类和对数据库进行操作的相关对象，所以在命名空间区域引用 using.System.Windows.Forms 和 using.System.Data.SqlClient 命名空间。

主要代码如下：

```
namespace PWMS.ModuleClass
{
    class MyModule
    {
        DataClass.MyMeans MyDataClass = new PWMS.DataClass.MyMeans();
        public static string ADDs = "";
        public static string FindValue = "";
        public static string Address_ID = "";
        public static string User_ID = "";
        public static string User_Name = "";
    }
}
```

#### (1) Show\_Form()方法

Show\_Form()方法通过 FrmName 参数传递的窗体名称，调用相应的子窗体，因本系统中存在公共窗体，也就是在同一个窗体模块中，可以显示不同的窗体，所以用参数 n 来进行标识。调用公共窗体，实际上就是通过不同的 SQL 语句，在显示窗体时以不同的数据进行显示。Show\_Form()方法的具体实现代码如下所示：

```
public void Show_Form(string FrmName, int n)
{
    if (n == 1)
    {
        if (FrmName == "人事档案浏览") //判断当前要打开的窗体
        {
            PerForm.F_ManFile FrmManFile = new PWMS.PerForm.F_ManFile();
            FrmManFile.Text = "人事档案浏览"; //设置窗体名称
            FrmManFile.ShowDialog(); //显示窗体
            FrmManFile.Dispose();
        }
        if (FrmName == "人事资料查询")
        {
            PerForm.F_Find FrmFind = new PWMS.PerForm.F_Find();
            FrmFind.Text = "人事资料查询";
            FrmFind.ShowDialog();
            FrmFind.Dispose();
        }
        if (FrmName == "人事资料统计")
        {

```

```

PerForm.F_Stat FrmStat = new PWMS.PerForm.F_Stat();
FrmStat.Text = "人事资料统计";
FrmStat.ShowDialog();
FrmStat.Dispose();
}
if (FrmName == "员工生日提示")
{
    InfoAddForm.F_ClewSet FrmClewSet = new PWMS.InfoAddForm.F_ClewSet();
    FrmClewSet.Text = "员工生日提示"; //设置窗体名称
    FrmClewSet.Tag = 1; //设置窗体的 Tag 属性, 用于在打开窗体时判断窗体的显示类型
    FrmClewSet.ShowDialog(); //显示窗体
    FrmClewSet.Dispose();
}
if (FrmName == "员工合同提示")
{
    InfoAddForm.F_ClewSet FrmClewSet = new PWMS.InfoAddForm.F_ClewSet();
    FrmClewSet.Text = "员工合同提示";
    FrmClewSet.Tag = 2;
    FrmClewSet.ShowDialog();
    FrmClewSet.Dispose();
}
if (FrmName == "日常记事")
{
    PerForm.F_WordPad FrmWordPad = new PWMS.PerForm.F_WordPad();
    FrmWordPad.Text = "日常记事";
    FrmWordPad.ShowDialog();
    FrmWordPad.Dispose();
}
if (FrmName == "通信录")
{
    PerForm.F_AddressList FrmAddressList = new PWMS.PerForm.F_AddressList();
    FrmAddressList.Text = "通信录";
    FrmAddressList.ShowDialog();
    FrmAddressList.Dispose();
}
if (FrmName == "备份/还原数据库")
{
    PerForm.F_HaveBack FrmHaveBack = new PWMS.PerForm.F_HaveBack();
    FrmHaveBack.Text = "备份/还原数据库";
    FrmHaveBack.ShowDialog();
    FrmHaveBack.Dispose();
}
if (FrmName == "清空数据库")
{
    PerForm.F_ClearData FrmClearData = new PWMS.PerForm.F_ClearData();
    FrmClearData.Text = "清空数据库";
    FrmClearData.ShowDialog();
    FrmClearData.Dispose();
}

if (FrmName == "重新登录")
{
    F_Login FrmLogin = new F_Login();
    FrmLogin.Tag = 2;
    FrmLogin.ShowDialog();
    FrmLogin.Dispose();
}
if (FrmName == "用户设置")
{
    PerForm.F_User FrmUser = new PWMS.PerForm.F_User();

```



```
        FrmUser.Text = "用户设置";
        FrmUser.ShowDialog();
        FrmUser.Dispose();
    }
    if (FrmName == "计算器")
    {
        System.Diagnostics.Process.Start("calc.exe");
    }
    if (FrmName == "记事本")
    {
        System.Diagnostics.Process.Start("notepad.exe");
    }
}
if (n == 2)
{
    String FrmStr = ""; //记录窗体名称
    if (FrmName == "民族类别设置") //判断要打开的窗体
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Folk"; //SQL 语句
        DataClass.MyMeans.Mean_Table = "tb_Folk"; //表名
        DataClass.MyMeans.Mean_Field = "FolkName"; //添加、修改数据的字段名
        FrmStr = FrmName;
    }
    if (FrmName == "职工类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_EmployeeGenre";
        DataClass.MyMeans.Mean_Table = "tb_EmployeeGenre";
        DataClass.MyMeans.Mean_Field = "EmployeeName";
        FrmStr = FrmName;
    }
    if (FrmName == "文化程度设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Kultur";
        DataClass.MyMeans.Mean_Table = "tb_Kultur";
        DataClass.MyMeans.Mean_Field = "KulturName";
        FrmStr = FrmName;
    }
    if (FrmName == "政治面貌设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Visage";
        DataClass.MyMeans.Mean_Table = "tb_Visage";
        DataClass.MyMeans.Mean_Field = "VisageName";
        FrmStr = FrmName;
    }
    if (FrmName == "部门类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Branch";
        DataClass.MyMeans.Mean_Table = "tb_Branch";
        DataClass.MyMeans.Mean_Field = "BranchName";
        FrmStr = FrmName;
    }
    if (FrmName == "工资类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Laborage";
        DataClass.MyMeans.Mean_Table = "tb_Laborage";
        DataClass.MyMeans.Mean_Field = "LaborageName";
        FrmStr = FrmName;
    }
    if (FrmName == "职务类别设置")
    {

```

```

        DataClass.MyMeans.Mean_SQL = "select * from tb_Business";
        DataClass.MyMeans.Mean_Table = "tb_Business";
        DataClass.MyMeans.Mean_Field = "BusinessName";
        FrmStr = FrmName;
    }
    if (FrmName == "职称类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Duthcall";
        DataClass.MyMeans.Mean_Table = "tb_Duthcall";
        DataClass.MyMeans.Mean_Field = "DuthcallName";
        FrmStr = FrmName;
    }
    if (FrmName == "奖惩类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_RPKind";
        DataClass.MyMeans.Mean_Table = "tb_RPKind";
        DataClass.MyMeans.Mean_Field = "RPKind";
        FrmStr = FrmName;
    }
    if (FrmName == "记事本类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_WordPad";
        DataClass.MyMeans.Mean_Table = "tb_WordPad";
        DataClass.MyMeans.Mean_Field = "WordPad";
        FrmStr = FrmName;
    }
    InfoAddForm.F_Basic FrmBasic = new PWMS.InfoAddForm.F_Basic();
    FrmBasic.Text = FrmStr; //设置窗体的名称
    FrmBasic.ShowDialog(); //显示调用的窗体
    FrmBasic.Dispose();
}
#endregion

```

## (2) GetMenu()方法

GetMenu()方法的主要功能是将 MenuStrip 菜单中的菜单项按照级别动态添加到 TreeView 控件的相应节点中。其中 treeV 参数表示要添加节点的 TreeView 控件, MenuS 参数表示要获取信息的 MenuStrip 菜单。GetMenu()方法的具体实现代码如下所示:

```

public void GetMenu(TreeView treeV, MenuStrip MenuS)
{
    for (int i=0; i<MenuS.Items.Count; i++) //遍历 MenuStrip 组件中的一级菜单项
    {
        //将一级菜单项的名称添加到 TreeView 组件的根节点中, 并设置当前节点的子节点 newNode1
        TreeNode newNode1 = treeV.Nodes.Add(MenuS.Items[i].Text);
        //将当前菜单项的所有相关信息存入到 ToolStripDropDownItem 对象中
        ToolStripDropDownItem newmenu = (ToolStripDropDownItem)MenuS.Items[i];
        //判断当前菜单项中是否有二级菜单项
        if (newmenu.HasDropDownItems && newmenu.DropDownItems.Count > 0)
            for (int j=0; j<newmenu.DropDownItems.Count; j++) //遍历二级菜单项
            {
                //将二级菜单名称添加到 TreeView 组件的子节点 newNode1 中,
                //并设置当前节点的子节点 newNode2
                TreeNode newNode2 = newNode1.Nodes.Add(newmenu.DropDownItems[j].Text);
                //将当前菜单项的所有相关信息存入到 ToolStripDropDownItem 对象中
                ToolStripDropDownItem newmenu2 =
                    (ToolStripDropDownItem)newmenu.DropDownItems[j];
                //判断二级菜单项中是否有三级菜单项
                if (newmenu2.HasDropDownItems && newmenu2.DropDownItems.Count > 0)

```

```
        for (int p=0; p<newmenu2.DropDownItems.Count; p++) //遍历三级菜单项
            //将三级菜单名称添加到 TreeView 组件的子节点 newNode2 中
            newNode2.Nodes.Add(newmenu2.DropDownItems[p].Text);
    }
}
```

### (3) Clear\_Control()方法

Clear\_Control()方法的主要功能是清空可视化控件集中指定控件的文本信息及图片，主要用于在添加数据信息时，对相应文本框进行清空。其中 Con 参数表示可视化控件的控件集合。Clear\_Control()方法的具体实现代码如下所示：

```
public void Clear_Control(Control.ControlCollection Con)
{
    foreach (Control C in Con) { //遍历可视化组件中的所有控件
        if (C.GetType().Name == "TextBox") //判断是否为 TextBox 控件
            if (((TextBox)C).Visible == true) //判断当前控件是否为显示状态
                ((TextBox)C).Clear(); //清空当前控件
        if (C.GetType().Name == "MaskedTextBox") //判断是否为 MaskedTextBox 控件
            if (((MaskedTextBox)C).Visible == true) //判断当前控件是否为显示状态
                ((MaskedTextBox)C).Clear(); //清空当前控件
        if (C.GetType().Name == "ComboBox") //判断是否为 ComboBox 控件
            if (((ComboBox)C).Visible == true) //判断当前控件是否为显示状态
                ((ComboBox)C).Text = ""; //清空当前控件的 Text 属性值
        if (C.GetType().Name == "PictureBox") //判断是否为 PictureBox 控件
            if (((PictureBox)C).Visible == true) //判断当前控件是否为显示状态
                ((PictureBox)C).Image = null; //清空当前控件的 Image 属性
    }
}
```

### (4) Find\_Grids()方法

Find\_Grids()方法的主要功能是，查找指定可视化控件集中控件名包含 TName 参数值的所用控件，并根据控件名称，获取相应表的字段名，当查找的控件为 TextBox 时，根据当时控件的部分名称查找相应的 ComboBox 控件(用来记录逻辑运算符)，通过 ANDSign 参数将具有相关性的控件组合成查询条件，存入到公共变量 FindValue 中。Find\_Grids()方法的具体实现代码如下所示：

```
public void Find_Grids(Control.ControlCollection GBox, string TName, string ANDSign)
{
    string sID = ""; //定义局部变量
    if (FindValue.Length>0)
        FindValue = FindValue + ANDSign;
    foreach (Control C in GBox) { //遍历控件集上的所有控件
        //判断是否是要遍历的控件
        if (C.GetType().Name == "TextBox" || C.GetType().Name == "ComboBox") {
            if (C.GetType().Name == "ComboBox" && C.Text!="") { //当指定控件不为空时
                sID = C.Name;
                if (sID.IndexOf(TName) > -1) { //当 TName 参数是当前控件名中的部分信息时
                    //用“_”符号分隔当前控件的名称，获取相应的字段名
                    string[] Astr = sID.Split(Convert.ToChar('_'));
                    //生成查询条件
                    FindValue = FindValue + "(" + Astr[1] + " = '" + C.Text + "'" + ANDSign;
                }
            }
            //如果当前为 TextBox 控件，并且控件不为空
            if (C.GetType().Name == "TextBox" && C.Text != "")

```



```

{
    sID = C.Name; //获取当前控件的名称
    if (sID.IndexOf(TName) > -1) //判断 TName 参数值是否为当前控件名的子字符串
    {
        //以“_”为分隔符将控件名存入到一维数组中
        string[] Astr = sID.Split(Convert.ToChar('_'));
        string m_Sgin = ""; //用于记录逻辑运算符
        string mID = ""; //用于记录字段名
        if (Astr.Length > 2) //当数组的元素个数大于2时
            mID = Astr[1] + "_" + Astr[2]; //将最后两个元素组成字段名
        else
            mID = Astr[1]; //获取当前条件所对应的字段名称
        foreach (Control Cl in GBox) //遍历控件集
        {
            if (Cl.GetType().Name == "ComboBox") //判断是否为 ComboBox 组件
                //判断当前组件名是否包含条件组件的部分文件名
                if ((Cl.Name).IndexOf(mID) > -1)
                {
                    if (Cl.Text == "") //当查询条件为空时
                        break; //退出本次循环
                    else
                    {
                        m_Sgin = Cl.Text; //将条件值存储到 m_Sgin 变量中
                        break;
                    }
                }
            }
            if (m_Sgin != "") //当该条件不为空时
                //组合 SQL 语句的查询条件
                FindValue = FindValue + "(" + mID + m_Sgin + C.Text + ")" + ANDSign;
        }
    }
}
if (FindValue.Length > 0) //当存储查询条件的变量不为空时,删除逻辑运算符 AND 和 OR
{
    if (FindValue.IndexOf("AND") > -1) //判断是否用 AND 连接条件
        FindValue = FindValue.Substring(0, FindValue.Length - 4);
    if (FindValue.IndexOf("OR") > -1) //判断是否用 OR 连接条件
        FindValue = FindValue.Substring(0, FindValue.Length - 3);
}
else
    FindValue = "";
}

```

### (5) GetAutocoding()方法

GetAutocoding()方法的主要功能是在添加数据时,自动获取添加数据的编号。其实现过程是通过表名和 ID 字段在表中查找最大的 ID 值,并将 ID 值加 1 进行返回,当表中无记录时,返回“0001”。TableName 参数表示进行自动编号的表名,ID 参数表示数据表的编号字段。GetAutocoding()方法的具体实现代码如下所示:

```

public String GetAutocoding(string TableName, string ID)
{
    //查找指定表中 ID 号为最大的记录
    SqlDataReader MyDR = MyDataClass.getcom("select max(" + ID + ") NID from " + TableName);
    int Num = 0;
    if (MyDR.HasRows) //当查找到记录时
    {

```

```

MyDR.Read(); //读取当前记录
if (MyDR[0].ToString() == "")
    return "0001";
Num = Convert.ToInt32(MyDR[0].ToString()); //将当前找到的最大编号转换成整数
++Num; //最大编号加1
string s = string.Format("{0:0000}", Num); //将整数值转换成指定格式的字符串
return s; //返回自动生成的编号
}
else
{
    return "0001"; //当数据表没有记录时, 返回 0001
}
}

```

### (6) TreeMenuF()方法

TreeMenuF()方法是在单击 TreeView 控件的节点时被调用的,其主要功能是通过所选节点的文本名称,在 MenuStrip 控件中进行遍历查找,如果找到,并且为可用状态,则通过 show\_form()方法动态调用相关的窗体。TreeMenuF()方法的具体实现代码如下所示:

```

public void TreeMenuF(MenuStrip MenuS, TreeNodeMouseClickEventArgs e)
{
    string Men = "";
    for (int i=0; i<MenuS.Items.Count; i++) //遍历 MenuStrip 控件中的主菜单项
    {
        Men = ((ToolStripDropDownItem)MenuS.Items[i]).Name; //获取主菜单项的名称
        if (Men.IndexOf("Menu") == -1) //如果 MenuStrip 控件的菜单项没有子菜单
        {
            //当节点名称与菜单项名称相等时
            if (((ToolStripDropDownItem)MenuS.Items[i]).Text == e.Node.Text)
                //判断当前菜单项是否可用
                if (((ToolStripDropDownItem)MenuS.Items[i]).Enabled == false)
                {
                    MessageBox.Show(
                        "当前用户无权限调用" + "\"\" + e.Node.Text + "\"\" + "窗体");
                    break;
                }
            else
                //调用相应的窗体
                Show_Form(((ToolStripDropDownItem)MenuS.Items[i]).Text.Trim(), 1);
        }
        ToolStripDropDownItem newmenu = (ToolStripDropDownItem)MenuS.Items[i];
        //遍历二级菜单项
        if (newmenu.HasDropDownItems && newmenu.DropDownItems.Count > 0)
            for (int j=0; j<newmenu.DropDownItems.Count; j++)
            {
                Men = newmenu.DropDownItems[j].Name; //获取二级菜单项的名称
                if (Men.IndexOf("Menu") == -1)
                {
                    if ((newmenu.DropDownItems[j]).Text == e.Node.Text)
                        if ((newmenu.DropDownItems[j]).Enabled == false)
                        {
                            MessageBox.Show("当前用户无权限调用"
                                + "\"\" + e.Node.Text + "\"\" + "窗体");
                            break;
                        }
                    else
                        Show_Form((newmenu.DropDownItems[j]).Text.Trim(), 1);
                }
            }
        ToolStripDropDownItem newmenu2 =

```

```

        (ToolStripDropDownItem)newmenu.DropDownItems[j];
        //遍历三级菜单项
        if (newmenu2.HasDropDownItems && newmenu2.DropDownItems.Count > 0)
            for (int p=0; p<newmenu2.DropDownItems.Count; p++)
            {
                if ((newmenu2.DropDownItems[p]).Text == e.Node.Text)
                {
                    if ((newmenu2.DropDownItems[p]).Enabled == false)
                    {
                        MessageBox.Show("当前用户无权限调用"
                            + "\"\" + e.Node.Text + "\"\" + "窗体");
                        break;
                    }
                }
                else
                {
                    if ((newmenu2.DropDownItems[p]).Text.Trim()
                        == "员工生日提示"
                        || (newmenu2.DropDownItems[p]).Text.Trim()
                        == "员工合同提示")
                        Show_Form((newmenu2.DropDownItems[p]).Text.Trim(), 1);
                    else
                        Show_Form((newmenu2.DropDownItems[p]).Text.Trim(), 2);
                }
            }
        }
    }
}

```

在上述代码中用到了 TreeView 控件，TreeView 控件是树视图控件，功能是用于显示节点的层次结构效果，并且能够在各节点内显示对应的子节点。用户可以扩展开或以折叠的方式将子节点信息显示出来。

TreeView 控件主要通过如下两个属性实现树形功能。

- Nodes: 包含树视图中的顶级节点列表，并且包含了 TreeNode 对象的集合，每个对象都具有一个 Nodes 属性，每个属性都可以包含自己的 TreeNodeCollection。
- SelectedNode: 设置当前选中的节点。

树视图中的每个对象都具有可用于定位树视图的属性，例如 FirstNode、LastNode、NextNode、PrevNode 和 Parent。

如果某个节点有子节点，则将子节点放到它的 Nodes 属性中。TreeView 控件本身具有 TopNode 属性，此属性是整个树视图的根节点。使用递归方法可以访问树视图中的每个节点，例如下面的代码使用了递归访问：

```

private void AccessNodeRecursive(TreeNode treeNode, List<TreeNode> nodes)
{
    // 处理每一个节点
    if (treeNode.Checked)
        nodes.Add(treeNode);
    // 对每一个子节点递归
    foreach (TreeNode tn in treeNode.Nodes)
    {
        AccessNodeRecursive(tn, nodes);
    }
}

```

在 TreeView 控件内，可以显示指定的图像，首先将 ImageList 分配给 TreeView 的 ImageList 属性，然后通过引用 Image 在 ImageList 中的位置来分配该 Image。可以使用如下



两个属性来实现图像的分配。

- **ImageIndex**: 设置当树节点未被选定时所显示的 Image 的索引值。
- **SelectedImageIndex**: 设置当树节点被选定时所显示的 Image 的索引值。

#### (7) Show\_Pope()方法

Show\_Pope()方法的主要功能是通过当前登录用户的名称,在权限用户表中查询当前用户的所有权限,并根据权限设置菜单栏中各菜单项的状态,其中 MenuS 参数是要设置的菜单栏控件,UName 参数为当前用户的名称。代码如下:

```
public void Show_Pope(Control.ControlCollection GBox, string TID)
{
    string sID = "";
    string CheckName = "";
    bool t = false;
    DataSet DSet = MyDataClass.getDataSet(
        "select ID,PopeName,Pope from tb_UserPope where ID='" + TID + "'", "tb_UserPope");
    for (int i=0; i<DSet.Tables[0].Rows.Count; i++)
    {
        sID = Convert.ToString(DSet.Tables[0].Rows[i][1]);
        if ((int)(DSet.Tables[0].Rows[i][2]) == 1)
            t = true;
        else
            t = false;
        foreach (Control C in GBox)
        {
            if (C.GetType().Name == "CheckBox")
            {
                CheckName = C.Name;
                if (CheckName.IndexOf(sID) > -1)
                {
                    ((CheckBox)C).Checked = t;
                }
            }
        }
    }
}
```

## 3.7 实现用户登录模块

视频讲解  光盘: 视频\第3章\实现用户登录模块.avi

在本系统中,登录模块主要是通过输入正确的用户名和密码进入主窗体,这样可以提高程序的安全性,保护数据资料不外泄。

### 3.7.1 登录模块技术分析

登录窗体使用了 `sqldatareader` 对象从数据源中检索只读数据集,该对象只允许以只读、向前的方式查看其中所存储的数据。

可以用该对象的 `GetString(n)`、`GetInt32(n)`、`GetDataTime(n)` 等方法读取指定字段的值,其中, `n` 表示当前表中字段的列数。

3.7.2 具体实现

本系统登录模块的具体实现步骤如下。

(1) 新建一个 Windows 窗体，命名为 F\_Login.cs，主要用于实现系统的登录功能。用到的控件如表 3-6 所示。

表 3-6 登录窗体的控件

| 控件类型    | 控件 ID    | 主要属性设置              | 用 途      |
|---------|----------|---------------------|----------|
| TextBox | textName | 无                   | 输入登录用户名  |
|         | textPass | PasswordChar 属性设置为* | 输入登录用户密码 |
| Button  | butLogin | Text 属性设置为“登录”      | 登录       |
|         | butClose | Text 属性设置为“取消”      | 取消       |

(2) 在登录窗体加载时，首先要用 DataClass 文件夹下的 MyMeans 类中的自定义方法 con\_open()来连接数据库，当数据库连接失败时，弹出提示信息，并关闭整个工程；否则，显示登录窗体，进行登录。

(3) 当用户输入用户名和密码后，单击“登录”按钮进行登录。在“登录”按钮的 Click 事件中，首先判断用户名和密码是否为空，如果为空，则弹出提示框，提示用户将登录信息填写完整，否则，将判断用户名和密码是否正确，如果正确，则进入系统。

(4) 由于系统的登录窗体与重新登录窗体调用的是同一个窗体，所以在单击“取消”按钮时，要通过该窗体的 Tag 属性值进行判断，如果当前是登录窗体，则关闭整个工程，否则只关闭当前窗体。

文件 F\_Login.cs 的主要实现代码如下所示：

```
public partial class F_Login : Form
{
    DataClass.MyMeans MyClass = new PWMS.DataClass.MyMeans();

    public F_Login()
    {
        InitializeComponent();
    }
    private void butClose_Click(object sender, EventArgs e)
    {
        if ((int)(this.Tag) == 1)
        {
            DataClass.MyMeans.Login_n = 3;
            Application.Exit();
        }
        else
        {
            if ((int)(this.Tag) == 2)
                this.Close();
        }
    }
    private void butLogin_Click(object sender, EventArgs e)
    {
        if (textName.Text != "" & textPass.Text != "")
        {
            SqlDataReader temDR = MyClass.getcom("select * from tb_Login where Name='"
                + textName.Text.Trim() + "' and Pass='" + textPass.Text.Trim() + "'");
```

```

        bool ifcom = temDR.Read();
        if (ifcom)
        {
            DataClass.MyMeans.Login_Name = textName.Text.Trim();
            DataClass.MyMeans.Login_ID = temDR.GetString(0);
            DataClass.MyMeans.My_con.Close();
            DataClass.MyMeans.My_con.Dispose();
            DataClass.MyMeans.Login_n = (int)(this.Tag);
            this.Close();
        }
        else
        {
            MessageBox.Show("用户名或密码错误!", "提示", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            textName.Text = "";
            textPass.Text = "";
        }
        MyClass.con_close();
    }
    else
    {
        MessageBox.Show("请将登录信息填写完整!", "提示", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    private void F_Login_Load(object sender, EventArgs e)
    {
        try
        {
            MyClass.con_open(); //连接数据库
            MyClass.con_close();
            textName.Text = "";
            textPass.Text = "";
        }
        catch
        {
            MessageBox.Show("数据库连接失败。", "提示", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            Application.Exit();
        }
    }
}

```

本系统登录窗体的执行效果如图 3-18 所示。



图 3-18 登录窗体的执行效果

## 3.8 主窗体详细设计

视频讲解 光盘：视频\第 3 章\主窗体详细设计.avi

主窗体是程序操作过程中必不可少的，是人机交互中的重要环节。通过主窗体，用户



可以调用系统相关的各子模块，快速掌握本系统的实现功能及操作方法，还可以通过主窗体的菜单栏，判断当前用户对各子模块的使用权限。

当登录窗体验证成功后，用户将进入主窗体，主窗体可分为 4 个部分：最上面是系统的菜单栏，可以通过它调用系统中的所有子窗体；菜单栏下面是常用的按钮区，以便于用户的操作；在窗体的左边是一个树形下拉列表，该列表的各节点与菜单栏相同，可以通过属性列表完整地显示该系统的所有子窗体及调用；在窗体的最下面，用状态栏显示当前登录的用户名。

本章案例的主窗体界面效果如图 3-19 所示。

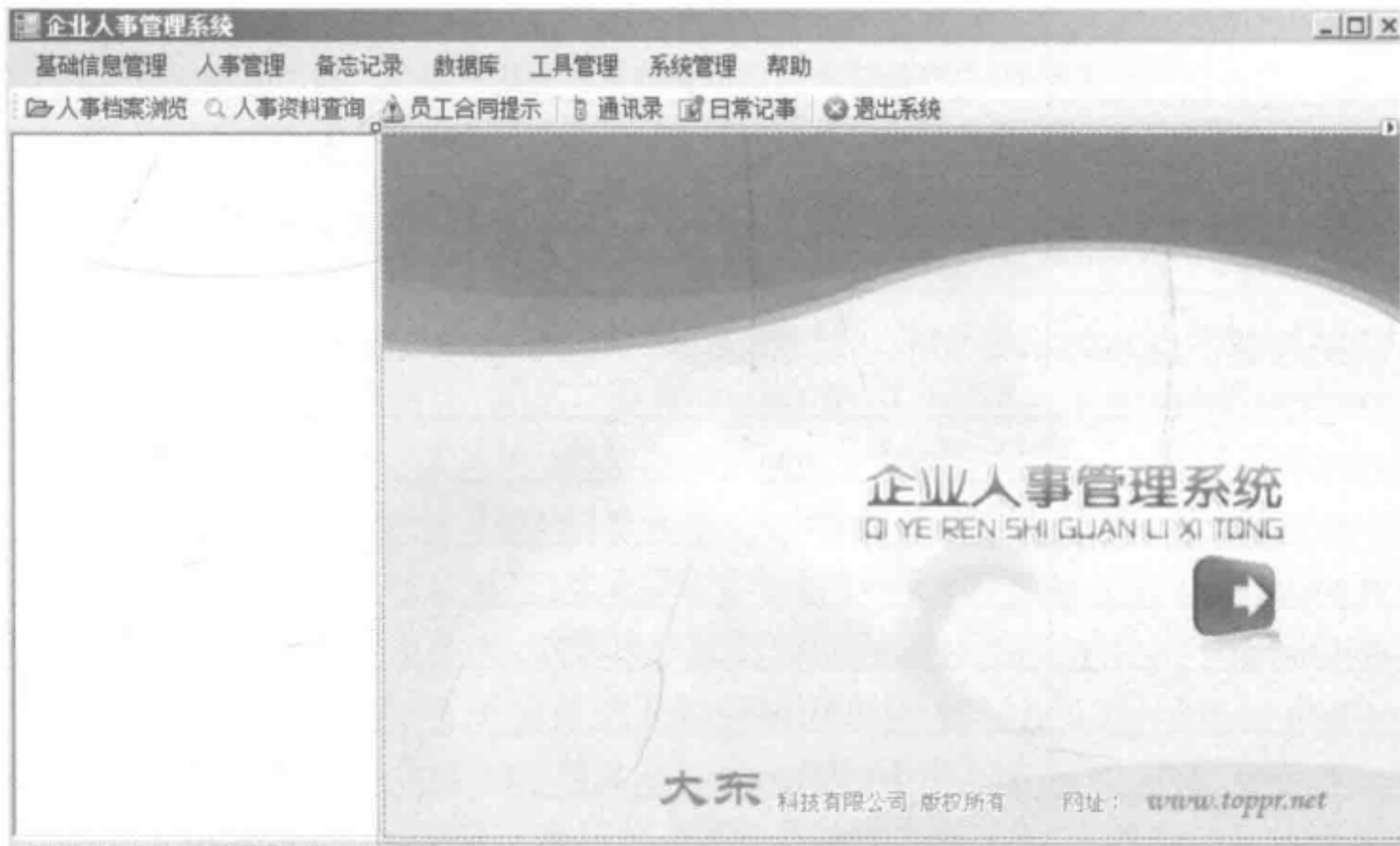


图 3-19 系统主窗体界面的执行效果

### 3.8.1 主窗体技术分析

当用户以普通用户身份进入主窗体时，主窗体中的菜单栏将根据当前用户的使用权限，对各类子菜单栏的使用状态进行相应的设置，当用属性类表调用子窗体时，如果有权限，则显示相应的子窗体，否则将弹出“当前用户无权限调用 XXX 窗体”对话框。

本窗体使用了 MenuStrip 控件的 ToolStripDropDownItem 对象和 TreeView 控件的 TreeNode 对象，使 MenuStrip 控件中的菜单项按照指定的级别动态添加到 TreeView 控件中。

ToolStripDropDownItem 对象用于存储各菜单项下的所有信息，并通过该对象的 DropDownItems 属性取得各子菜单项的名称，然后通过 TreeNode 对象 Nodes 属性的 Add() 方法将 MenuStrip 控件的菜单项添加到 TreeView 控件中。

注意 ToolStripDropDownItem，这是 ToolStripMenuItem、ToolStripDropDownButton 和 ToolStripSplitButton 的抽象基类，它可以直接承载项或将其他项承载在下拉容器中。

可以通过将 DropDown 属性设置为 ToolStripDropDown 以及设置 ToolStripDropDown 的 Items 属性来执行此操作。可直接通过 DropDownItems 属性访问这些下拉项。

ToolStripDropDownItem 包含了如表 3-7 所示的事件。

表 3-7 ToolStripDropDownItem 包含的事件

| 名 称                    | 说 明                                    |
|------------------------|----------------------------------------|
| AvailableChanged       | 当 Available 属性的值更改时发生                  |
| BackColorChanged       | 当 BackColor 属性的值更改时发生                  |
| Click                  | 在单击 ToolStripItem 时发生                  |
| DisplayStyleChanged    | 在 DisplayStyle 更改后发生                   |
| Disposed               | 当通过调用 Dispose 方法释放组件时发生                |
| DoubleClick            | 当用鼠标双击项时发生                             |
| DragDrop               | 当用户拖动某项后再释放鼠标按钮时发生，指示此项应该被放入该项内        |
| DragEnter              | 当用户将某项拖动到该项的工作区内时发生                    |
| DragLeave              | 当用户拖动某项并且鼠标指针不再悬停在此项的工作区上方时发生          |
| DragOver               | 当用户将某项拖动到此项的工作区上方时发生                   |
| DropDownClosed         | 当 ToolStripDropDown 关闭时发生              |
| DropDownItemClicked    | 在单击 ToolStripDropDown 时发生              |
| DropDownOpened         | 当 ToolStripDropDown 已经打开时发生            |
| DropDownOpening        | 当 ToolStripDropDown 正在打开时发生            |
| EnabledChanged         | 在 Enabled 属性值更改后发生                     |
| ForeColorChanged       | 在 ForeColor 属性值更改时发生                   |
| GiveFeedback           | 在执行拖动操作期间发生                            |
| LocationChanged        | 当更新 ToolStripItem 的位置时发生               |
| MouseDown              | 当鼠标指针放置在该项上时，在按鼠标按钮时发生                 |
| MouseEnter             | 在鼠标指针进入项时发生                            |
| MouseHover             | 当鼠标指针悬停在项上时发生                          |
| MouseLeave             | 当鼠标指针离开项时发生                            |
| MouseMove              | 当鼠标指针移到该项上时发生                          |
| MouseUp                | 当鼠标指针位于该项上时，在松开鼠标按钮时发生                 |
| OwnerChanged           | 当 Owner 属性更改时发生                        |
| Paint                  | 在重绘项时发生                                |
| QueryAccessibilityHelp | 当具有辅助功能的客户端应用程序调用 ToolStripItem 的帮助时发生 |
| QueryContinueDrag      | 在拖放操作期间发生，并且允许拖动源确定是否应取消拖放操作           |
| RightToLeftChanged     | 在 RightToLeft 属性值更改时发生                 |
| TextChanged            | 当 Text 属性的值更改时发生                       |
| VisibleChanged         | 当 Visible 属性的值更改时发生                    |

3.8.2 具体实现

本系统主窗体模块的具体实现步骤如下。

(1) 新建一个 Windows 应用程序，将默认创建的窗体命名为 F\_Main.cs，用于制做当前系统的主窗体，主要控件如表 3-8 所示。

表 3-8 主窗体控件

| 控件类型        | 控件 ID        | 主要属性设置                              | 用 途           |
|-------------|--------------|-------------------------------------|---------------|
| MenuStrip   | menuStrip1   | Items 中添加 7 个 MenuItem 菜单项及相应的子菜单项  | 实现系统主窗体中的菜单项  |
| ToolStrip   | toolStrip1   | Items 中添加 6 个 Button 按钮             | 实现系统主窗体中的常用按钮 |
| TreeView    | TreeView1    | 将 Dock 设为 Left                      | 以树形方式显示菜单栏    |
| StatusStrip | statusStrip1 | Items 中添加 4 个 toolStripStatusLabel1 | 实现系统的状态栏      |

(2) 在主窗体加载时，首先要调用登录窗体，当登录窗体验证成功后，判断所用的窗体是否为登录窗体或重新登录窗体，如果是，则通过自定义方法 Preen\_Main()对窗体进行初始化。

(3) 在本窗体中自定义了一个 Preen\_Main()方法，该方法用于显示当前登录用户的名称，并将菜单栏中的各项动态地添加到树形下拉列表中，根据当前用户的权限，设置菜单栏的可用状态。

(4) 为了使用户重新登录后，能够在主窗体的菜单栏中根据用户权限重新设置各菜单项的可用状态，可以在主窗体被激活时，重新根据用户权限对窗体进行初始化。

(5) 当主窗体显示后，单击菜单栏中的各菜单项，可以显示相应的子窗体。为了使程序的制作过程更加简便，将所有子窗体的调用封装到了 MyModule 公共类的 Show\_Form()方法中，只需要获取当前调用窗体的名称及标识，便可以得到相应的窗体。

(6) 可以在 treeView1 组件的节点单击事件(NodeMouseClick)中调用相应的子窗体。  
文件 F\_Main.cs 的具体实现代码如下所示：

```
public partial class F_Main : Form
{
    DataClass.MyMeans MyClass = new PWMS.DataClass.MyMeans();
    ModuleClass.MyModule MyMenu = new PWMS.ModuleClass.MyModule();
    public F_Main()
    {
        InitializeComponent();
    }
    #region 通过权限对主窗体进行初始化
    /// <summary>
    /// 对主窗体初始化.
    /// </summary>
    private void Preen_Main()
    {
        //在状态栏中显示当前登录的用户名
        statusStrip1.Items[2].Text = DataClass.MyMeans.Login_Name;
        treeView1.Nodes.Clear();
        //调用公共类 MyModule 下的 GetMenu() 方法,
        //将 menuStrip1 控件的子菜单添加到 treeView1 控件中
        MyMenu.GetMenu(treeView1, menuStrip1);
        MyMenu.MainMenuF(menuStrip1); //将菜单栏中的各子菜单项设为不可用状态
        //根据权限设置相应子菜单的可用状态
        MyMenu.MainPope(menuStrip1, DataClass.MyMeans.Login_Name);
    }
}
```



```
}
#endregion

private void F_Main_Load(object sender, EventArgs e)
{
    F_Login FrmLogin = new F_Login(); //声明登录窗体, 进行调用
    FrmLogin.Tag = 1; //将登录窗体的 Tag 属性设为 1, 表示调用的是登录窗体
    FrmLogin.ShowDialog();
    FrmLogin.Dispose();
    //当调用的是登录窗体时
    if (DataClass.MyMeans.Login_n == 1)
    {
        Preen_Main(); //自定义方法, 通过权限对窗体进行初始化
        MyMenu.PactDay(1); //MyModule 类中的自定义方法, 用于查找指定时间内过生日的职工
        MyMenu.PactDay(2); //MyModule 类中的自定义方法, 用于查找合同到期的职工
    }
    DataClass.MyMeans.Login_n = 3; //将公共变量设为 3, 便于控制登录窗体的关闭
}

private void F_Main_Activated(object sender, EventArgs e)
{
    if (DataClass.MyMeans.Login_n == 2) //当调用的是重新登录窗体时
        Preen_Main(); //自定义方法, 通过权限对窗体进行初始化
    DataClass.MyMeans.Login_n = 3;
}

private void 系统退出ToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}

public void Tool_Folk_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 2);
}

private void Tool_Stuffbusic_Click(object sender, EventArgs e)
{
    //用 MyModule 公共类中的 Show_Form() 方法调用各窗体
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_ClewBirthday_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_ClewBargain_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_Stufind_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_Stusum_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}
```

```

private void Tool_DayWordPad_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_AddressBook_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_Back_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_Clear_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_NewLogon_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void Tool_Setup_Click(object sender, EventArgs e)
{
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}

private void treeView1_NodeMouseClick(object sender, TreeNodeMouseClickEventArgs e)
{
    if (e.Node.Text.Trim() == "系统退出") //如果当前节点的文本为“系统退出”
    {
        Application.Exit(); //关闭整个工程
    }
    MyMenu.TreeMenuF(menuStrip1, e); //用MyModule 公共类中的TreeMenuF()方法调用各窗体
}

private void Button_Close_Click(object sender, EventArgs e)
{
    this.Close();
}

private void Button_Stuffbusic_Click(object sender, EventArgs e)
{
    if (Tool_Stuffbusic.Enabled==true)
        Tool_Stuffbusic_Click(sender, e);
    else
        MessageBox.Show("当前用户无权限调用" + "\"" +
            + ((ToolStripButton)sender).Text + "\"" + "窗体");
}

private void Button_Stufind_Click(object sender, EventArgs e)
{
    if (Tool_Stufind.Enabled == true)
        Tool_Stufind_Click(sender, e);
    else
        MessageBox.Show("当前用户无权限调用" + "\""

```

```

        + ((ToolStripButton)sender).Text + "\" + "窗体");
    }

    private void Button_ClewBargain_Click(object sender, EventArgs e)
    {
        if (Tool_ClewBargain.Enabled == true)
            Tool_ClewBargain_Click(sender, e);
        else
            MessageBox.Show("当前用户无权限调用" + "\" +
                + ((ToolStripButton)sender).Text + "\" + "窗体");
    }

    private void Botton_AddressBook_Click(object sender, EventArgs e)
    {
        if (Tool_AddressBook.Enabled == true)
            Tool_AddressBook_Click(sender, e);
        else
            MessageBox.Show("当前用户无权限调用" + "\" +
                + ((ToolStripButton)sender).Text + "\" + "窗体");
    }

    private void Botton_DayWordPad_Click(object sender, EventArgs e)
    {
        if (Tool_DayWordPad.Enabled == true)
            Tool_DayWordPad_Click(sender, e);
        else
            MessageBox.Show("当前用户无权限调用" + "\" +
                + ((ToolStripButton)sender).Text + "\" + "窗体");
    }

    private void Tool_Counter_Click(object sender, EventArgs e)
    {
        MyMenu.Show_Form(sender.ToString().Trim(), 1);
    }
}

```

### 3.9 实现人事档案浏览模块

视频讲解  光盘：视频\第3章\实现人事档案浏览模块.avi

人事档案浏览窗体用来对职工的基本信息、家庭情况、工作简历、培训记录等进行浏览，以及进行添加、修改、删除的操作。在主窗体中，可以通过菜单栏中的“人事管理”→“人事档案浏览”命令调用人事档案浏览窗体，也可以通过“人事档案浏览”常用按钮或树形下拉列表进行调用。

人事档案浏览窗体由 4 部分组成，分别为分类查询、浏览按钮、职工名称表和信息操作，其中分类查询主要是通过职工的类别，对职工进行简单查询；浏览按钮是通过按钮对职工名称表进行浏览；职工名称表用来显示当前所记录的所有职工名称；信息操作用来对职工相关信息进行添加、修改、删除、浏览等操作，并可以将职工的基本信息在 Word 文档中以自定义表格的形式进行显示。

本窗体为了便于对职工基本信息、工作简历、家庭关系等选项卡中的信息进行添加、修改操作，主要利用了 TabControl 控件和 GroupBox 属性获取当前控件内的所有控件集，遍历当前控件内的所有可视化控件，并获取指定控件的文本信息。通过获取的文本信息，可



以根据相应的数据表字段组合成 Insert 和 Update 的 SQL 语句，以实现添加和修改的操作。

本章案例中人事档案浏览模块的具体实现步骤如下。

(1) 新建一个 Windows 窗体，命名为 F\_ManFiles.cs，主要用于实现人事档案浏览功能。人事档案浏览窗体的控件如表 3-9 所示。

表 3-9 人事档案浏览窗体的控件

| 控件类型         | 控件 ID         | 主要属性设置                      | 用 途            |
|--------------|---------------|-----------------------------|----------------|
| Button       | N_First       | 在 BackgroundImage 属性中添加背景图片 | 实现数据表的浏览       |
| DataGridView | dataGridView1 | 在 Columns 中添加两个列、编号及名称      | 在数据表中只显示两个列的信息 |
| TabControl   | tabControl1   | 在 TabPages 中添加 6 个选项卡       | 显示职工的不同信息      |

(2) 在人事档案浏览窗体加载时，首先通过 MyMeans 类中的 getDataSet()方法，利用公共变量 AllSql 所记录的 SQL 语句对职工基本信息表进行查询，并显示在 dataGridView1 控件中。为了便于在职工基本信息表中对数据的编辑，将相应数据表的信息动态添加到 ComboBox 控件中。定义一个自定义方法 Grid\_Inof()，主要将 dataGridView1 控件中的当前记录在指定的控件上进行显示。

(3) 在人事档案浏览窗体加载后，要将已记录的职工信息显示在“职工基本信息”、“家庭关系”、“培训记录”、“奖励记录”和“个人简历”选项卡的相应文本框中，要先在 dataGridView1 控件的 CellEnter 事件中通过 MyMeans 公共类中的 getDataSet()方法对相应的数据表进行查询，然后将查询的结果显示在各选项卡的 DataGridView 控件中。

(4) 本窗体的“工作简历”、“家庭关系”、“培训记录”和“奖惩记录”选项卡，都是针对某一职工进行多条记录的操作，为了便于各选项卡的添加、修改、删除操作，只有“工作简历”选项卡中放置了操作按钮，当选择其他选项卡时，将操作按钮动态移植到相应的选项卡中，并根据选项卡的不同，改变操作按钮的功能。该操作可以在 tabControl1 控件的 Click 事件中完成。

文件 F\_ManFiles.cs 的具体实现代码如下所示：

```
public partial class F_ManFile : Form
{
    public F_ManFile()
    {
        InitializeComponent();

        #region 当前窗体的所有共公变量
        DataClass.MyMeans MyDataClass = new PWMS.DataClass.MyMeans();
        ModuleClass.MyModule MyMC = new PWMS.ModuleClass.MyModule();
        public static DataSet MyDS_Grid;
        public static string tem_Field = "";
        public static string tem_Value = "";
        public static string tem_ID = "";
        public static int hold_n = 0;
        public static byte[] imgBytesIn; //用来存储图片的二进制数
        public static int Ima_n = 0; //判断是否对图片进行了操作
        public static string Part_ID = ""; //存储数据表的 ID 信息
        #endregion
    }
}
```

```

public void ShowData_Image(byte[] DI, PictureBox Ima) //显示数据库图片
{
    byte[] buffer = DI;
    MemoryStream ms = new MemoryStream(buffer);
    Ima.Image = Image.FromStream(ms);
}

#region 显示“职工基本信息”表中的指定记录
/// <summary>
/// 动态读取指定的记录行, 并进行显示
/// </summary>
/// <param name="DGrid">DataGridView 控件</param>
/// <returns>返回 string 对象</returns>
public string Grid_Inof(DataGridView DGrid)
{
    byte[] pic; //定义一个字节数组
    //当 DataGridView 控件的记录>1 时, 将当前行中的信息显示在相应的控件上
    if (DGrid.RowCount > 1)
    {
        S_0.Text = DGrid[0, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_1.Text = DGrid[1, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_2.Text = Convert.ToString(DGrid[2,
            DGrid.CurrentCell.RowIndex].Value).Trim();
        S_3.Text = MyMC.Date_Format(Convert.ToString(DGrid[3,
            DGrid.CurrentCell.RowIndex].Value).Trim());
        S_4.Text = Convert.ToString(DGrid[4,
            DGrid.CurrentCell.RowIndex].Value).Trim();
        S_5.Text = DGrid[5, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_6.Text = DGrid[6, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_7.Text = DGrid[7, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_8.Text = DGrid[8, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_9.Text = DGrid[9, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_10.Text = MyMC.Date_Format(Convert.ToString(DGrid[10,
            DGrid.CurrentCell.RowIndex].Value).Trim());
        S_11.Text = Convert.ToString(DGrid[11,
            DGrid.CurrentCell.RowIndex].Value).Trim();
        S_12.Text = DGrid[12, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_13.Text = DGrid[13, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_14.Text = DGrid[14, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_15.Text = DGrid[15, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_16.Text = DGrid[16, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_17.Text = DGrid[17, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_18.Text = DGrid[18, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_19.Text = DGrid[19, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_20.Text = DGrid[20, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_21.Text = MyMC.Date_Format(Convert.ToString(DGrid[21,
            DGrid.CurrentCell.RowIndex].Value).Trim());
        S_22.Text = DGrid[22, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_23.Text = DGrid[24, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_24.Text = DGrid[25, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_25.Text = Convert.ToString(DGrid[26,
            DGrid.CurrentCell.RowIndex].Value).Trim();
        S_26.Text = DGrid[27, DGrid.CurrentCell.RowIndex].Value.ToString();
        S_27.Text = MyMC.Date_Format(Convert.ToString(DGrid[28,
            DGrid.CurrentCell.RowIndex].Value).Trim());
        S_28.Text = MyMC.Date_Format(Convert.ToString(DGrid[29,
            DGrid.CurrentCell.RowIndex].Value).Trim());
        S_29.Text = Convert.ToString(DGrid[30,
            DGrid.CurrentCell.RowIndex].Value).Trim();
        try
    }
}

```



```

{
    //将数据库中的图片存入到字节数组中
    pic =
        (byte[]) (MyDS_Grid.Tables[0].Rows[DGrid.CurrentRow.Index][23]);
    MemoryStream ms = new MemoryStream(pic); //将字节数组存入到二进制流中
    S_Photo.Image = Image.FromStream(ms); //二进制流在 Image 控件中显示
}
catch { S_Photo.Image = null; } //当出现错误时, 将 Image 控件清空
tem_ID = S_0.Text.Trim(); //获取当前职工的编号
//返回当前职工的姓名
return DGrid[1, DGrid.CurrentRow.Index].Value.ToString();
}
else
{
    //使用 MyMeans 公共类中的 Clear_Control() 方法清空指定控件集中的相应控件
    MyMC.Clear_Control(tabControl1.TabPages[0].Controls);
    tem_ID = "";
    return "";
}
}
#endregion

#region 按条件显示“职工基本信息”表的内容
/// <summary>
/// 通过公共变量动态进行查询
/// </summary>
/// <param name="C_Value">条件值</param>
public void Condition_Lookup(string C_Value)
{
    MyDS_Grid = MyDataClass.getDataSet("Select * from tb_Stuffbasic where "
        + tem_Field + "='" + tem_Value + "'", "tb_Stuffbasic");
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    textBox1.Text = Grid_Inof(dataGridView1); //显示职工信息表的当前记录
}
#endregion

#region 将图片转换成字节数组
public void Read_Image(OpenFileDialog openF, PictureBox MyImage) //
{
    openF.Filter = "*.jpg|*.jpg|*.bmp|*.bmp"; //指定 OpenFileDialog 控件打开的文件格式
    if (openF.ShowDialog(this) == DialogResult.OK) //如果打开了图片文件
    {
        try
        {
            //将图片文件存入到 PictureBox 控件中
            MyImage.Image = System.Drawing.Image.FromFile(openF.FileName);
            string strimg = openF.FileName.ToString(); //记录图片的所在的路径
            //将图片以文件流的形式进行保存
            FileStream fs = new FileStream(strimg, FileMode.Open, FileAccess.Read);
            BinaryReader br = new BinaryReader(fs);
            imgBytesIn = br.ReadBytes((int)fs.Length); //将流读入字节数组中
        }
        catch
        {
            MessageBox.Show("您选择的图片不能被读取或文件类型不对!", "错误",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            S_Photo.Image = null;
        }
    }
}
}

```



```
#endregion

private void F_ManFile_Load(object sender, EventArgs e)
{
    //用 dataGridView1 控件显示职工的名称
    MyDS_Grid = MyDataClass.getDataSet(DataClass.MyMeans.AllSql, "tb_Stuffbusic");
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    dataGridView1.AutoGenerateColumns = true; //是否自动创建列
    dataGridView1.Columns[0].Width = 60;
    dataGridView1.Columns[1].Width = 80;

    //隐藏 dataGridView1 控件中不需要的列字段
    for (int i=2; i<dataGridView1.ColumnCount; i++)
    {
        dataGridView1.Columns[i].Visible = false;
    }

    MyMC.MaskedTextBox_Format(S_3); //指定 MaskedTextBox 控件的格式
    MyMC.MaskedTextBox_Format(S_10);
    MyMC.MaskedTextBox_Format(S_21);
    MyMC.MaskedTextBox_Format(S_27);
    MyMC.MaskedTextBox_Format(S_28);

    MyMC.CoPassData(S_2, "tb_Folk"); //向“民族类别”列表框中添加信息
    MyMC.CoPassData(S_5, "tb_Kultur"); //向“文化程度”列表框中添加信息
    MyMC.CoPassData(S_8, "tb_Visage"); //向“政治面貌”列表框中添加信息
    MyMC.CoPassData(S_12, "tb_EmployeeGenre"); //向“职工类别”列表框中添加信息
    MyMC.CoPassData(S_13, "tb_Business"); //向“职务类别”列表框中添加信息
    MyMC.CoPassData(S_14, "tb_Laborage"); //向“工资类别”列表框中添加信息
    MyMC.CoPassData(S_15, "tb_Branch"); //向“部门类别”列表框中添加信息
    MyMC.CoPassData(S_16, "tb_Duthcall"); //向“职称类别”列表框中添加信息
    MyMC.CityInfo(S_23, "select distinct beaware from tb_City", 0);

    //使 S_BeAware 控件具有查询功能
    S_23.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
    S_23.AutoCompleteSource = AutoCompleteSource.ListItems;

    textBox1.Text = Grid_Inof(dataGridView1); //显示职工信息表的首记录
    DataClass.MyMeans.AllSql = "Select * from tb_Stuffbusic";
}

private void Sut_Add_Click(object sender, EventArgs e)
{
    //清空职工基本信息的相应文本框
    MyMC.Clear_Control(tabControll.TabPages[0].Controls);
    S_0.Text = MyMC.GetAutocoding("tb_Stuffbusic", "ID"); //自动添加编号
    hold_n = 1; //用于记录添加操作的标识
    MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Save, 0, 0, 1, 1);
    groupBox5.Text = "当前正在添加信息";
    Img_Clear.Enabled = true; //使图片选择按钮为可用状态
    Img_Save.Enabled = true;
}

private void S_BeAware_TextChanged(object sender, EventArgs e)
{
    S_24.Items.Clear();
    MyMC.CityInfo(S_24, "select beaware,city from tb_City where beaware='"
        + S_23.Text.Trim() + "'", 1);
}
```

```
private void tabControl1_Click(object sender, EventArgs e)
{
    groupBox5.Enabled = true;
    Sut_Delete.Enabled = true;
    MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Save, 1, 1, 0, 0);
    if (tabControl1.SelectedTab.Name == "tabPage1") //如果选择的是“职工基本信息”选项卡
    {
        hold_n = 0; //恢复原始标识
        MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Save, 1, 1, 0, 0);
        groupBox5.Text = "";
        Ima_n = 0; //标识是否选择了职工照片
        Img_Clear.Enabled = false; //使按钮为不可用状态
        Img_Save.Enabled = false;
        Sub_Table.Enabled = true;
    }
    //如果选择的是“工作经历”、“家庭关系”、“培训记录”和“奖惩记录”选项卡
    if (tabControl1.SelectedTab.Name == "tabPage2"
        || tabControl1.SelectedTab.Name == "tabPage3"
        || tabControl1.SelectedTab.Name == "tabPage4"
        || tabControl1.SelectedTab.Name == "tabPage5")
    {
        groupBox5.Enabled = false; //使窗体中的操作按钮为不可用状态
        Sub_Table.Enabled = false;
        if (tabControl1.SelectedTab.Name == "tabPage2") //“工作经历”选项卡
        {
            groupBox6.Parent = (TabPage)tabPage2;
            MyMC.MaskedTextBox_Format(Word_2); //指定 MaskedTextBox 控件的格式
            MyMC.MaskedTextBox_Format(Word_3);
        }
        if (tabControl1.SelectedTab.Name == "tabPage3") //“家庭关系”选项卡
        {
            groupBox6.Parent = (TabPage)tabPage3;
            MyMC.MaskedTextBox_Format(Family_4);
        }
        if (tabControl1.SelectedTab.Name == "tabPage4") //“培训记录”选项卡
        {
            groupBox6.Parent = (TabPage)tabPage4;
            MyMC.MaskedTextBox_Format(TrainNote_3);
            MyMC.MaskedTextBox_Format(TrainNote_4);
        }
        if (tabControl1.SelectedTab.Name == "tabPage5") //“奖惩记录”选项卡
        {
            groupBox6.Parent = (TabPage)tabPage5;
            MyMC.MaskedTextBox_Format(RANDP_3);
            MyMC.MaskedTextBox_Format(RANDP_5);
            MyMC.CoPassData(RANDP_2, "tb_RPKind"); //向“奖惩类别”列表框中添加信息
        }
        MyMC.Ena_Button(Part_Add, Part_Amend, Part_Cancel, Part_Save, 1, 1, 0, 0);
    }
    if (tabControl1.SelectedTab.Name == "tabPage6") //“个人简历”选项卡
    {
        //使窗体中的操作按钮为不可用
        MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Delete, 0, 0, 0, 0);
        Sut_Save.Enabled = true; //将窗体中的“保存”按钮设为可用状态
    }
}

private void comboBox1_TextChanged(object sender, EventArgs e)
{

```

```
switch (comboBox1.SelectedIndex) //向 comboBox2 控件中添加相应的查询条件
{
    case 0: //职工姓名
    {
        MyMC.CityInfo(comboBox2,
            "select distinct StuffName from tb_Stuffbasic", 0);
        tem_Field = "StuffName";
        break;
    }
    case 1: //性别
    {
        comboBox2.Items.Clear();
        comboBox2.Items.Add("男");
        comboBox2.Items.Add("女");
        tem_Field = "Sex";
        break;
    }
    case 2: //民族类别
    {
        MyMC.CoPassData(comboBox2, "tb_Folk");
        tem_Field = "Folk";
        break;
    }
    case 3: //文化程度
    {
        MyMC.CoPassData(comboBox2, "tb_Kultur");
        tem_Field = "Kultur";
        break;
    }
    case 4: //政治面貌
    {
        MyMC.CoPassData(comboBox2, "tb_Visage");
        tem_Field = "Visage";
        break;
    }
    case 5: //职工类别
    {
        MyMC.CoPassData(comboBox2, "tb_EmployeeGenre");
        tem_Field = "Employee";
        break;
    }
    case 6: //职务类别
    {
        MyMC.CoPassData(comboBox2, "tb_Business");
        tem_Field = "Business";
        break;
    }
    case 7: //部门类别
    {
        MyMC.CoPassData(comboBox2, "tb_Branch");
        tem_Field = "Branch";
        break;
    }
    case 8: //职称类别
    {
        MyMC.CoPassData(comboBox2, "tb_Duthcall");
        tem_Field = "Duthcall";
        break;
    }
}
```



```

        case 9: //工资类别
        {
            MyMC.CoPassData(comboBox2, "tb_Laborage");
            tem_Field = "Laborage";
            break;
        }
    }
}

private void N_First_Click(object sender, EventArgs e)
{
    try
    {
        int ColInd = 0;
        if (dataGridView1.CurrentCell.ColumnIndex == -1
            || dataGridView1.CurrentCell.ColumnIndex > 1)
            ColInd = 0;
        else
            ColInd = dataGridView1.CurrentCell.ColumnIndex;
        if (((Button)sender).Name == "N_First")
        {
            dataGridView1.CurrentCell = this.dataGridView1[ColInd, 0];
            MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 0, 0, 1, 1);
        }
        if (((Button)sender).Name == "N_Previous")
        {
            if (dataGridView1.CurrentCell.RowIndex == 0)
            {
                MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 0, 0, 1, 1);
            }
            else
            {
                dataGridView1.CurrentCell = this.dataGridView1[ColInd,
                    dataGridView1.CurrentCell.RowIndex - 1];
                MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 1, 1);
            }
        }
        if (((Button)sender).Name == "N_Next")
        {
            if (dataGridView1.CurrentCell.RowIndex == dataGridView1.RowCount - 2)
            {
                MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 0, 0);
            }
            else
            {
                dataGridView1.CurrentCell = this.dataGridView1[ColInd,
                    dataGridView1.CurrentCell.RowIndex + 1];
                MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 1, 1);
            }
        }
        if (((Button)sender).Name == "N_Cauda")
        {
            dataGridView1.CurrentCell = this.dataGridView1[ColInd,
                dataGridView1.RowCount - 2];
            MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 0, 0);
        }
    }
    catch {}
}

```

```

private void N_Previous_Click(object sender, EventArgs e)
{
    N_First_Click(sender, e);
}

private void N_Next_Click(object sender, EventArgs e)
{
    N_First_Click(sender, e);
}

private void N_Cauda_Click(object sender, EventArgs e)
{
    N_First_Click(sender, e);
}

private void dataGridView1_CellEnter(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        if (dataGridView1.CurrentCell.RowIndex > -1)
        {
            textBox1.Text = Grid_Inof(dataGridView1); //显示职工信息表的当前记录
            //使窗体中的编辑按钮可用
            MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 1, 1);
            //获取工作简历表中的信息
            DataSet WDset = MyDataClass.getDataSet("select Sut_ID,ID,BeginDate
            as 开始时间,EndDate as 结束时间, Branch as 部门, Business as 职务, WordUnit
            as 工作单位 from tb_WordResume where Sut_ID='" + tem_ID + "'",
            "tb_WordResume");
            //将WDset 存储的信息显示在 dataGridView2 控件中
            MyMC.Correlation_Table(WDset, dataGridView2);
            if (WDset.Tables[0].Rows.Count < 1) //当WDset 中没有信息时
                //清空相应的控件
                MyMC.Clear_Grids(
                    WDset.Tables[0].Columns.Count, groupBox7.Controls, "Word_");
            //获取家庭关系表中的信息
            DataSet FDset = MyDataClass.getDataSet("select Sut_ID,ID,LeaguerName
            as 家庭成员名称,Nexus as 与本人的关系, BirthDate as 出生日期, WordUnit
            as 工作单位, Business as 职务, Visage as 政治面貌, Phone
            as 电话 from tb_Family where Sut_ID='" + tem_ID + "'", "tb_Family");
            MyMC.Correlation_Table(FDset, dataGridView3);
            if (FDset.Tables[0].Rows.Count < 1)
                MyMC.Clear_Grids(FDset.Tables[0].Columns.Count,
                    groupBox10.Controls, "Famity_");
            //获取工作简历表中的信息
            DataSet TDset = MyDataClass.getDataSet("select Sut_ID,ID,TrainFashion
            as 培训方式,BeginDate as 培训开始时间, EndDate as 培训结束时间, Speciality
            as 培训专业, TrainUnit as 培训单位, KulturMemo as 培训内容, Charge as 费用,
            Effect as 效果 from tb_TrainNote where Sut_ID='" + tem_ID + "'",
            "tb_TrainNote");
            MyMC.Correlation_Table(TDset, dataGridView4);
            if (TDset.Tables[0].Rows.Count < 1)
                MyMC.Clear_Grids(TDset.Tables[0].Columns.Count,
                    groupBox12.Controls, "TrainNote_");
            //获取奖惩记录表中的信息
            DataSet RDset = MyDataClass.getDataSet("select Sut_ID,ID,RPKind
            as 奖惩种类,RPDate as 奖惩时间, SealMan as 批准人, QuashDate as 撤消时间,
            QuashWhys as 撤消原因 from tb_RANDP where Sut_ID='" + tem_ID + "'",
            "tb_RANDP");
            MyMC.Correlation_Table(RDset, dataGridView5);
        }
    }
}

```

```

        if (RDset.Tables[0].Rows.Count < 1)
            MyMC.Clear_Grids(RDset.Tables[0].Columns.Count,
                groupBox14.Controls, "RANDP_");
        //获取个人简历表中的信息
        SqlDataReader Read_Memo = MyDataClass.getcom(
            "Select * from tb_Individual where ID='" + tem_ID + "'");
        if (Read_Memo.Read())
            Ind_Mome.Text = Read_Memo[1].ToString();
        else
            Ind_Mome.Clear();

        //MyMC.Show_DGrid(dataGridView2, groupBox7.Controls, "Word_");
    }
}
catch {}
}

private void comboBox2_TextChanged(object sender, EventArgs e)
{
    try
    {
        tem_Value = comboBox2.SelectedItem.ToString();
        Condition_Lookup(tem_Value);
    }
    catch
    {
        comboBox2.Text = "";
        MessageBox.Show("只能以选择方式查询。");
    }
}

private void button1_Click(object sender, EventArgs e)
{
    tem_Field = "";
    MyDS_Grid = MyDataClass.getDataSet(DataClass.MyMeans.AllSql, "tb_Stuffbusic");
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    textBox1.Text = Grid_Inof(dataGridView1); //显示职工信息表的当前记录
}

private void Sut_Amend_Click(object sender, EventArgs e)
{
    hold_n = 2; //用于记录修改操作的标识
    MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Save, 0, 0, 1, 1);
    groupBox5.Text = "当前正在修改信息";
    Img_Clear.Enabled = true; //使图片选择按钮为可用状态
    Img_Save.Enabled = true;
}

private void Sut_Cancel_Click(object sender, EventArgs e)
{
    hold_n = 0; //恢复原始标识
    MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Save, 1, 1, 0, 0);
    groupBox5.Text = "";
    Ima_n = 0;
    if (tem_Field == "")
        button1_Click(sender, e);
    else
        Condition_Lookup(tem_Value);
    Img_Clear.Enabled = false;
    Img_Save.Enabled = false;
}

```



```

private void Sut_Save_Click(object sender, EventArgs e)
{
    if (tabControll1.SelectedTab.Name == "tabPage6") //如果当前是“个人简历”选项卡
    {
        //通过 MyMeans 公共类中的 getcom() 方法查询当前职工是否添加了个人简历
        SqlDataReader Read_Memo = MyDataClass.getcom(
            "Select * from tb_Individual where ID='" + tem_ID + "'");
        if (Read_Memo.Read()) //如果有记录
            //将当前设置的个人简历进行修改
            MyDataClass.getsqlcom("update tb_Individual set Memo='"
                + Ind_Mome.Text + "' where ID='" + tem_ID + "'");
        else
            //如果没有记录, 则进行添加操作
            MyDataClass.getsqlcom("insert into tb_Individual (ID,Memo) values('"
                + tem_ID + "', '" + Ind_Mome.Text + "')");
    }
    else //如果当前是“职工基本信息”选项卡
    {
        //定义字符串变量, 并存储将“职工基本信息表”中的所有字段
        string All_Field =
            "ID, StuffName, Folk, Birthday, Age, Kultur, Marriage, Sex, Visage, IDCard, Workdate, WorkLength,
            Employee, Business, Laborage, Branch, Duthcall, Phone, Handset, School, Speciality, Graduated
            ate, Address, BeAware, City, M_Pay, Bank, Pact_B, Pact_E, Pact_Y";

        if (hold_n == 1 || hold_n == 2) //判断当前是添加, 还是修改操作
        {
            ModuleClass.MyModule.ADDs = ""; //清空 MyModule 公共类中的 ADDs 变量
            //用 MyModule 公共类中的 Part_SaveClass() 方法组合添加或修改 SQL 语句
            MyMC.Part_SaveClass(All_Field, S_0.Text.Trim(), "",
                tabControll1.TabPages[0].Controls, "S_", "tb_Stuffbasic", 30, hold_n);
            //如果 ADDs 变量不为空,
            //则通过 MyMeans 公共类中的 getsqlcom() 方法执行添加、修改操作
            if (ModuleClass.MyModule.ADDs != "")
                MyDataClass.getsqlcom(ModuleClass.MyModule.ADDs);
        }
        if (Ima_n > 0) //如果图片标识大于 0
        {
            //通过 MyModule 公共类中的 SaveImage() 方法将图片存入数据库中
            MyMC.SaveImage(S_0.Text.Trim(), imgBytesIn);
        }
        Sut_Cancel_Click(sender, e); //调用“取消”按钮的单击事件
    }
}

private void button7_Click(object sender, EventArgs e)
{
    Read_Image(openFileDialog1, S_Photo);
    Ima_n = 1;
}

private void button8_Click(object sender, EventArgs e)
{
    S_Photo.Image = null;
    imgBytesIn = new byte[65536];
    Ima_n = 2;
}

private void button2_Click(object sender, EventArgs e)
{

```

```

hold_n = 1;
if (tabControll.SelectedTab.Name == "tabPage2")
{
    MyMC.Clear_Control(this.groupBox7.Controls);
    Part_ID = MyMC.GetAutocoding("tb_WordResume", "ID"); //自动添加编号;
}
if (tabControll.SelectedTab.Name == "tabPage3")
{
    MyMC.Clear_Control(this.groupBox10.Controls);
    Part_ID = MyMC.GetAutocoding("tb_Family", "ID"); //自动添加编号;
}
if (tabControll.SelectedTab.Name == "tabPage4")
{
    MyMC.Clear_Control(this.groupBox12.Controls);
    Part_ID = MyMC.GetAutocoding("tb_TrainNote", "ID"); //自动添加编号;
}
if (tabControll.SelectedTab.Name == "tabPage5")
{
    MyMC.Clear_Control(this.groupBox14.Controls);
    Part_ID = MyMC.GetAutocoding("tb_RANDP", "ID"); //自动添加编号;
}
MyMC.Ena_Button(Part_Add, Part_Amend, Part_Cancel, Part_Save, 1, 0, 1, 1);
}

private void Part_Save_Click(object sender, EventArgs e)
{
    string s = "";
    if (tabControll.SelectedTab.Name == "tabPage2")
    {
        s = "ID,Sut_ID,BeginDate,EndDate,Branch,Business,WordUnit";
        //"select Sut_ID,ID,BeginDate as 开始时间,EndDate as 结束时间, Branch as 部门,
        // Business as 职务, WordUnit as 工作单位 from tb_WordResume"
        ModuleClass.MyModule.ADDs = "";
        if (hold_n == 2)
        {
            if (dataGridView2.RowCount < 2)
            {
                MessageBox.Show("数据表为空, 不可以修改");
            }
            else
            {
                Part_ID = dataGridView2[1,
                    dataGridView2.CurrentCell.RowIndex].Value.ToString();
            }
            MyMC.Part_SaveClass(s, tem_ID, Part_ID, this.groupBox7.Controls,
                "Word_", "tb_WordResume", 7, hold_n);
        }
    }
    if (tabControll.SelectedTab.Name == "tabPage3")
    {
        s =
            "ID,Sut_ID,LeaguerName,Nexus,BirthDate,WordUnit,Business,Visage,Phone";
        ModuleClass.MyModule.ADDs = "";
        if (hold_n == 2)
        {
            if (dataGridView3.RowCount < 2)
            {
                MessageBox.Show("数据表为空, 不可以修改");
            }
            else
            {
                Part_ID = dataGridView3[1,
                    dataGridView3.CurrentCell.RowIndex].Value.ToString();
            }
        }
    }
}

```



```

        MyMC.Part_SaveClass(s, tem_ID, Part_ID, this.groupBox10.Controls,
            "Famity_", "tb_Family", 9, hold_n);
    }
    if (tabControll1.SelectedTab.Name == "tabPage4")
    {
        s = "ID,Sut_ID,TrainFashion,BeginDate,EndDate,Speciality,TrainUnit,
            KulturMemo,Charge,Effect";
        ModuleClass.MyModule.ADDs = "";
        if (hold_n == 2)
        {
            if (dataGridView4.RowCount < 2)
            {
                MessageBox.Show("数据表为空, 不可以修改");
            }
            else
            {
                Part_ID = dataGridView4[1,
                    dataGridView4.CurrentCell.RowIndex].Value.ToString();
            }
            MyMC.Part_SaveClass(s, tem_ID, Part_ID, this.groupBox12.Controls,
                "TrainNote_", "tb_TrainNote", 10, hold_n);
        }
    }
    if (tabControll1.SelectedTab.Name == "tabPage5")
    {
        s = "ID,Sut_ID,RPKind,RPDate,SealMan,QuashDate,QuashWhys";
        ModuleClass.MyModule.ADDs = "";
        if (hold_n == 2)
        {
            if (dataGridView5.RowCount < 2)
            {
                MessageBox.Show("数据表为空, 不可以修改");
            }
            else
            {
                Part_ID = dataGridView5[1,
                    dataGridView5.CurrentCell.RowIndex].Value.ToString();
            }
            MyMC.Part_SaveClass(s, tem_ID, Part_ID, this.groupBox14.Controls, "RANDP_",
                "tb_RANDP", 7, hold_n);
        }
    }
    if (ModuleClass.MyModule.ADDs != "")
        MyDataClass.getsqlcom(ModuleClass.MyModule.ADDs);
    Part_Cancel_Click(sender, e);
}

private void Part_Amend_Click(object sender, EventArgs e)
{
    hold_n = 2;
    MyMC.Ena_Button(Part_Add, Part_Amend, Part_Cancel, Part_Save, 0, 1, 1, 1);
}

private void Part_Cancel_Click(object sender, EventArgs e)
{
    if (tabControll1.SelectedTab.Name == "tabPage2")
    {
        DataSet WDset = MyDataClass.getDataSet("select Sut_ID,ID,BeginDate
            as 开始时间,EndDate as 结束时间, Branch as 部门, Business as 职务, WordUnit
            as 工作单位 from tb_WordResume where Sut_ID='" + tem_ID + "'",
            "tb_WordResume");
        MyMC.Correlation_Table(WDset, dataGridView2);
    }
    if (tabControll1.SelectedTab.Name == "tabPage3")
    {

```



```

        DataSet FDset = MyDataClass.getDataSet("select Sut_ID,ID,LeaguerName
        as 家庭成员名称,Nexus as 与本人的关系, BirthDate as 出生日期, WordUnit
        as 工作单位, Business as 职务, Visage as 政治面貌, Phone as 电话 from tb_Family
        where Sut_ID='" + tem_ID + "'", "tb_Family");
        MyMC.Correlation_Table(FDset, dataGridView3);
    }
    if (tabControll1.SelectedTab.Name == "tabPage4")
    {
        DataSet TDset = MyDataClass.getDataSet("select Sut_ID,ID,TrainFashion
        as 培训方式,BeginDate as 培训开始时间, EndDate as 培训结束时间, Speciality
        as 培训专业, TrainUnit as 培训单位, KulturMemo as 培训内容, Charge as 费用,
        Effect as 效果 from tb_TrainNote where Sut_ID='" + tem_ID + "'",
        "tb_TrainNote");
        MyMC.Correlation_Table(TDset, dataGridView4);
    }
    if (tabControll1.SelectedTab.Name == "tabPage5")
    {
        DataSet RDset = MyDataClass.getDataSet("select Sut_ID,ID,RPKind
        as 奖惩种类,RPDate as 奖惩时间, SealMan as 批准人, QuashDate as 撤消时间,
        QuashWhys as 撤消原因 from tb_RANDP where Sut_ID='" + tem_ID + "'",
        "tb_RANDP");
        MyMC.Correlation_Table(RDset, dataGridView5);
    }
    hold_n = 0; //恢复原始标识
    MyMC.Ena_Button(Part_Add, Part_Amend, Part_Cancel, Part_Save, 1, 1, 0, 0);
}

private void dataGridView2_CellEnter(object sender, DataGridViewCellEventArgs e)
{
    MyMC.Show_DGrid(dataGridView2, groupBox7.Controls, "Word_");
}

private void S_Pact_B_Leave(object sender, EventArgs e)
{
    MyMC.Estimate_Date((MaskedTextBox) sender);
}

private void S_Pact_B_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, "", 0);
}

private void S_Pact_E_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, "", 0);
}

private void S_Pact_E_Leave(object sender, EventArgs e)
{
    bool TDate = MyMC.Estimate_Date((MaskedTextBox) sender);
    if (TDate == true)
    {
        if (MyMC.Date_Format(S_27.Text) != "" && MyMC.Date_Format(S_28.Text) != "")
        {
            if (Convert.ToDateTime(S_28.Text) <= Convert.ToDateTime(S_27.Text))
                MessageBox.Show("当前日期必须大于它前一个日期。");
        }
    }
}

private void S_GraduateDate_Leave(object sender, EventArgs e)
{
    MyMC.Estimate_Date((MaskedTextBox) sender);
}

```

```

    }

    private void S_GraduateDate_KeyPress(object sender, KeyPressEventArgs e)
    {
        MyMC.Estimate_Key(e, "", 0);
    }

    private void S_Workdate_Leave(object sender, EventArgs e)
    {
        MyMC.Estimate_Date((MaskedTextBox)sender);
    }

    private void S_Workdate_KeyPress(object sender, KeyPressEventArgs e)
    {
        MyMC.Estimate_Key(e, "", 0);
    }

    private void S_M_Pay_KeyPress(object sender, KeyPressEventArgs e)
    {
        MyMC.Estimate_Key(e, ((TextBox)sender).Text, 1);
    }

    private void S_Pact_Y_KeyPress(object sender, KeyPressEventArgs e)
    {
        MyMC.Estimate_Key(e, "", 0);
    }

    private void S_Age_KeyPress(object sender, KeyPressEventArgs e)
    {
        MyMC.Estimate_Key(e, "", 0);
    }

    private void S_WorkLength_KeyPress(object sender, KeyPressEventArgs e)
    {
        MyMC.Estimate_Key(e, "", 0);
    }

    private void Part_Delete_Click(object sender, EventArgs e)
    {
        string Delete_Table = "";
        string Delete_ID = "";
        if (tabControll1.SelectedTab.Name == "tabPage2")
        {
            if (dataGridView2.RowCount < 2)
            {
                MessageBox.Show("数据表为空, 不可以删除。");
                return;
            }
            MyMC.Clear_Control(this.groupBox7.Controls);
            Delete_ID = dataGridView2[1,
                dataGridView2.CurrentCell.RowIndex].Value.ToString();
            Delete_Table = "tb_WordResume";
        }
        if (tabControll1.SelectedTab.Name == "tabPage3")
        {
            if (dataGridView3.RowCount < 2)
            {
                MessageBox.Show("数据表为空, 不可以删除。");
                return;
            }
            MyMC.Clear_Control(this.groupBox10.Controls);
        }
    }

```

```

Delete_ID = dataGridView3[1,
    dataGridView3.CurrentCell.RowIndex].Value.ToString();
Delete_Table = "tb_Family";
}
if (tabControl1.SelectedTab.Name == "tabPage4")
{
    if (dataGridView4.RowCount < 2)
    {
        MessageBox.Show("数据表为空, 不可以删除。");
        return;
    }
    MyMC.Clear_Control(this.groupBox12.Controls);
    Delete_ID = dataGridView4[1,
        dataGridView4.CurrentCell.RowIndex].Value.ToString();
    Delete_Table = "tb_TrainNote";
}
if (tabControl1.SelectedTab.Name == "tabPage5")
{
    if (dataGridView5.RowCount < 2)
    {
        MessageBox.Show("数据表为空, 不可以删除。");
        return;
    }
    MyMC.Clear_Control(this.groupBox14.Controls);
    Delete_ID = dataGridView5[1,
        dataGridView5.CurrentCell.RowIndex].Value.ToString();
    Delete_Table = "tb_RANDP";
}
if ((Delete_ID.Trim()).Length > 0)
{
    MyDataClass.getsqlcom(
        "Delete " + Delete_Table + " where ID='" + Delete_ID + "'");
    Part_Cancel_Click(sender, e);
}
}

private void Sut_Delete_Click(object sender, EventArgs e)
{
    if (dataGridView1.RowCount < 2) //判断 dataGridView1 控件中是否有记录
    {
        MessageBox.Show("数据表为空, 不可以删除。");
        return;
    }
    //删除职工信息表中的当前记录, 及其他相关表中的信息
    MyDataClass.getsqlcom("Delete tb_Stuffbasic where ID='" + S_0.Text.Trim() + "'");
    MyDataClass.getsqlcom(
        "Delete tb_WordResume where Sut_ID='" + S_0.Text.Trim() + "'");
    MyDataClass.getsqlcom("Delete tb_Family where Sut_ID='" + S_0.Text.Trim() + "'");
    MyDataClass.getsqlcom(
        "Delete tb_TrainNote where Sut_ID='" + S_0.Text.Trim() + "'");
    MyDataClass.getsqlcom("Delete tb_RANDP where Sut_ID='" + S_0.Text.Trim() + "'");
    MyDataClass.getsqlcom(
        "Delete tb_WordResume where Sut_ID='" + S_0.Text.Trim() + "'");
    MyDataClass.getsqlcom("Delete tb_Individual where ID='" + S_0.Text.Trim() + "'");
    Sut_Cancel_Click(sender, e); //调用“取消”按钮的单击事件
}

private void but_Table_Click(object sender, EventArgs e)
{
    object Nothing = System.Reflection.Missing.Value;

```



```

object missing = System.Reflection.Missing.Value;
//创建 Word 文档
Word.Application wordApp = new Word.ApplicationClass();
Word.Document wordDoc =
    wordApp.Documents.Add(ref Nothing, ref Nothing, ref Nothing, ref Nothing);
wordApp.Visible = true;

//设置文档宽度
wordApp.Selection.PageSetup.LeftMargin =
    wordApp.CentimetersToPoints(float.Parse("2"));
wordApp.ActiveWindow.ActivePane.HorizontalPercentScrolled = 11;
wordApp.Selection.PageSetup.RightMargin =
    wordApp.CentimetersToPoints(float.Parse("2"));

Object start = Type.Missing;
Object end = Type.Missing;

PictureBox pp = new PictureBox(); //新建一个 PictureBox 控件
int pl = 0;
for (int i=0; i<MyDS_Grid.Tables[0].Rows.Count; i++)
{
    try
    {
        //将数据库中的图片转换成二进制流
        byte[] pic = (byte[]) (MyDS_Grid.Tables[0].Rows[i][23]);
        MemoryStream ms = new MemoryStream(pic); //将字节数组存入到二进制流中
        pp.Image = Image.FromStream(ms); //二进制流在 Image 控件中显示
        pp.Image.Save(@"C:\22.bmp"); //将图片存入到指定的路径
    }
    catch
    {
        pl = 1;
    }
    object rng = Type.Missing;
    string strInfo =
        "职工基本信息表" + "(" + MyDS_Grid.Tables[0].Rows[i][1].ToString() + ")";
    start = 0;
    end = 0;
    wordDoc.Range(ref start, ref end).InsertBefore(strInfo); //插入文本
    wordDoc.Range(ref start, ref end).Font.Name = "Verdana"; //设置字体
    wordDoc.Range(ref start, ref end).Font.Size = 20; //设置字体大小
    wordDoc.Range(ref start, ref end).ParagraphFormat.Alignment =
        Word.WdParagraphAlignment.wdAlignParagraphCenter; //设置字体居中

    start = strInfo.Length;
    end = strInfo.Length;
    wordDoc.Range(ref start, ref end).InsertParagraphAfter(); //插入回车

    object missingValue = Type.Missing;
    //如果 location 超过已有字符的长度将会出错。一定要比“明细表”串多一个字符
    object location = strInfo.Length;
    Word.Range rng2 = wordDoc.Range(ref location, ref location);

    wordDoc.Tables.Add(rng2, 14, 6, ref missingValue, ref missingValue);
    wordDoc.Tables.Item(1).Rows.HeightRule =
        Word.WdRowHeightRule.wdRowHeightAtLeast;
    wordDoc.Tables.Item(1).Rows.Height =
        wordApp.CentimetersToPoints(float.Parse("0.8"));
    wordDoc.Tables.Item(1).Range.Font.Size = 10;
    wordDoc.Tables.Item(1).Range.Font.Name = "宋体";

```

```
//设置表格样式
wordDoc.Tables.Item(1).Borders.Item(Word.WdBorderType.wdBorderLeft)
.LineStyle = Word.WdLineStyle.wdLineStyleSingle;
wordDoc.Tables.Item(1).Borders.Item(Word.WdBorderType.wdBorderLeft)
.LineWidth = Word.WdLineWidth.wdLineWidth050pt;
wordDoc.Tables.Item(1).Borders.Item(Word.WdBorderType.wdBorderLeft)
.Color = Word.WdColor.wdColorAutomatic;
wordApp.Selection.ParagraphFormat.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphRight; //设置右对齐

//第5行显示
wordDoc.Tables.Item(1).Cell(1, 5)
.Merge(wordDoc.Tables.Item(1).Cell(5, 6));
//第6行显示
wordDoc.Tables.Item(1).Cell(6, 5)
.Merge(wordDoc.Tables.Item(1).Cell(6, 6));
//第9行显示
wordDoc.Tables.Item(1).Cell(9, 4)
.Merge(wordDoc.Tables.Item(1).Cell(9, 6));
//第12行显示
wordDoc.Tables.Item(1).Cell(12, 2)
.Merge(wordDoc.Tables.Item(1).Cell(12, 6));
//第13行显示
wordDoc.Tables.Item(1).Cell(13, 2)
.Merge(wordDoc.Tables.Item(1).Cell(13, 6));
//第14行显示
wordDoc.Tables.Item(1).Cell(14, 2)
.Merge(wordDoc.Tables.Item(1).Cell(14, 6));

//第1行赋值
wordDoc.Tables.Item(1).Cell(1, 1).Range.Text = "职工编号: ";
wordDoc.Tables.Item(1).Cell(1, 2).Range.Text =
MyDS_Grid.Tables[0].Rows[i][0].ToString();
wordDoc.Tables.Item(1).Cell(1, 3).Range.Text = "职工姓名: ";
wordDoc.Tables.Item(1).Cell(1, 4).Range.Text =
MyDS_Grid.Tables[0].Rows[i][1].ToString();

//插入图片
if (p1 == 0)
{
string FileName = @"C:\22.bmp"; //图片所在路径
object LinkToFile = false;
object SaveWithDocument = true;
//指定图片插入的区域
object Anchor = wordDoc.Tables.Item(1).Cell(1, 5).Range;
//将图片插入到单元格中
wordDoc.Tables.Item(1).Cell(1, 5).Range.InlineShapes
.AddPicture(FileName, ref LinkToFile, ref SaveWithDocument, ref Anchor);
}
p1 = 0;

//第2行赋值
wordDoc.Tables.Item(1).Cell(2, 1).Range.Text = "民族类别: ";
wordDoc.Tables.Item(1).Cell(2, 2).Range.Text =
MyDS_Grid.Tables[0].Rows[i][2].ToString();
wordDoc.Tables.Item(1).Cell(2, 3).Range.Text = "出生日期: ";
try
{
wordDoc.Tables.Item(1).Cell(2, 4).Range.Text =
```



```

        Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0]
            .Rows[i][3]).ToShortDateString());
    }
    catch { wordDoc.Tables.Item(1).Cell(2, 4).Range.Text = ""; }
    //Convert.ToString(MyDS_Grid.Tables[0].Rows[i][3]);
    //第3行赋值
    wordDoc.Tables.Item(1).Cell(3, 1).Range.Text = "年龄: ";
    wordDoc.Tables.Item(1).Cell(3, 2).Range.Text =
        Convert.ToString(MyDS_Grid.Tables[0].Rows[i][4]);
    wordDoc.Tables.Item(1).Cell(3, 3).Range.Text = "文化程度: ";
    wordDoc.Tables.Item(1).Cell(3, 4).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][5].ToString();
    //第4行赋值
    wordDoc.Tables.Item(1).Cell(4, 1).Range.Text = "婚姻: ";
    wordDoc.Tables.Item(1).Cell(4, 2).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][6].ToString();
    wordDoc.Tables.Item(1).Cell(4, 3).Range.Text = "性别: ";
    wordDoc.Tables.Item(1).Cell(4, 4).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][7].ToString();
    //第5行赋值
    wordDoc.Tables.Item(1).Cell(5, 1).Range.Text = "政治面貌: ";
    wordDoc.Tables.Item(1).Cell(5, 2).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][8].ToString();
    wordDoc.Tables.Item(1).Cell(5, 3).Range.Text = "单位工作时间: ";
    try
    {
        wordDoc.Tables.Item(1).Cell(5, 4).Range.Text =
            Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0]
                .Rows[0][10]).ToShortDateString());
    }
    catch { wordDoc.Tables.Item(1).Cell(5, 4).Range.Text = ""; }
    //第6行赋值
    wordDoc.Tables.Item(1).Cell(6, 1).Range.Text = "籍贯: ";
    wordDoc.Tables.Item(1).Cell(6, 2).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][24].ToString();
    wordDoc.Tables.Item(1).Cell(6, 3).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][25].ToString();
    wordDoc.Tables.Item(1).Cell(6, 4).Range.Text = "身份证: ";
    wordDoc.Tables.Item(1).Cell(6, 5).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][9].ToString();
    //第7行赋值
    wordDoc.Tables.Item(1).Cell(7, 1).Range.Text = "工龄: ";
    wordDoc.Tables.Item(1).Cell(7, 2).Range.Text =
        Convert.ToString(MyDS_Grid.Tables[0].Rows[i][11]);
    wordDoc.Tables.Item(1).Cell(7, 3).Range.Text = "职工类别: ";
    wordDoc.Tables.Item(1).Cell(7, 4).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][12].ToString();
    wordDoc.Tables.Item(1).Cell(7, 5).Range.Text = "职务类别: ";
    wordDoc.Tables.Item(1).Cell(7, 6).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][13].ToString();
    //第8行赋值
    wordDoc.Tables.Item(1).Cell(8, 1).Range.Text = "工资类别: ";
    wordDoc.Tables.Item(1).Cell(8, 2).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][14].ToString();
    wordDoc.Tables.Item(1).Cell(8, 3).Range.Text = "部门类别: ";
    wordDoc.Tables.Item(1).Cell(8, 4).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][15].ToString();
    wordDoc.Tables.Item(1).Cell(8, 5).Range.Text = "职称类别: ";
    wordDoc.Tables.Item(1).Cell(8, 6).Range.Text =
        MyDS_Grid.Tables[0].Rows[i][16].ToString();

```



```
//第9行赋值
wordDoc.Tables.Item(1).Cell(9, 1).Range.Text = "月工资: ";
wordDoc.Tables.Item(1).Cell(9, 2).Range.Text =
    Convert.ToString(MyDS_Grid.Tables[0].Rows[i][26]);
wordDoc.Tables.Item(1).Cell(9, 3).Range.Text = "银行账号: ";
wordDoc.Tables.Item(1).Cell(9, 4).Range.Text =
    MyDS_Grid.Tables[0].Rows[i][27].ToString();
//第10行赋值
wordDoc.Tables.Item(1).Cell(10, 1).Range.Text = "合同起始日期: ";
try
{
    wordDoc.Tables.Item(1).Cell(10, 2).Range.Text =
        Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0]
            .Rows[i][28]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(10, 2).Range.Text = ""; }
//Convert.ToString(MyDS_Grid.Tables[0].Rows[i][28]);
wordDoc.Tables.Item(1).Cell(10, 3).Range.Text = "合同结束日期: ";
try
{
    wordDoc.Tables.Item(1).Cell(10, 4).Range.Text =
        Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0]
            .Rows[i][29]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(10, 4).Range.Text = ""; }
//Convert.ToString(MyDS_Grid.Tables[0].Rows[i][29]);
wordDoc.Tables.Item(1).Cell(10, 5).Range.Text = "合同年限: ";
wordDoc.Tables.Item(1).Cell(10, 6).Range.Text =
    Convert.ToString(MyDS_Grid.Tables[0].Rows[i][30]);
//第11行赋值
wordDoc.Tables.Item(1).Cell(11, 1).Range.Text = "电话: ";
wordDoc.Tables.Item(1).Cell(11, 2).Range.Text =
    MyDS_Grid.Tables[0].Rows[i][17].ToString();
wordDoc.Tables.Item(1).Cell(11, 3).Range.Text = "手机: ";
wordDoc.Tables.Item(1).Cell(11, 4).Range.Text =
    MyDS_Grid.Tables[0].Rows[i][18].ToString();
wordDoc.Tables.Item(1).Cell(11, 5).Range.Text = "毕业时间: ";
try
{
    wordDoc.Tables.Item(1).Cell(11, 6).Range.Text =
        Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0]
            .Rows[i][21]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(11, 6).Range.Text = ""; }
//Convert.ToString(MyDS_Grid.Tables[0].Rows[i][21]);
//第12行赋值
wordDoc.Tables.Item(1).Cell(12, 1).Range.Text = "毕业学校: ";
wordDoc.Tables.Item(1).Cell(12, 2).Range.Text =
    MyDS_Grid.Tables[0].Rows[i][19].ToString();
//第13行赋值
wordDoc.Tables.Item(1).Cell(13, 1).Range.Text = "主修专业: ";
wordDoc.Tables.Item(1).Cell(13, 2).Range.Text =
    MyDS_Grid.Tables[0].Rows[i][20].ToString();
//第14行赋值
wordDoc.Tables.Item(1).Cell(14, 1).Range.Text = "家庭地址: ";
wordDoc.Tables.Item(1).Cell(14, 2).Range.Text =
    MyDS_Grid.Tables[0].Rows[i][22].ToString();

wordDoc.Range(ref start, ref end).InsertParagraphAfter(); //插入回车
wordDoc.Range(ref start, ref end).ParagraphFormat.Alignment =
```

```

        Word.WdParagraphAlignment.wdAlignParagraphCenter; //设置字体居中
    }
}

private void dataGridView3_CellEnter(object sender, DataGridViewCellEventArgs e)
{
    MyMC.Show_DGrid(dataGridView3, groupBox10.Controls, "Famity_");
}

private void dataGridView4_CellEnter(object sender, DataGridViewCellEventArgs e)
{
    MyMC.Show_DGrid(dataGridView4, groupBox12.Controls, "TrainNote_");
}

private void dataGridView5_CellEnter(object sender, DataGridViewCellEventArgs e)
{
    MyMC.Show_DGrid(dataGridView5, groupBox14.Controls, "RANDP_");
}

private void button1_Click_1(object sender, EventArgs e)
{
    //用 dataGridView1 控件显示职工的名称
    MyDS_Grid = MyDataClass.getDataSet(DataClass.MyMeans.AllSql, "tb_Stuffbasic");
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    dataGridView1.AutoGenerateColumns = true; //是否自动创建列
    dataGridView1.Columns[0].Width = 60;
    dataGridView1.Columns[1].Width = 80;

    //隐藏 dataGridView1 控件中不需要的列字段
    for (int i=2; i<dataGridView1.ColumnCount; i++)
    {
        dataGridView1.Columns[i].Visible = false;
    }
}

```

人事档案浏览窗体的执行效果如图 3-20 所示。

图 3-20 人事档案浏览窗体的执行效果



## 3.10 实现人事资料查询模块

视频讲解 光盘：视频\第3章\实现人事资料查询模块

在人事资料查询窗体中，可以通过在“基本信息”和“个人信息”区域中设置查询条件，对职工基本信息进行查询。在本节的内容中，将详细讲解实现人事资料查询模块的具体流程。

### 3.10.1 人事资料查询窗体的技术分析

人事资料查询窗体是将本窗体中的各个查询条件控件，按编码规则进行命名，可以通过各控件的部分名称，对控件集进行遍历，将相关联的控件组合成指定的查询条件，然后，在指定的数据表中进行查询。人事资料查询模块的执行效果如图 3-21 所示。

图 3-21 人事资料查询模块的执行效果

### 3.10.2 具体实现

新建一个 Windows 应用程序，将默认创建的 Windows 窗体命名为 F\_Find.cs，用于制作人事资料查询窗体。

在人事资料查询窗体加载时，首先要通过 MyModule 公共类中的 CoPassData()方法，将指定表中的数据添加到 ComboBox 控件中，然后用 dataGridView1 控件显示职工信息表中的全部记录。

在窗体上设置完查询条件后，单击“查询”按钮进行查询，该按钮是通过 MyMeans 公共类的 Find\_Grids()方法将指定控件集上的控件组合成查询语句，通过 getDataSet()方法查询



数据表中的记录，并显示在 dataGridView1 控件上。

文件 F\_Find.cs 的具体实现代码如下所示：

```
public partial class F_Find : Form
{
    public F_Find()
    {
        InitializeComponent();
    }
    ModuleClass.MyModule MyMC = new PWMS.ModuleClass.MyModule();
    DataClass.MyMeans MyDataClass = new PWMS.DataClass.MyMeans();
    public static DataSet MyDS_Grid;
    public string ARsign = " AND ";
    public static string Sut_SQL = "select ID as 编号, StuffName as 职工姓名, Folk as 民族类别, Birthday as 出生日期, Age as 年龄, Kultur as 文化程度, Marriage as 婚姻, Sex as 性别, Visage as 政治面貌, IDCard as 身份证号, Workdate as 单位工作时间, WorkLength as 工龄, Employee as 职工类别, Business as 职务类别, Laborage as 工资类别, Branch as 部门类别, Duthcall as 职称类别, Phone as 电话, Handset as 手机, School as 毕业学校, Speciality as 主修专业, GraduateDate as 毕业时间, M_Pay as 月工资, Bank as 银行账号, Pact_B as 合同开始时间, Pact_E as 合同结束时间, Pact_Y as 合同年限, BeAware as 籍贯所在省, City as 籍贯所在市 from tb_Stuffbasic";

    #region 清空控件集上的控件信息
    /// <summary>
    /// 清空 GroupBox 控件上的控件信息。
    /// </summary>
    /// <param name="n">控件个数</param>
    /// <param name="GBox">GroupBox 控件的数据集</param>
    /// <param name="TName">获取信息控件的部分名称</param>
    private void Clear_Box(int n, Control.ControlCollection GBox, string TName)
    {
        for (int i=0; i<n; i++)
        {
            foreach (Control C in GBox)
            {
                if (C.GetType().Name == "TextBox" || C.GetType().Name == "MaskedTextBox" || C.GetType().Name == "ComboBox")
                {
                    if (C.Name.IndexOf(TName)>-1)
                    {
                        C.Text = "";
                    }
                }
            }
        }
    }
    #endregion

    private void F_Find_Load(object sender, EventArgs e)
    {
        MyMC.CoPassData(Find_Folk, "tb_Folk"); //向“民族类别”列表框中添加信息
        MyMC.CoPassData(Find_Kultur, "tb_Kultur"); //向“文化程度”列表框中添加信息
        MyMC.CoPassData(Find_Visage, "tb_Visage"); //向“政治面貌”列表框中添加信息
        //向“职工类别”列表框中添加信息
        MyMC.CoPassData(Find_Employee, "tb_EmployeeGenre");
        MyMC.CoPassData(Find_Business, "tb_Business"); //向“职务类别”列表框中添加信息
        MyMC.CoPassData(Find_Laborage, "tb_Laborage"); //向“工资类别”列表框中添加信息
        MyMC.CoPassData(Find_Branch, "tb_Branch"); //向“部门类别”列表框中添加信息
        MyMC.CoPassData(Find_Duthcall, "tb_Duthcall"); //向“职称类别”列表框中添加信息
        //向下拉列表中添加省名
        MyMC.CityInfo(Find_BeAware, "select distinct beaware from tb_City", 0);
        //向下拉列表中添加市名
    }
}
```

```

MyMC.CityInfo(Find_School, "select distinct School from tb_Stuffbusic", 0);
//向下拉列表中添加主修专业
MyMC.CityInfo(Find_Speciality,
    "select distinct Speciality from tb_Stuffbusic", 0);
MyMC.MaskedTextBox_Format(Find1_WorkDate); //指定 MaskedTextBox 控件的格式
MyMC.MaskedTextBox_Format(Find2_WorkDate);
//根据 SQL 语句进行查询
MyDS_Grid = MyDataClass.getDataSet(Sut_SQL, "tb_Stuffbusic");
dataGridView1.DataSource = MyDS_Grid.Tables[0];
dataGridView1.AutoGenerateColumns = true;
}

private void Find_BeAware_TextChanged(object sender, EventArgs e)
{
    Find_City.Items.Clear();
    MyMC.CityInfo(Find_City, "select beaware,city from tb_City where beaware='"
        + Find_BeAware.Text.Trim() + "'", 1);
}

private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    ARsign = " AND ";
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    ARsign = " OR ";
}

private void button1_Click(object sender, EventArgs e)
{
    ModuleClass.MyModule.FindValue = ""; //清空存储查询语句的变量
    string Find_SQL = Sut_SQL; //存储显示数据表中所有信息的 SQL 语句
    //将指定控件集下的控件组合成查询条件
    MyMC.Find_Grids(groupBox1.Controls, "Find", ARsign);
    MyMC.Find_Grids(groupBox2.Controls, "Find", ARsign);
    //当合同的起始日期和结束日期不为空时
    if (MyMC.Date_Format(Find1_WorkDate.Text) != ""
        && MyMC.Date_Format(Find2_WorkDate.Text) != "")
    {
        if (ModuleClass.MyModule.FindValue != "") //如果 FindValue 字段不为空
            //用 ARsign 变量连接查询条件
            ModuleClass.MyModule.FindValue =
                ModuleClass.MyModule.FindValue + ARsign;
        //设置合同日期的查询条件
        ModuleClass.MyModule.FindValue = ModuleClass.MyModule.FindValue + " ("
            + "workdate>='" + Find1_WorkDate.Text
            + "' AND workdate<='" + Find2_WorkDate.Text + "')";
    }
    if (ModuleClass.MyModule.FindValue != "") //如果 FindValue 字段不为空
        //将查询条件添加到 SQL 语句的尾部
        Find_SQL = Find_SQL + " where " + ModuleClass.MyModule.FindValue;
    //按照指定的条件进行查询
    MyDS_Grid = MyDataClass.getDataSet(Find_SQL, "tb_Stuffbusic");
    //在 dataGridView1 控件是显示查询的结果
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    dataGridView1.AutoGenerateColumns = true;
    checkBox1.Checked = false;
}

```



```

private void Find1_WorkDate_Leave(object sender, EventArgs e)
{
    MyMC.Estimate_Date((MaskedTextBox)sender);
}

private void Find1_WorkDate_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, "", 0);
}

private void Find2_WorkDate_Leave(object sender, EventArgs e)
{
    bool TDate = MyMC.Estimate_Date((MaskedTextBox)sender);
    if (TDate == true)
        if (MyMC.Date_Format(Find1_WorkDate.Text) != ""
            && MyMC.Date_Format(Find2_WorkDate.Text) != "")
        {
            if (Convert.ToDateTime(Find2_WorkDate.Text)
                <= Convert.ToDateTime(Find1_WorkDate.Text))
                MessageBox.Show("当前日期必须大于它前一个日期。");
        }
}

private void Find2_WorkDate_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, "", 0);
}

private void Find_Age_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, "", 0);
}

private void Find_M_Pay_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, ((TextBox)sender).Text, 1);
}

private void Find_WorkLength_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, "", 0);
}

private void Find_Pact_Y_KeyPress(object sender, KeyPressEventArgs e)
{
    MyMC.Estimate_Key(e, "", 0);
}

private void checkBox1_Click(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
    {
        MyDS_Grid = MyDataClass.getDataSet(Sut_SQL, "tb_Stuffbusic");
        dataGridView1.DataSource = MyDS_Grid.Tables[0];
        dataGridView1.AutoGenerateColumns = true;
    }
}

private void button3_Click(object sender, EventArgs e)
{

```



```
this.Close();  
}  
  
private void button2_Click(object sender, EventArgs e)  
{  
    Clear_Box(7, groupBox1.Controls, "Find_");  
    Clear_Box(12, groupBox2.Controls, "Find");  
    Clear_Box(4, groupBox2.Controls, "Sign");  
}  
}
```

到此为止，本项目的核心功能全部介绍完毕。为节省本书的篇幅，其他模块的具体实现过程未详细给出，读者可参阅本书附带光盘中的源码和视频资料。



## 第4章 在线留言簿系统

高  
凡项目开发

当代社会中，各种网络应用随处可见，QQ、MSN、博客、微博、论坛……随着人们生活节奏的加快，网络交流已经成为司空见惯的事情，而作为交流手段之一的在线留言系统，更是深受人们的青睐。通过在线留言系统，身处世界各地的人们可以在线交流。

在本章的内容中，将讲解在线留言簿系统的运行流程，并通过具体的实例来讲解其实现过程。



### 赠送的超值电子书

- 031 使用 if 语句
- 032 使用 switch 语句
- 033 使用 while 语句
- 034 使用 do...while 语句
- 035 使用 for 语句
- 036 使用 break 语句
- 037 使用 continue 语句
- 038 使用 return 语句
- 039 使用 goto 语句
- 040 if 语句的嵌套



## 4.1 企业沟通之道

视频讲解 光盘：视频\第4章\企业沟通之道.avi

职场如战场，职场菜鸟们需要经过摸爬滚打的锻炼后，才能立于不败之地。

在职场生涯中，程序员们往往不只是生活在开发团队这个小圈子中，还需要与其他部门进行沟通。

### 4.1.1 开发公司的部门现状

对于大多数开发公司来说，整个公司通常分为财务部、市场部、销售部、产品部、开发部和培训部。

- **财务部：**负责公司财务、费用、预算决策和战略规划等工作。
- **销售部：**负责销售本公司的产品，拉客户来公司做项目。
- **产品部：**负责与客户沟通，了解客户的真实需求，将客户的需求传达给开发人员。
- **开发部：**负责程序开发，包括从规划到调试。
- **培训部：**负责内部员工培训，提升员工的业务能力。
- **市场部：**负责市场调研、市场宣传和产品包装等工作。

上述每个部门都有一个部门总监，直接向总经理负责。每个部门之间相互协调配合，使整个公司运转起来。

下面接着分析程序员所在的开发部，除了部门总监外，下面还有项目经理和程序员。具体结构如图4-1所示。



图 4-1 开发部的组织结构

其中项目经理都是高级软件工程师，每位项目经理下的程序员都实行弹性调动。即每当一个项目来临时，项目经理可以抽调任何程序员，组织自己的项目团队。

另外，每位程序员的任务也不一样。在一个开发团队中，有的人负责具体编码工作，有的人负责产品测试。

### 4.1.2 赢在职场——探讨部门之间的沟通之道

在开发公司中，与程序员所在的开发部打交道最多是产品部，请看下面的一个场景：

晚上8点多了，程序员小菜仍在公司噼里啪啦地敲着代码，此时，产品部的赵经理走过来：“小菜呀，我发现A页面上的XXX几个字很不美观，大小也不合适，你调整一下吧！”。

小菜忙打开 Visual Studio，经过长达7、8分钟的等待，终于将项目加载完毕了，小菜瞪着布满血丝的双眼，从几千个页面中，找到了赵经理所说的A页面，然后找到了他提到的那几个字，问赵经理说：“字改成多大的？”。

“多大的？就改大一点就行呀！”。小菜把原来的9pt更改成了10pt，Ctrl+S后，按下F5键，经过漫长的等待，页面终于出来了，赵经理看了看说：“还有点小，再大点！”，小菜只好又将10pt更改成了11pt。

赵经理端着杯子喝了口水说：“这次感觉大了点，再调小一点”，此时快8:30了，小菜和好朋友商量好8:30在超市门口见的，见赵经理没完没了地改，于是极不情愿地又将字体改成了10pt，按F5键，见赵经理还站在边上一个劲地端详呢，小菜拿起包，快速地按Windows+U键，关机闪人了，一边走，一边想：“什么人呀！事先没有界面设计，没有美工，程序员将界面做出来之后，对字体还调来调去，没完没了！”

上述案例只是修改字体大小，工作量还算简单。如果产品部在设计产品伊始便出现问题，而开发部已经根据当初的规划完成了整个开发工作，那么，如果客户此时不满意，开发部将面临整个项目需要修改架构重新开发的境地，这将给公司带来不可估量的损失。

部门与部门之间的沟通的确比较麻烦，有时虽然明知很多问题的根源，但因为身处不同的部门，却没有力气去解决。

例如在上述案例中，产品部觉得稍微调整一下是十分简单的，此时的开发人员应该有一些容人之量，然后主动和对方部门进行沟通，做交流，尽量本着解决问题的态度来做事。

但是开发人员总会觉得这样改来改去体现了产品部门对研发人员工作成果的不尊重。站在开发者的角度，最不希望的就是干了活却因为产品部的原因被毙。

当发生类似情况时，应该按照如下规则进行处理。

(1) 产品部：首先要解决的问题就是学会如何尊重开发者的劳动，认识到开发人员不是实现自己毫无边际遐想的工具。如果不认识到这点，工作也很难进展。

(2) 开发部：在开发过程中，如果一直抱怨这个不公、那个不平，却不思考解决这些问题的方法，也是有不妥之处的。当程序员因为情绪不满影响到工作动力和态度的时候，应该及时找到问题所在，并加以解决。当出现沟通问题后，研发人员应该将出现问题的地方及时反馈给项目经理，让他去帮助大家来解决此类问题，毕竟抱怨是没有用的。

(3) 企业运维：产品部和开发部之间的沟通不要仅仅局限于口头，建议把与项目有关的沟通全部实现文字化，将需求和实现统一为纸质文字化操作，并经过部门经理签字盖章。这样，每个部门发出的所有指令都具有很强的效力，能够避免出现问题时的相互扯皮问题。

## 4.2 第一个盈利项目

视频讲解 光盘：视频\第4章\第一个盈利项目.avi

本项目的客户是一家在线零售店的店主，要求开发一个“在线留言簿系统”。

### 4.2.1 组建团队

整个项目开发团队成员的具体职责如下所示。

- 软件工程师 A：负责实现项目规划，撰写系统设计规划书。
- 软件工程师 B：负责文件概览工作。
- 软件工程师 C：负责搭建并设计数据库。
- 软件工程师 D：负责留言系统的具体编码工作。
- 软件工程师 E：负责系统整体框架设计，协调项目中各个模块的进展，并设计数据库访问层。

本项目的具体实现流程如图 4-2 所示。

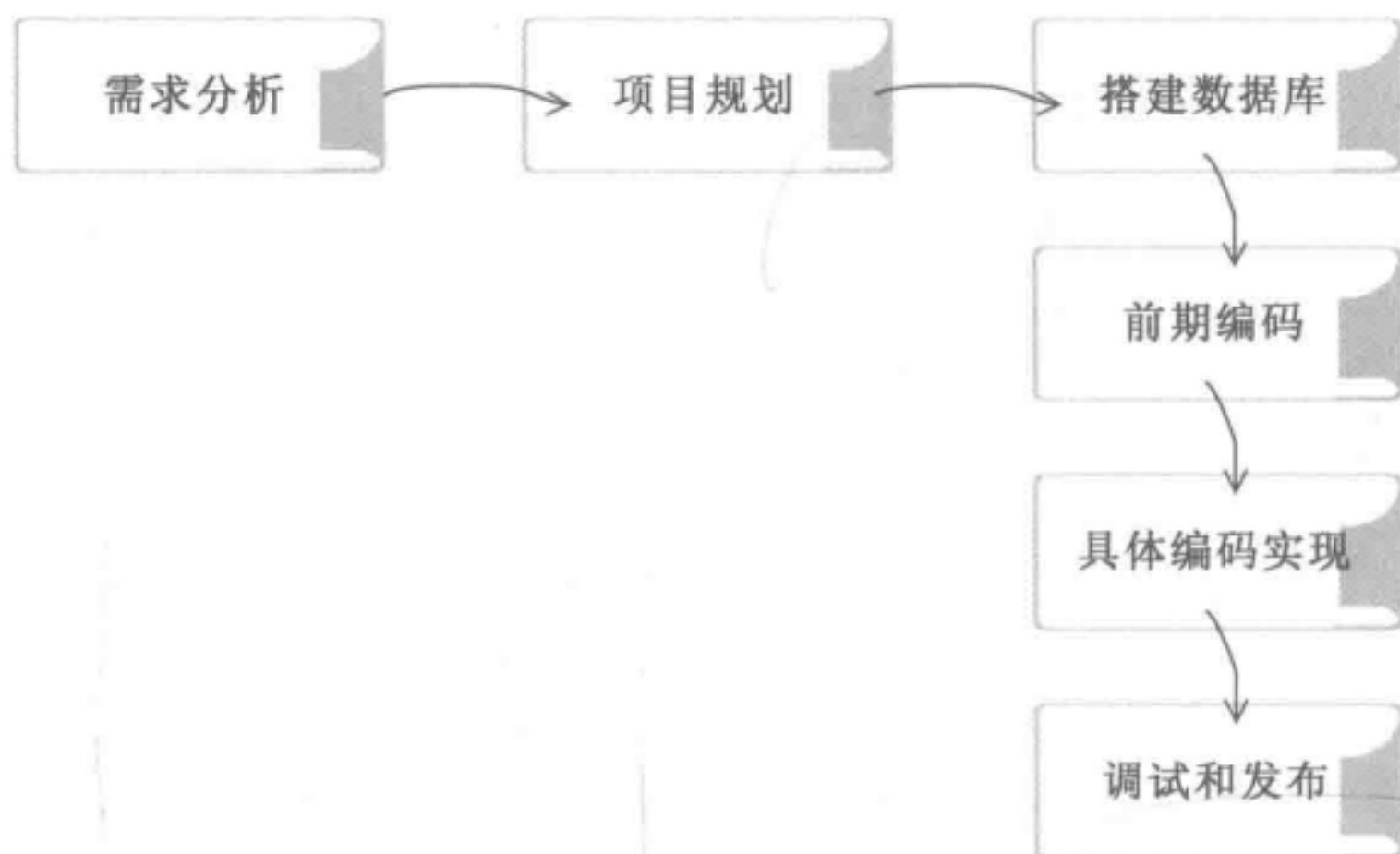


图 4-2 开发流程

### 4.2.2 系统规划

系统规划是一个项目的基础，是任何项目的第一步工作。很多初学者在做项目时，往往看到功能之后，就忙不迭地要新建工程，并开始编码了，结果却总是忽略了这样或那样的要求，最后需要经过多次修改，造成了欲速则不达的后果。

所以，针对本项目，我们吸取了以前的教训，在前期阶段一丝不苟地分析。首先召开了一个动员会议，并为项目的进展做一个简单的规划。大家一致认为：本项目包括后台数据库的建立、维护以及前端应用程序的开发两个方面。应用程序的开发采用目前比较流行的 ADO 数据库访问技术，并将每个数据库表的字段和操作封装到相应的类中，使应用程序



的各个窗体都能够共享对表的操作，而不需要重复编码，使程序更加易于维护，从而能将面向对象的程序设计思想成功地应用于应用程序设计中，这也是本系统的优势和特色，具体实现流程如图 4-3 所示。

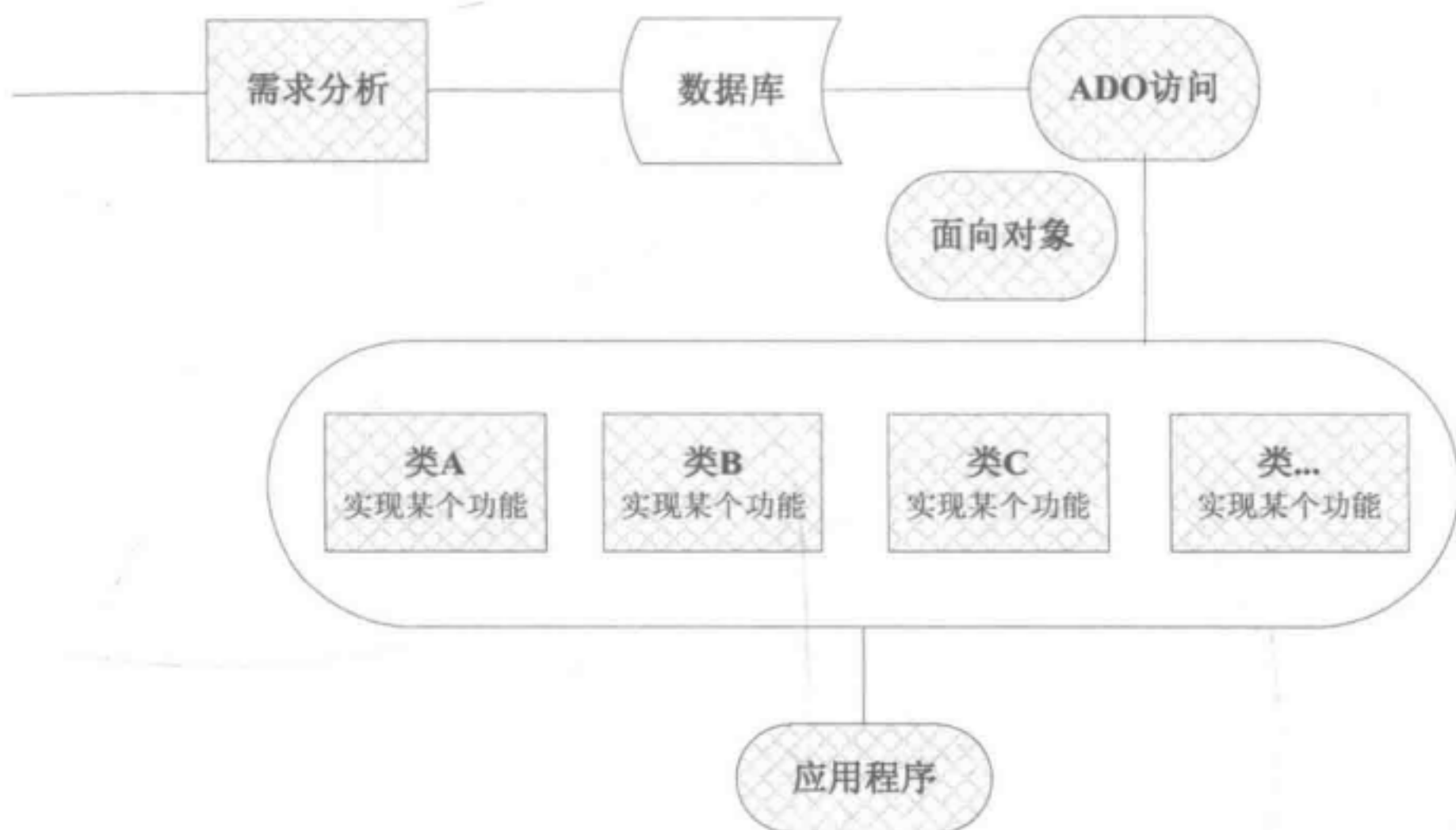


图 4-3 实现流程

要开发一个在线留言簿系统，首先需要进行系统需求分析和总体设计，分析系统的使用对象和用户需求，设计系统的体系结构和数据库表结构，决定使用的开发工具和后台数据库，规划项目的开发进度等。

## 4.3 系统概述和总体设计

视频讲解 光盘：视频\第4章\系统概述和总体设计.avi

本项目的规划书分为两部分，分别是在在线留言簿模块功能原理和在线留言簿系统构成模块。按照预先分配的任务，由软件工程师 A 负责系统概述和总体设计，这是我们整个项目的第一步。

### 4.3.1 在线留言簿模块的功能原理

Web 站点的在线留言簿系统的实现原理比较清晰明了，其主要操作是对数据库中的数据进行添加和删除操作。在实现的过程中，往往是根据系统的需求而进行不同功能模块的设置。在线留言簿模块的必备功能如下：

- (1) 提供信息发布表单，供用户发布新的留言。
- (2) 将用户发布的留言添加到系统库中。
- (3) 在页面内显示系统库中的留言数据。
- (4) 对某条留言数据进行在线回复。
- (5) 删除系统内不需要的留言。

## 4.3.2 在线留言簿系统的构成模块

一个典型的在线留言簿系统的构成模块如下。

- 信息发表模块：用户可以在系统上发布新的留言信息。
- 信息显示模块：用户发布的留言信息能够在系统上显示。
- 留言回复模块：可以对用户发布的留言进行回复，以实现相互间的交互。
- 系统管理模块：站点管理员能够对发布的信息进行管理和控制。

上述应用模块的具体运行流程如图 4-4 所示。

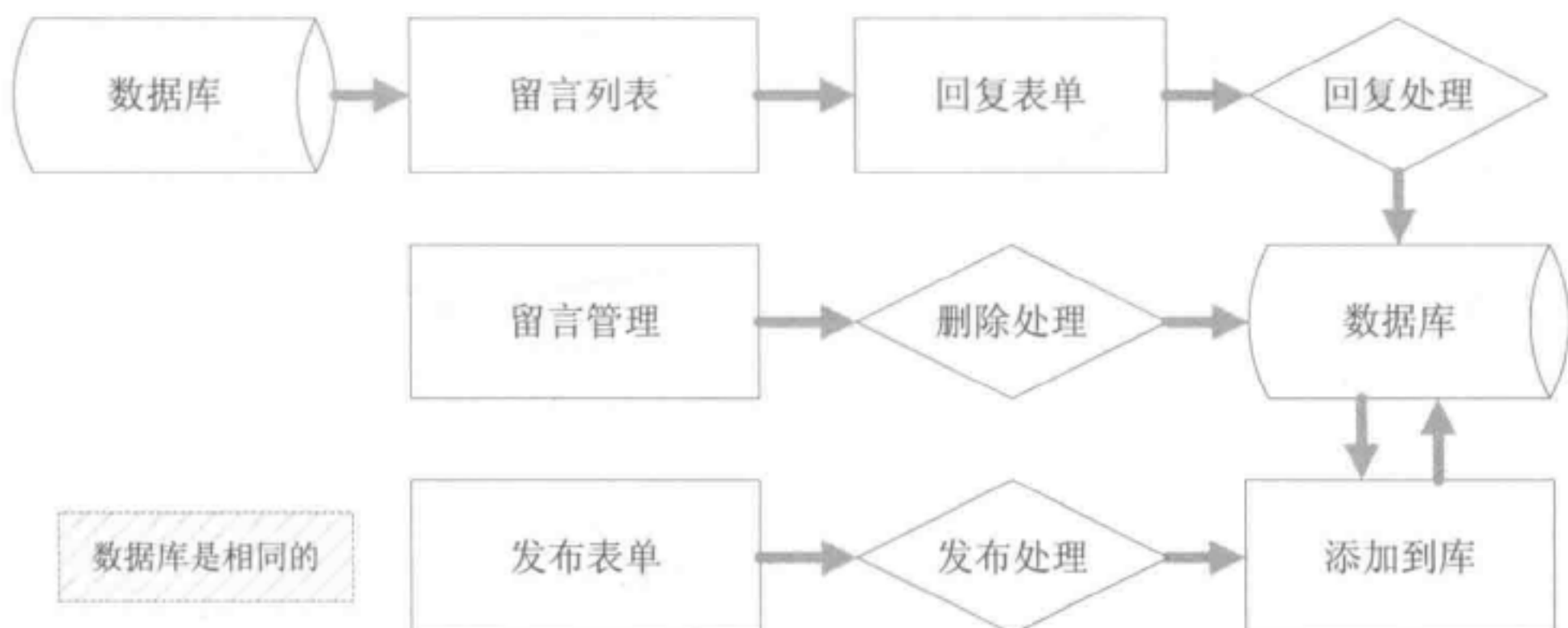


图 4-4 在线留言簿系统的运行流程

## 4.4 规划系统文件

视频讲解 光盘：视频\第 4 章\规划系统文件.avi

通过前面的介绍，初步了解了在线留言簿模块的原理和具体的运行流程。在接下来的内容中，可以根据各构成功能模块来规划系统实现文件。

### 4.4.1 规划文件

要想开发在线留言簿系统，需要先了解留言簿系统的功能需求。读者可以通过在网络上光顾一些留言簿站点，并尝试发表多个留言信息的方式来了解留言簿系统的功能需求。然后，根据总结的模块功能和规划的结构图，可以规划出了整个项目的实现文件。

具体说明如下。

- 系统配置文件：功能是对项目程序进行总体配置。
- 样式设置模块：功能是设置系统文件的显示样式。
- 数据库文件：功能是搭建系统数据库平台，保存系统的登录数据。
- 留言簿列表文件：功能是将系统内的留言信息以列表样式显示出来。
- 发布留言模块：功能是向系统内添加新的留言数据。
- 留言管理页面：功能是删除系统内部需要的留言数据。

### 4.4.2 选择开发工具

本项目使用 Visual Studio 2013 作为开发工具,预期各个文件在 Visual Studio 2013 解决方案管理器中的效果如图 4-5 所示。



图 4-5 预期规划的项目文件

## 4.5 设计数据库

视频讲解 光盘: 视频\第4章\设计数据库.avi

本项目系统的开发主要包括后台数据库的建立、维护以及前端应用程序的开发两个方面。数据库设计是在线留言系统设计开发的一个重要组成部分。

### 4.5.1 后台数据库及数据库访问接口的选择

数据库设计工作在软件工程中的地位十分重要,在开发数据库管理信息系统时,需要选择后台数据库和相应的数据库访问接口。后台数据库的选择需要考虑用户需求、系统功能和性能要求等因素。考虑到系统所要管理的数据量比较大,且需要多用户同时运行访问,本项目将使用 SQL Server 2008 作为后台数据库管理平台。

### 4.5.2 数据库结构的设计

具体的数据库设计需要参考系统需求分析,由需求分析的规划可知,整个项目对象有两种信息,所以对应的数据库也需要包含这两种信息,从而系统需要包含两个数据库表。

- Message: 留言信息表。
- Reply: 回复信息表。



软件工程师 B 给出了具体数据库表的书面文件。

在 SQL Server 2008 中创建一个名为“Liuyan”的数据库，并新建两个表：Message 和 Reply。

(1) 表 Message 用于保存留言信息，具体设计结构如表 4-1 所示。

表 4-1 Message 信息表的结构

| 字段名称       | 数据类型         | 是否主键 | 默认值  | 功能描述  |
|------------|--------------|------|------|-------|
| ID         | int          | 是    | 递增 1 | 编号    |
| Title      | varchar(200) | 否    | Null | 标题    |
| Message    | text         | 否    | Null | 内容    |
| CreateDate | datetime     | 否    | Null | 时间    |
| IP         | varchar(20)  | 否    | Null | IP 地址 |
| Email      | varchar(250) | 否    | Null | 邮箱    |
| Status     | tinyint      | 否    | 0    | 状态    |

(2) 表 Reply 用于保存留言回复信息，具体设计结构如表 4-2 所示。

表 4-2 Reply 信息表的结构

| 字段名称       | 数据类型          | 是否主键 | 默认值  | 功能描述  |
|------------|---------------|------|------|-------|
| ID         | int           | 是    | 递增 1 | 编号    |
| Reply      | varchar(1000) | 否    | Null | 内容    |
| CreateDate | datetime      | 否    | Null | 时间    |
| IP         | varchar(20)   | 否    | Null | IP 地址 |
| MessageID  | int           | 否    | Null | 留言编号  |

## 4.6 系统配置和数据库访问层

视频讲解 光盘：视频\第 4 章\系统配置和数据库访问层.avi

系统配置和数据库访问层的工作由作者来完成，此步骤是整个项目的基础，项目中的具体功能将以此为基础进行扩展。

由于深知此步骤的重要性，所以一直通宵达旦，尽力寻求最优方案。

### 4.6.1 系统配置

经过总体构成功能的分析后，接下来就可以根据各构成功能模块做实质性的工作了。具体有如下两个工作：

- 新建网站项目。
- 实现。

## 1. 新建网站项目

实质性工作的第一步是创建一个 Visual Studio 2013 项目，流程如下。

(1) 打开 Visual Studio 2013，选择“文件”→“项目”→“网站”菜单命令，在弹出的“新建网站”对话框中创建一个名为“Liuyan”的网站项目，如图 4-6 所示。

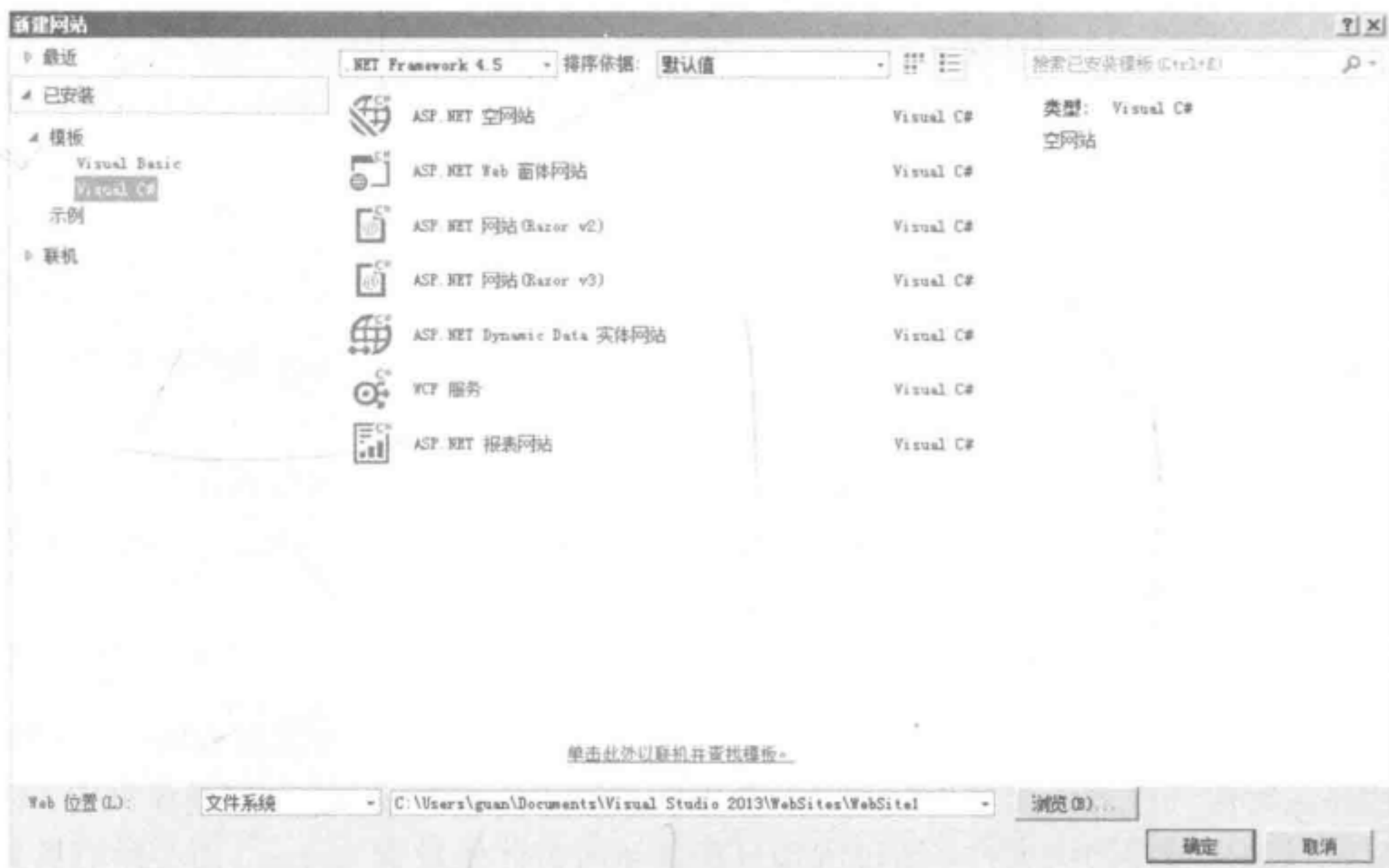


图 4-6 新建网站项目

(2) 然后根据 4.4.1 小节中的规划文件，分别创建对应的程序文件，并分别命名。创建完毕后的效果如图 4-7 所示。



图 4-7 Visual Studio 2013 解决方案中的程序文件

## 2. 配置系统文件

这个在线留言簿项目是由几个程序文件实现的。要想让这些程序成功运行，需要一个前提：配置系统文件。

在以 Visual Studio 2013 开发 ASP.NET 程序时，系统配置文件是 Web.config，其主要功能是设置数据库的连接参数，并配置系统与 Ajax 服务器的相关内容。

### (1) 配置连接字符串参数

配置连接字符串参数即设置系统程序连接数据库的参数，对应的实现代码如下所示：

```
<connectionStrings>
  <add name="SQLCONNECTIONSTRING" connectionString="data source=GUAN\AAA;
    user id=sa;pwd=8888888;database=liuyan" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

其中，source 设置连接的数据库服务器；user id 和 pwd 分别指定数据库的登录名和密码；database 设置连接数据库的名称。

Web.config 文件是 ASP.NET 的基本文件，通常用于存储系统的公用信息，数据库的连接语句就在里面建立。而上述代码是通用的 ASP.NET 配置代码，但是在 ASP.NET 代码调试时，需要加入如下调试代码：

```
<compilation defaultLanguage="c#" debug="true" />
```

设置 compilation debug="true" 后就启用了 ASPX 调试。如果设置为 false，将会提高此应用程序运行时的性能。设置为 true 后，会把调试符号(.pdb 信息)插入到编译页中。因为这将创建执行起来较慢的大文件，所以应该只在调试时将此值设置为 true，而在所有其他时候都设置为 false。

### (2) 配置 Ajax 服务器参数

配置 Ajax 服务器参数即配置 Ajax Control Toolkit 程序集参数，这样，系统页面在引用 AjaxControlToolkit.dll 中的控件时，不需要额外添加<Register>代码。上述功能在<controls>元素内的对应实现代码如下所示：

```
<pages>
  <controls>
    <add namespace="AjaxControlToolkit" assembly="AjaxControlToolkit"
      tagPrefix="ajaxToolkit"/>
    <add tagPrefix="asp" namespace="System.Web.UI"
      assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
      PublicKeyToken=31bf31056ad364e35" />
  </controls>
</pages>
```

在 ASP.NET Web 项目中，文件 Web.config 十分重要。在 ASP.NET 中，资源的配置信息包含在一组配置文件中，每个文件都被命名为 Web.config。每个配置文件都包含 XML 标记和子标记的嵌套层次结构，这些标记带有指定配置设置的属性。因为这些标记必须是格式正确的 XML，所以标记、子标记和属性是区分大小写的。标记名和属性名是 Camel 大小写形式的，这意味着标记名的第一个字符是小写的，任何后面连接单词的第一个字母是大写的。属性值是 Pascal 大小写形式的，这意味着第一个字符是大写的，任何后面连接单词的第一个字母也是大写的。true 和 false 例外，它们总是小写的。总结完毕之后，决定早点



休息，为接下来的数据库访问层设计做好准备。

## 4.6.2 数据库访问层设计

很多初学者虽然已经多次使用了数据访问层，但是其具体过程还是不很了解。其实，数据访问层有时候也称为持久层，其功能主要是负责数据库的访问。简单的说法就是实现对数据表的 Select、Insert、Update、Delete 操作。如果要加入 ORM 的元素，那么就会包括对象和数据表之间的映射，以及对象实体的持久化。

在本项目中，文件 lei.cs 的功能是实现应用程序的数据库访问层，在 ASPNETAJAXWeb.AjaxLeaveword 空间内建立 Message 类，并实现对系统库中数据的处理。上述功能的实现流程如图 4-8 所示。

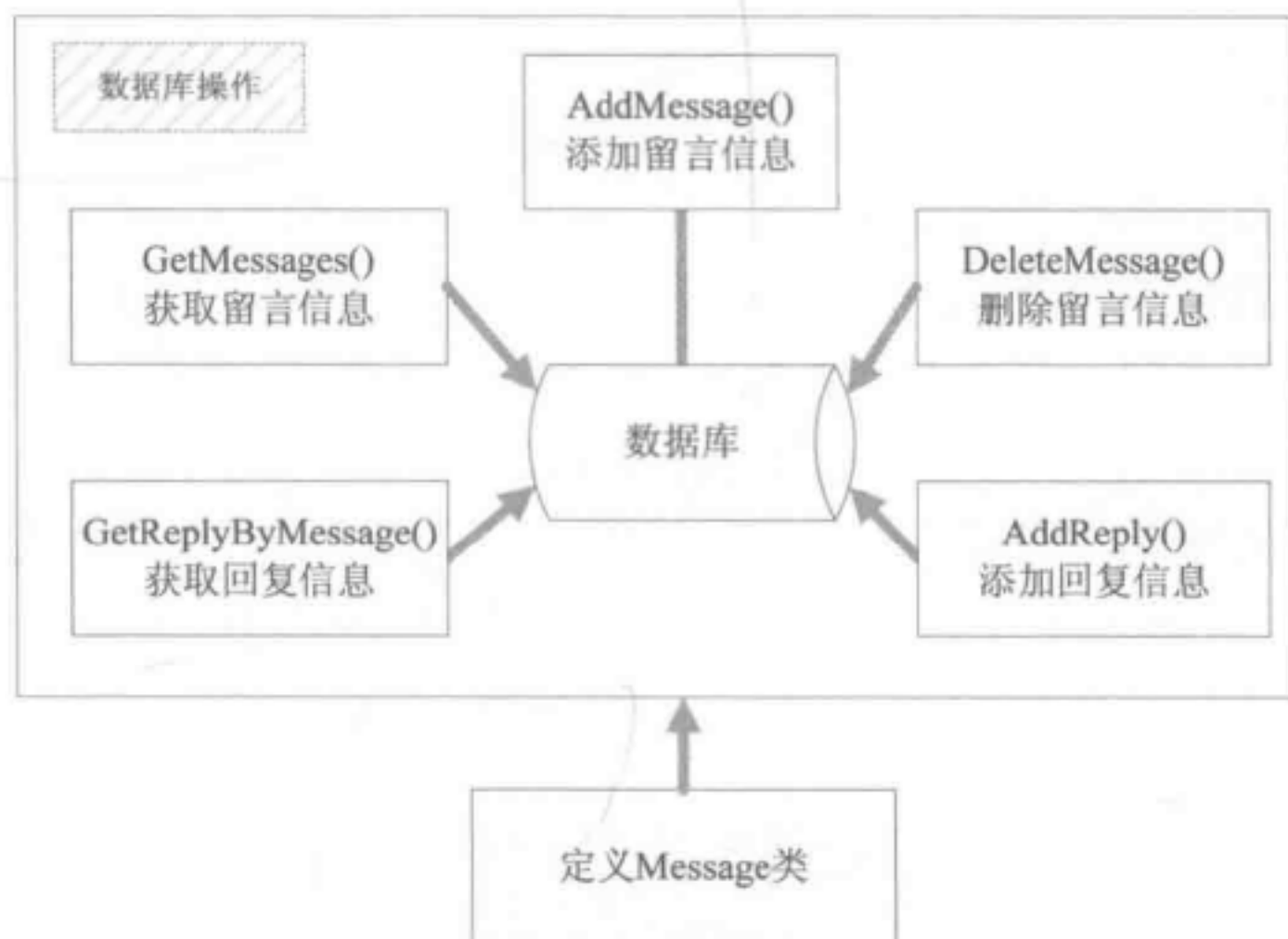


图 4-8 数据访问层的实现流程

数据访问层是.NET 框架的核心知识，建立合理的数据库访问层结构，可以提高系统的效率，并为后期维护带来极大的方便。通过实现数据的访问模式，达到对物理数据库中的表、视图等的访问。应用程序对数据库的访问有如下 3 种方式。

- 事务脚本：存储过程。
- ORM：对象-关系映射。
- 表模型：以物理数据表为基本单位进行访问，类似于.NET 中的 DataTable。

我们认为，在.NET 中，第三种方式更为容易实现。因为表和视图有很多相似点，不同的是，视图是只读的。通过表模型可以很好地解决实现和表现的结合，并可解决效率问题，这在大型站点中是十分重要的。

下面来看文件 lei.cs 的具体实现流程。

### 1. 定义 Message 类

定义 Message 类的实现代码如下所示：

```
using System;  
using System.Data;  
using System.Configuration;
```

```
using System.Data.SqlClient;
namespace ASPNETAJAXWeb.AjaxLeaveword
{
    public class Message
    {
        public Message()
        {
            ...
        }
    }
}
```

## 2. 获取系统内的留言信息

获取系统内的留言信息就是获取系统库内已存在的留言信息，这个功能是由方法 GetMessages()实现的。该方法的具体实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数，将其保存在 connectionString 字符串中。

(2) 使用该连接字符串创建 con 对象，实现数据库连接。

(3) 新建获取数据库留言数据的 SQL 查询语句。

(4) 创建获取数据的对象 da。

(5) 打开数据库连接，获取查询数据。

(6) 将获取的查询结果保存在 ds 中，并返回 ds。

上述功能的对应实现代码如下所示：

```
public DataSet GetMessages()
{
    ///获取连接字符串
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT * FROM Message Order by CreateDate DESC";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        ///打开连接
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch(Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
        con.Close();
    }
    return ds;
}
```

数据库的设计真的很重要，因为几乎所有的动态 Web 站点的内容都是基于数据库数据的，所以，对数据库的操作应该充分考虑效率问题。在此，给读者提出一个小小的建议，建议读者充分利用所在机器内存中缓存的 ADO 对象。

### 3. 添加系统留言信息

所谓添加系统留言信息，就是将新发布的留言信息添加到系统数据库中，此功能是由方法 AddMessage(string title, string message, string ip, string email)实现的。该方法的具体实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数，并将其保存在 connectionString 字符串中。

(2) 使用该连接字符串创建 con 对象，实现数据库连接。

(3) 使用 SQL 添加语句，然后创建 cmd 对象，准备进行插入操作。

(4) 打开数据库连接，执行新数据插入操作。

(5) 将数据插入操作所涉及的行数保存在 result 中。

(6) 插入成功，则返回 result 值，失败则返回-1。

上述功能的对应实现代码如下所示：

```
public int AddMessage(string title, string message, string ip, string email)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "INSERT INTO Message (Title, Message, IP, Email, CreateDate, Status)
        VALUES (@Title, @Message, @IP, @Email, GETDATE(), 0)";
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@Title", SqlDbType.VarChar, 200);
    cmd.Parameters.Add("@Message", SqlDbType.Text);
    cmd.Parameters.Add("@IP", SqlDbType.VarChar, 20);
    cmd.Parameters.Add("@Email", SqlDbType.VarChar, 255);
    cmd.Parameters[0].Value = title;
    cmd.Parameters[1].Value = message;
    cmd.Parameters[2].Value = ip;
    cmd.Parameters[3].Value = email;
    int result = -1;
    try
    {
        ///打开连接
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
        con.Close();
    }
}
```



```
    return result;  
}
```

#### 4. 删除系统留言信息

所谓删除系统留言信息，就是把系统内存在的留言数据从系统数据库中删除，此功能是由方法 `DeleteMessage(int messageID)` 实现的。具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 字符串中。

(2) 使用该连接字符串创建 `con` 对象，实现数据库连接。

(3) 使用 SQL 删除语句，然后创建 `cmd` 对象，准备删除操作。

(4) 打开数据库连接，执行新数据删除操作。

(5) 将数据删除操作所涉及的行数保存在 `result` 中。

(6) 删除成功，则返回 `result` 值，失败则返回 -1。

上述功能的对应实现代码如下所示：

```
public int DeleteMessage(int messageID)  
{  
    string connectionString =  
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;  
    SqlConnection con = new SqlConnection(connectionString);  
    ///创建 SQL 语句  
    string cmdText = "DELETE Message WHERE ID = @ID";  
    SqlCommand cmd = new SqlCommand(cmdText, con);  
    ///创建参数并赋值  
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);  
    cmd.Parameters[0].Value = messageID;  
    int result = -1;  
    try  
    {  
        ///打开连接  
        con.Open();  
        ///操作数据  
        result = cmd.ExecuteNonQuery();  
    }  
    catch (Exception ex)  
    {  
        ///抛出异常  
        throw new Exception(ex.Message, ex);  
    }  
    finally  
    {  
        ///关闭连接  
        con.Close();  
    }  
    return result;  
}
```

#### 5. 获取系统内的留言回复信息

获取系统内的留言回复信息，就是查询系统数据库内用户对留言的回复信息数据，此功能是由方法 `GetReplyByMessage(int messageID)` 实现的。具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString`

字符串中。

- (2) 使用该连接字符串创建 con 对象，实现数据库连接。
- (3) 新建查询数据库留言回复数据的 SQL 查询语句。
- (4) 创建获取数据的对象 da。
- (5) 打开数据库连接，获取查询数据。
- (6) 将获取的查询结果保存在 ds 中，并返回 ds。

上述功能的对应实现代码如下所示：

```
public DataSet GetReplyByMessage(int messageID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText =
        "SELECT * FROM Reply WHERE MessageID = @MessageID Order by CreateDate DESC";
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@MessageID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = messageID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
        con.Close();
    }
    return ds;
}
```

## 6. 添加留言回复信息

添加留言回复信息，就是把新发布的留言回复信息添加到系统数据库中，此功能是由方法 AddReply(string message, string ip, int messageID)实现的。具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数，并将其保存在 connectionString 字符串中。
- (2) 使用该连接字符串创建 con 对象，实现数据库连接。
- (3) 使用 SQL 添加语句，然后创建 cmd 对象，准备插入操作。
- (4) 打开数据库连接，执行新数据插入操作。
- (5) 将数据插入操作所涉及的行数保存在 result 中。
- (6) 插入成功，则返回 result 值，失败则返回-1。

上述功能的对应实现代码如下所示：

```
public int AddReply(string message, string ip, int messageID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    string cmdText = "INSERT INTO Reply(Reply,IP,CreateDate,MessageID)
        VALUES(@Reply,@IP,GETDATE(),@MessageID)";
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@Reply", SqlDbType.VarChar, 1000);
    cmd.Parameters.Add("@Ip", SqlDbType.VarChar, 20);
    cmd.Parameters.Add("@MessageID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = message;
    cmd.Parameters[1].Value = ip;
    cmd.Parameters[2].Value = messageID;
    int result = -1;
    try
    {
        ///打开连接
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch(Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
        con.Close();
    }
    return result;
}
```

在上述各处理方法中，使用了 SQL 的查询、添加和删除语句，对系统数据库内的数据进行了操作处理。

在现实 Web 应用系统中，各类应用的数据库相关操作都是基于上述 3 种操作的。SQL 语句是数据库技术的核心知识之一，读者可以在百度中通过检索“SQL 教程”关键字来获取其相关知识。

到此为止，完成了数据访问层的编码工作。

在 C# 程序中，经常会涉及一些存储在数据库中的常用信息。这些信息对于每一个访问用户都是相同的。若每一个用户访问时，都要去数据库里取出来，然后显示给用户，会加重数据库服务器负载，使之无法快速服务于更重要的事务处理。而且 Web 服务器也必须不停地创建 ADO 对象，从而消耗大量资源，导致当用户很多时几乎失去响应。如果能把一些常用信息事先存储在内存中，当用户访问时，直接从内存中取出，显示给用户，则可以大大减小系统的压力，提高响应速度。

在具体应用时，可以把已经取得了数据的 RecordSet 对象存储在 Application 变量中。当用户访问时，从 Application 变量中取得 RecordSet 对象，而不需再次建立数据库连接。也可以将 RecordSet 对象里的数据存储在数组中，然后再将数组存储在 Application 变量中，使用时，以数组的方式读取。



## 4.7 具体编码

视频讲解  光盘：视频\第4章\具体编码.avi

因为在系统框架设计中已经编写好了共用类，完成了数据访问层的设计，所以，接下来编码工作的思路就变得十分清晰了。在编码过程中，只需在已经编写类的基础上进行扩充，即可完成整个编码工作。

### 4.7.1 留言数据显示

留言数据显示模块的功能是，将系统库内的留言信息以列表的样式显示出来，并提供新留言发布表单，将发表的数据添加到系统数据库中。上述功能的实现文件如下：

- Index.aspx。
- Index.aspx.cs。
- Yanzhengma.aspx。
- AjaxService.cs。

#### 1. 留言列表显示页面

文件 Index.aspx 的功能是，插入专用控件，将系统内的数据读取并显示出来，然后提供发布表单，供用户发布新留言。其具体实现流程如下。

- (1) 插入 1 个 GridView 控件，以列表样式显示数据库内的数据。
- (2) 在表格内显示各留言的数据内容。
- (3) 添加 3 个链接，供留言发布、留言回复和留言管理操作。
- (4) 调用 Ajax 程序集内的 DynamicPopulate 控件，实现动态显示留言回复内容。

文件 Index.aspx 中的主要实现代码如下所示：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Index.aspx.cs"
StyleSheetTheme="css" Inherits="Board" %>
...
<form id="form1" runat="server">
    <asp:ScriptManager ID="sm" runat="server" >
        <Services>
            <asp:ServiceReference Path="AjaxService.asmx" />
        </Services>
    </asp:ScriptManager>
    <table class="Table" border="0" cellpadding="0" cellspacing="0" align="center">
        <tr><td colspan="2">
            <asp:UpdatePanel runat="server" ID="up">
                <ContentTemplate>
                    <asp:GridView ID="gvMessage" runat="server" Width="100%"
                        AutoGenerateColumns="False" SkinID="mm" ShowHeader="False">
                        <Columns>
                            <asp:TemplateField>
                                <ItemTemplate>
                                    <table align="center" cellpadding="3" cellspacing="0" class="Table">
                                        <tr>
                                            <td>作者: <a href='mailto:<%# Eval("Email") %>'><%# Eval("Email") %></a>
```





```
</ajaxToolkit:TextBoxWatermarkExtender>
</td>
</tr>
|  |
| --- |
|IP 地址:
  |
|电子邮件:
 </asp:RequiredFieldValidator>         <asp:RegularExpressionValidator ID="revEmail" runat="server" ControlToValidate="tbEmail" Display="None" ErrorMessage="电子邮件格式不正确，请输入如下形式的电子邮件：<br />mnmn@nnn.com" ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\\w+([-.]\\w+)*"></asp:RegularExpressionValidator>         <ajaxToolkit:TextBoxWatermarkExtender ID="wmeEmail" runat="server" TargetControlID="tbEmail" WatermarkText="请输入电子邮件" WatermarkCssClass="Watermark"></ajaxToolkit:TextBoxWatermarkExtender>         <ajaxToolkit:ValidatorCalloutExtender ID="vceEmail" runat="server" TargetControlID="revEmail" HighlightCssClass="Validator"></ajaxToolkit:ValidatorCalloutExtender> |
|留言内容:
  |
|验证码:
 <asp:Label ID="lbMessage" runat="server" ForeColor="red" CssClass="Text"></asp:Label> |
|  |

```



[illegible]

### 3. 调用验证码文件

文件 Yanzhengma.aspx 的功能是调用 bin 目录内的 ASPNETAJAXWeb.ValidateCode.dll 控件，实现验证码显示效果。验证码文件 Yanzhengma.aspx 的具体实现代码如下所示：

```
<%@ Page Language="C#" AutoEventWireup="false"
    Inherits="ASPNETAJAXWeb.ValidateCode.Page.ValidateCode" %>
```

#### 4. 留言展开回复模块

留言展开回复模块的功能是，当单击某留言后的“展开”链接后，将动态显示此留言的回复数据。具体实现流程如下。

- (1) 调用 Ajax 的 DynamicPopulate 控件, 用于实现动态显示效果。
- (2) 调用文件 AjaxService.cs 内的 GetReplyByMessage 方法, 获取回复内容。

文件 AjaxService.cs 的具体实现代码如下所示:

```

///开始引入新的命名空间
using System.Data;
using System.Text;
using System.Web.Script.Services;
using ASPNETAJAXWeb.AjaxLeaveword;
/// AjaxService 的摘要说明
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
///添加脚本服务
[System.Web.Script.Services.ScriptService()]
public class AjaxService : System.Web.Services.WebService {
    public AjaxService ()
    { }
    [WebMethod]
    public string GetReplyByMessage(string contextKey)
    {
        ///获取参数 ID
        int messageID = -1;
    }
}

```

```

if(Int34.TryParse(contextKey,out messageID) == false)
{
    return string.Empty;
}
Message message = new Message();
DataSet ds = message.GetReplyByMessage(messageID);
if(ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0)
{
    return string.Empty;
}
StringBuilder returnHtml = new StringBuilder();
foreach(DataRow row in ds.Tables[0].Rows)
{
    returnHtml.AppendFormat("<div>{0}于[{1}] 回复</div>",
        row["IP"], row["CreateDate"]);
    returnHtml.Append("<br />");
    returnHtml.AppendFormat("<div>{0}</div>", row["Reply"]);
    returnHtml.Append("<br />");
}
return returnHtml.ToString();
}
}

```

通过上述代码处理，执行系统留言列表页面后，将首先默认显示留言数据，而不显示留言的回复数据。单击某留言后的“展开”链接后，此留言的回复信息将动态地显示。

在上面的留言回复处理过程中，通过 foreach 语句对于内容进行了 HTML 化处理，因为只有处理后，才能使回复内容以浏览者希望的格式显示。

但是这里有一个问题，对于初学者来说，在代码中添加 HTML 转换代码变得十分复杂。不但在视觉上使程序员感觉到繁琐，而且在后期维护上也会感到无所适从，并且也不能保证所有的特殊字符都能被成功转换。

其实，在网络中有专门处理 HTML 标记的工具，例如 HtmlArea。HtmlArea 是一款很简洁的 WYSIWYG(所见即所得)编辑器，是一个纯 JavaScript+HTML 的编辑器，理论上可以套在任何语言平台上，经过实际使用，可以与 ASP.NET 4.5 + Ajax 很好地结合。

无论是留言系统，还是新闻系统，只要涉及了信息发布和维护的项目，都可以使用现成的文本编辑器。市面上免费的文本编辑器比较多，并且使用方法简单，功能强大，是提高我们开发效率的重要工具，建议广大读者多使用第三方的文本编辑器。

## 4.7.2 留言分页列表显示模块

一页网页的容量是有限的，并且为了方便，用户浏览留言簿系统的留言内容，不可能将很多条留言信息显示在一个网页上，所以需要使用分页技术。留言分页列表显示模块的功能是，将系统数据库内的留言信息以分页列表的样式显示出来。

上述功能的实现文件如下：

- LeavewordFen.aspx。
- LeavewordFen.aspx.cs。

### 1. 留言分页显示页面

留言分页显示页面文件 LeavewordFen.aspx 的功能是，插入专用控件，将系统内的数据



读出来,然后将获取的留言数据以分页样式显示。其具体实现流程如下。

- (1) 插入 1 个 GridView 控件,用于以列表样式显示留言的信息。包括留言者、邮箱地址、时间和留言内容等。
- (2) 通过 GridView 控件设置分页显示留言数为 5。
- (3) 通过 GridView 控件设置分页处理事件为 gvMessage\_PageIndexChanging。
- (4) 通过 PagerSettings 设置分页模式为 NumericFirstLast。

文件 LeavewordFen.aspx 的主要代码如下所示:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LeavewordFen.aspx.cs"
    StylesheetTheme="css" Inherits="BoardPaging" %>
...
<form id="form1" runat="server">
    <asp:ScriptManager ID="sm" runat="server" />
    <table class="Table" border="0" cellpadding="0" cellspacing="0" align="center">
        <asp:UpdatePanel runat="server" ID="up">
            <ContentTemplate>
                <asp:GridView ID="gvMessage" runat="server" Width="100%"
                    AutoGenerateColumns="False" SkinID="mm" ShowHeader="False"
                    AllowPaging="True" OnPageIndexChanging="gvMessage_PageIndexChanging"
                    PageSize="5">
                    <Columns>
                        <asp:TemplateField>
                            <ItemTemplate>
                                <table class="Table" cellpadding="3" cellspacing="0">
                                    <tr>
                                        <td>作者: <a href='mailto:<## Eval("Email") %>'><## Eval("Email") %></a>
                                        于[<## Eval("IP") %>]、[<## Eval("CreateDate") %>] 留言</td>
                                    </tr>
                                    <tr><td><hr size="1" /></td></tr>
                                    <tr><td class="Title"> <## Eval("Title") %></td></tr>
                                    <tr><td><## Eval("Message") %></td></tr>
                                </table>
                            </ItemTemplate>
                        </asp:TemplateField>
                    </Columns>
                    <PagerSettings Mode="NumericFirstLast" />
                </asp:GridView>
            </ContentTemplate>
        </asp:UpdatePanel>
    </td></tr>
</table>
</form>
```

## 2. 分页处理

分页处理文件 LeavewordFen.aspx.cs 的功能是,定义分页事件对留言数据进行重新处理。实现流程如下。

- (1) 引入 AjaxLeaveword 命名空间。
- (2) 定义 Page\_Load 载入页面文件。
- (3) 定义 BindPageData()读取并显示留言信息。
- (4) 声明分页事件 gvMessage\_PageIndexChanging(object sender,GridViewPageEventArgs e), 设置 gvMessage 控件的新页码,然后重新绑定 gvMessage 控件数据。

文件 LeavewordFen.aspx.cs 的主要代码如下所示:



```
public partial class BoardPaging : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    private void BindPageData()
    {
        ///获取数据
        Message message = new Message();
        DataSet ds = message.GetMessages();
        ///显示数据
        gvMessage.DataSource = ds;
        gvMessage.DataBind();
    }
    protected void gvMessage_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        ///设置新页面, 并重新绑定数据
        gvMessage.PageIndex = e.NewPageIndex;
        BindPageData();
    }
}
```

经过上述代码设置, 程序执行后, 将首先按照分页模式显示第一分页的数据, 如图 4-9 所示; 当单击下方的对应分页链接后, 将来到指定的页面。

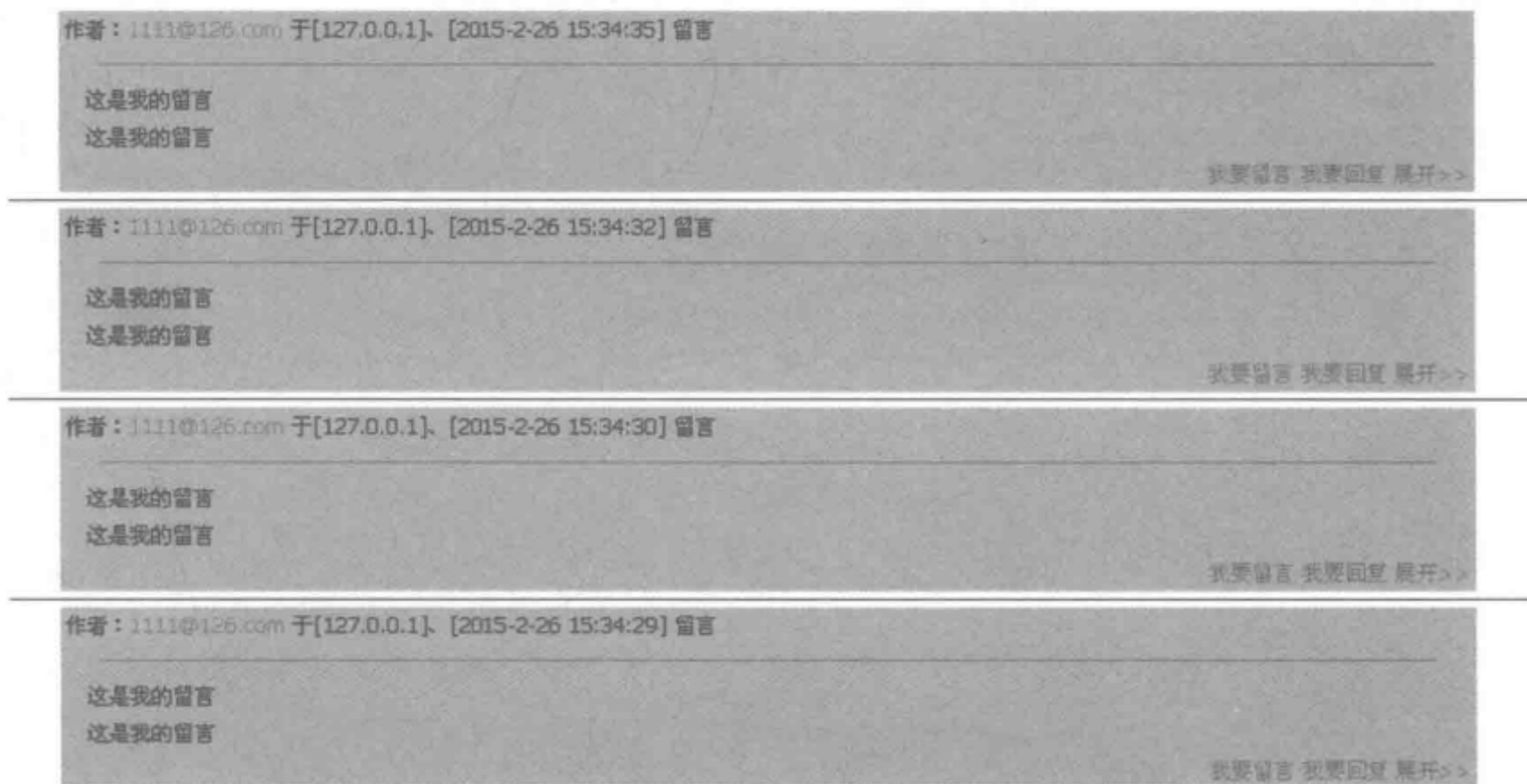


图 4-9 分页的默认显示效果

分页模块是 Web 系统中的常用模块之一, 对于各种动态站点来说, 通过分页计数, 能够用更好的效果将站点内容展示在浏览用户面前。对于 ASP.NET 程序员来说, 是不必烦恼的事。因为 ASP.NET 固有的 GridView 控件很好地实现了分页处理功能, 并且通过它本身的属性可以灵活设置。除了使用 GridView 控件进行分页处理外, 还可以结合数据在库中的保存方式来分页。常见的分页方式有两种, 分别是存储过程分页和控件分页。

### 4.7.3 留言回复模块

接下来,开始设计留言回复模块,此模块的功能是提供系统内留言的回复表单,供用户发布对某留言的回复信息。上述功能的实现文件如下:

- Huifu.aspx。
- Huifu.aspx.cs。

#### 1. 留言回复表单页面文件 Huifu.aspx

留言回复表单页面文件 Huifu.aspx 的功能是,提供留言回复表单,供用户发布对某留言的回复信息。具体的实现流程如下。

- (1) 插入 3 个 TextBox 控件,分别用于 IP 地址、回复内容和验证码的输入框。
- (2) 插入 1 个 CustomValidator 控件,用于对回复内容的验证。
- (3) 插入 1 个 TextBoxWatermarkExtender 控件,用于显示水印提示。
- (4) 插入 1 个 ValidatorCalloutExtender 控件,用于实现多样式验证。
- (5) 调用验证码文件 Yanzhengma.aspx,实现验证码显示。
- (6) 定义 MessageValidator(source, argument),来控制输入的回复内容。

文件 Huifu.aspx 的主要代码如下所示:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Huifu.aspx.cs"
    StylesheetTheme="css" Inherits="Reply" %>
...
<form id="form1" runat="server">
    <asp:ScriptManager ID="sm" runat="server" />
    <table class="Table" border="0" cellpadding="2" bgcolor="Black" cellspacing="1"
        align="center">
        <tr bgcolor="white"><td colspan="2"><hr /></td></tr>
        <tr bgcolor="white">
            <td>IP 地址: </td>
            <td width="90%"><asp:TextBox ID="tbIP" runat="server" Enabled="false"
                SkinID="nn" Width="40%"></asp:TextBox></td>
        </tr>
        <tr bgcolor="white">
            <td valign="top">回复内容: </td>
            <td width="90%">
                <asp:TextBox ID="tbMessage" runat="server" Height="200px" SkinID="nn"
                    TextMode="MultiLine" Width="80%"></asp:TextBox>
                <asp:CustomValidator ID="cvMessage" runat="server"
                    ClientValidationFunction="MessageValidator"
                    ControlToValidate="tbMessage"
                    Display="None" ErrorMessage="长度至少为 10, 最多为 1000。">
                </asp:CustomValidator>
                <ajaxToolkit:TextBoxWatermarkExtender ID="wmeMessage" runat="server"
                    TargetControlID="tbMessage" WatermarkText="请输入留言内容"
                    WatermarkCssClass="Watermark">
                </ajaxToolkit:TextBoxWatermarkExtender>
                <ajaxToolkit:ValidatorCalloutExtender ID="vceMessage" runat="server"
                    TargetControlID="cvMessage" HighlightCssClass="Validator">
                </ajaxToolkit:ValidatorCalloutExtender>
            </td></tr>
        <tr bgcolor="white">
            <td>验证码: </td>
            <td width="90%">
```



&lt;/form&gt;

用 Ajax 控件显示对应的提示。

## 2. 回复数据处理页面

的回复数据添加到系统数据库中。具体实现流程如下。

- (1) 引入命名空间，声明类 Reply。
- (2) 通过 Page\_Load 载入初始化回复表单界面。
- (3) IP 地址判断处理，如果 IP 为空，则停止处理。
- (4) 定义 btnCommit\_Click，进行数据处理。
- (5) 验证码判断处理，如果非法，则输出提示。
- (6) 将数据添加到系统库中。

文件 Huifu.aspx.cs 的主要代码如下所示:

```
...
using ASPNETAJAXWeb.ValidateCode.Page;
public partial class Reply : System.Web.UI.Page
{
    int messageID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取客户端的 IP 地址
        tbIP.Text = Request.UserHostAddress;
        if (Request.Params["MessageID"] != null)
        {
            messageID = Int32.Parse(Request.Params["MessageID"].ToString());
        }
    }
}
```



```

    }
    btnCommit.Enabled = messageID > 0 ? true : false;
}
protected void btnCommit_Click(object sender, EventArgs e)
{
    if (Session[ValidateCode.VALIDATECODEKEY] != null)
    {
        ///验证验证码是否相等
        if (tbCode.Text != Session[ValidateCode.VALIDATECODEKEY].ToString())
        {
            lbMessage.Text = "验证码输入错误, 请重新输入";
            return;
        }
        Message message = new Message();
        ///发表回复
        if (message.AddReply(tbMessage.Text, Request.UserHostAddress, messageID) > 0)
        {
            ///重定向到留言页面
            Response.Redirect("Index.aspx");
        }
    }
}
protected void btnClear_Click(object sender, EventArgs e)
{
    tbMessage.Text = string.Empty;
}
}

```

在本项目的编码过程中, 多次用到了数据操作技术。在 C# 程序中, 初学者容易混淆与数据操作相关的技术。在开发本项目的过程中, 因为作者当时的基本功不够扎实, 所以在开发初期的工作进度徘徊不前, 有些看似简单的问题却浪费了很长时间去解决。那时深深体会到了扎实基本功的重要性, 所以暗下决心以后要刻苦学习, 一定要把基础打牢固。

#### 4.7.4 添加留言信息模块

添加留言信息模块的功能是, 将用户发布的留言信息添加到系统数据库中。上述功能是由文件 Board.aspx.cs 实现的, 其具体实现流程如下。

- (1) 引入命名空间, 声明类 Board。
- (2) 通过 Page\_Load 载入初始化发布表单界面。
- (3) IP 地址判断处理, 如果 IP 为空, 则停止处理。
- (4) 定义 btnCommit\_Click, 进行数据处理。
- (5) 验证码判断处理, 如果非法, 则输出提示。
- (6) 将数据添加到系统库中。

文件 Board.aspx.cs 的主要代码如下所示:

```

public partial class Board : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取客户端的 IP 地址
        tbIP.Text = Request.UserHostAddress;
        if (!Page.IsPostBack)
        {
            BindPageData();
        }
    }
}

```

```

    }
    sm.RegisterAsyncPostBackControl(tbMessage);
}
private void BindPageData()
{
    ///获取数据
    Message message = new Message();
    DataSet ds = message.GetMessages();
    ///显示数据
    gvMessage.DataSource = ds;
    gvMessage.DataBind();
}
protected void btnCommit_Click(object sender, EventArgs e)
{
    if(Session[ValidateCode.VALIDATECODEKEY] != null)
    {
        ///检查验证码是否相等
        if(tbCode.Text != Session[ValidateCode.VALIDATECODEKEY].ToString())
        {
            lbMessage.Text = "验证码输入错误, 请重新输入";
            return;
        }
        Message message = new Message();
        ///发表留言
        if(message.AddMessage(tbTitle.Text, tbMessage.Text,
            Request.UserHostAddress, tbEmail.Text) > 0)
        {
            ///重新显示数据
            BindPageData();
        }
    }
}
protected void btnClear_Click(object sender, EventArgs e)
{
    tbMessage.Text = string.Empty;
}
}

```

从上述留言发布模块的实现过程中可以看出, 留言回复和留言发布的实现过程基本类似, 都是基于数据库的添加处理, 不同的是, 留言发布的数据添加到库内的留言信息表, 而发布的回复数据被添加到库内的回复信息表内。

### 4.7.5 留言管理模块

俗话说, “没有规矩不成方圆”, 这句古话很有道理。家庭有家庭会议, 公司有规章制度。作为舆论大平台的在线留言簿系统来说, 一定要抵制违法言论的出现。所以留言管理模块不仅仅是保证系统数据库够用, 删除不需要的留言数据, 更重要的功能是删除违法的信息。

留言管理功能的实现文件如下:

- Guanli.aspx。
- Guanli.aspx.cs。

#### 1. 留言管理列表页面

留言管理列表页面文件 Guanli.aspx 的功能是, 将系统内的留言数据以分页列表样式显

示出来, 并提供每条留言的删除按钮。具体实现流程如下。

- (1) 插入 1 个 GridView 控件, 用于以列表样式显示留言的信息。包括留言者、邮箱地址、时间和留言内容等。
- (2) 通过 GridView 控件设置分页显示留言数为 5。
- (3) 通过 GridView 控件设置分页处理事件为 gvMessage\_PageIndexChanging。
- (4) 在每条留言的后面插入 1 个 Button 按钮, 用于激活删除处理事件。
- (5) 通过 PagerSettings 设置分页模式为 NextPreviousFirstLast。

文件 Guanli.aspx 的主要代码如下所示:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Guanli.aspx.cs"
    Inherits="BoardManage" StylesheetTheme="css" %>
...
<form id="form1" runat="server">
    <asp:ScriptManager ID="sm" runat="server" />
    <table class="Table" border="0" cellpadding="0" cellspacing="0" align="center">
        <tr><td colspan="2">
            <asp:UpdatePanel runat="server" ID="up">
                <ContentTemplate>
                    <asp:GridView ID="gvMessage" runat="server" Width="100%"
                        AutoGenerateColumns="False"
                        SkinID="mm" ShowHeader="False" AllowPaging="True"
                        OnPageIndexChanging="gvMessage_PageIndexChanging"
                        PageSize="5" OnRowDataBound="gvMessage_RowDataBound"
                        OnRowCommand="gvMessage_RowCommand">
                        <Columns>
                            <asp:TemplateField>
                                <ItemTemplate>
                                    <table class="Table" cellpadding="3" cellspacing="0">
                                        <tr><td>作者: <a href='mailto:<# Eval("Email") %>'><# Eval("Email") %></a>
                                            于[<# Eval("IP") %>]、[<# Eval("CreateDate") %>] 留言</td>
                                        <td align="right">
                                            <asp:Button ID="btnDelete" CommandArgument='<# Eval("ID") %>'
                                                CommandName="del" runat="server" Text="删除该留言" CssClass="Button"
                                                CausesValidation="false" />
                                        </td></tr>
                                        <tr><td colspan="2"><hr size="1" /></td></tr>
                                        <tr><td colspan="2" class="Title"> <# Eval("Title") %></td></tr>
                                        <tr><td colspan="2"> <# Eval("Message") %></td></tr>
                                    </table>
                                </ItemTemplate>
                            </asp:TemplateField>
                        </Columns>
                        <PagerSettings Mode="NextPreviousFirstLast" />
                    </asp:GridView>
                </ContentTemplate>
            </asp:UpdatePanel>
        </td></tr>
    </table>
</form>
```

上述实例代码执行后, 将以分页列表的样式显示系统内的留言数据, 并在每条留言的后面显示一个删除操作按钮。当单击某留言后的“删除该留言”按钮后, 将会激活删除处理程序。由此可见, 无论是本节介绍的留言管理列表界面, 还是前面介绍的留言列表显示页面, 都是采用了 GridView 控件来实现信息显示的。GridView 控件是 ASP.NET 中的核心



控件，它能灵活地绑定数据，并且通过属性来设置元素的显示格式。

## 2. 留言删除处理页面

留言删除处理页面文件 BoardManage.aspx.cs 的功能是，将系统留言数据进行分页处理，并将用户选中的留言数据从系统数据库中删除。具体实现流程如下。

- (1) 引入命名空间，声明类 BoardManage。
- (2) 通过 Page\_Load 载入初始化留言管理列表界面。
- (3) 获取并显示系统内的数据。
- (4) 设置分页处理事件，对数据进行重新绑定。
- (5) 定义 gvMessage\_RowDataBound(object sender, GridViewRowEventArgs e)，弹出“删除确认”对话框。
- (6) 定义 gvMessage\_RowCommand(object sender, GridViewCommandEventArgs e)，将用户选中的数据从系统数据库中删除。

文件 BoardManage.aspx.cs 的主要代码如下所示：

```
public partial class BoardManage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    private void BindPageData()
    {
        ///获取数据
        Message message = new Message();
        DataSet ds = message.GetMessages();
        ///显示数据
        gvMessage.DataSource = ds;
        gvMessage.DataBind();
    }
    protected void gvMessage_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        ///设置新页面，并重新绑定数据
        gvMessage.PageIndex = e.NewPageIndex;
        BindPageData();
    }
    protected void gvMessage_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        Button button = (Button)e.Row.FindControl("btnDelete");
        if (button != null)
        {
            button.Attributes.Add(
                "onclick", "return confirm(\"您确认要删除当前行的留言吗？\")");
        }
    }
    protected void gvMessage_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        if (e.CommandName.ToLower() == "del")
        {
            ///删除选择的留言
            Message message = new Message();
        }
    }
}
```

```

        if (message.DeleteMessage (Int34.Parse (e.CommandArgument.ToString ())) > 0)
        {
            BindPageData (); ///重新绑定数据
        }
    }
}

```

上述代码执行后的显示效果如下：当用户单击“删除该留言”按钮后，将首先弹出“删除确认”对话框。如果单击“取消”按钮，则返回列表页面，如果单击“确定”按钮，则将此留言数据从系统内删除。

到此为止，终于完成了整个项目的编码工作。在编码过程中，发生了一个没有影响项目进度的小插曲，在规划时，确定软件工程师 D 来完成具体的编码工作，因为在日常工作中 D 最为注重办事效率，所以对他很放心，就没全程监督他的工作进度。但是因为 D 有私事，不能参与本项目的工作。无可奈何之下，只能由作者亲自上阵。现在想来还感觉当时真是赶着鸭子上架啊。

一个项目在规划之初，通常会与客户商定某时间内能够完成某些模块，到时候用户会来检查，这个检查结果对整个项目的运作很重要。如果没有完成，客户会怀疑我们的工作效率，甚至水平。反之，如果我们很好地完成了预期的任务，客户会更加相信我们。

通过此事，向读者敲响警钟：项目进度十分重要，一定要全程及时监控各个成员的完成进度。一个项目是由一个团队完成的，在项目进行时，成员之间要及时沟通。这样，不但能及时了解项目的进度，而且能够防止发生意外。即使出现了意外，也能及时了解到，这样，就能在第一时间内想办法解决。

## 4.8 测试运行

视频讲解 光盘：视频\第 4 章\测试运行.avi

将本项目命名为“Leaveword”，首先看主界面，执行结果如图 4-10 所示。

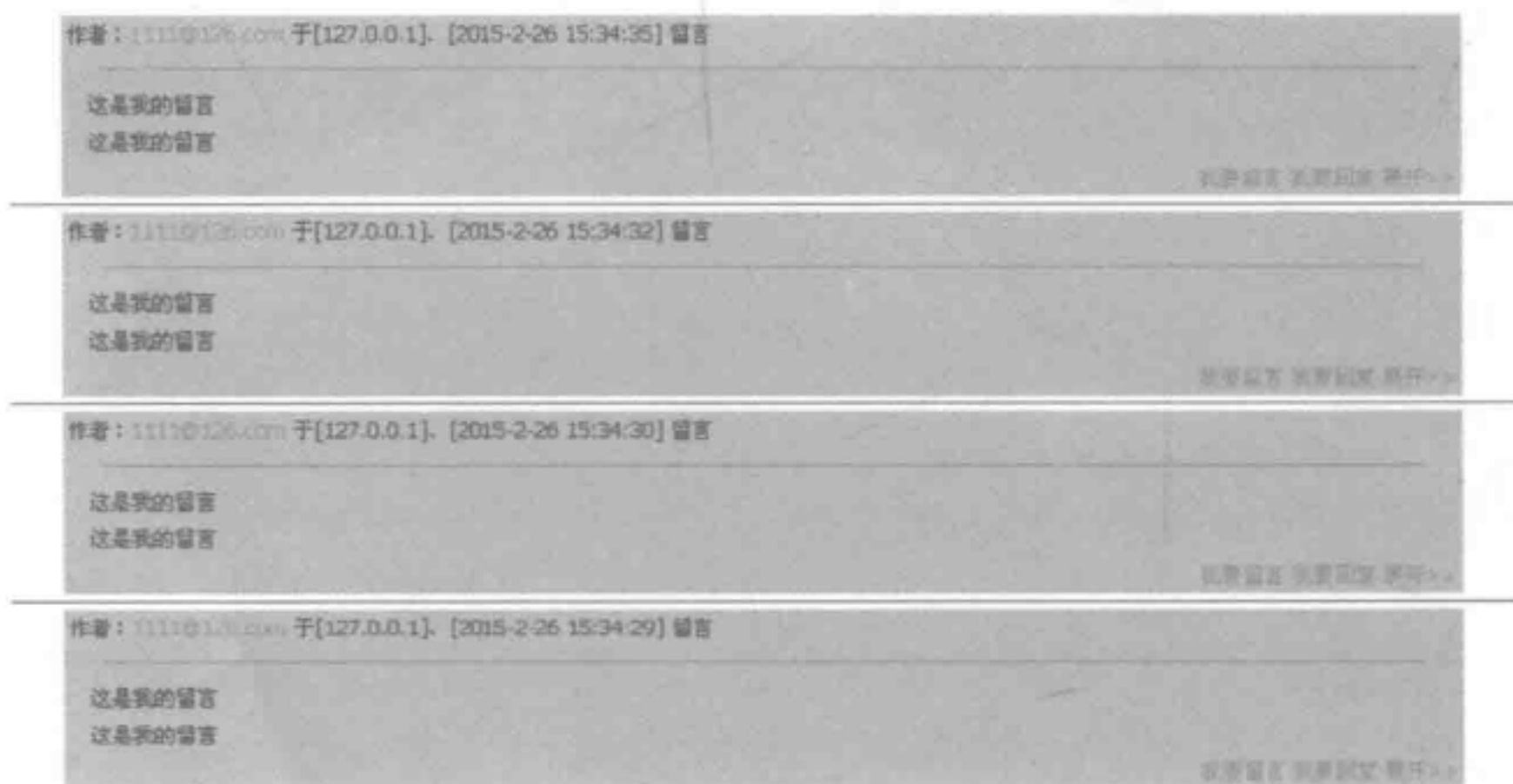


图 4-10 主界面

接着看留言发布表单界面，如图 4-11 所示。

留言标题：

请输入留言标题

IP地址：

127.0.0.1

电子邮件：

请输入电子邮件

留言内容：

请输入留言内容

验证码：

871167

提交

预览

清空

图 4-11 留言发布表单界面

最后看管理界面，如图 4-12 所示。

|                                                       |      |
|-------------------------------------------------------|------|
| 作者：1111@126.com 于[127.0.0.1]、 [2005-2-26 15:34:35] 留言 | 删除留言 |
| 这是我的留言<br>这是我的留言                                      |      |
| 作者：1111@126.com 于[127.0.0.1]、 [2015-2-26 15:34:32] 留言 | 删除留言 |
| 这是我的留言<br>这是我的留言                                      |      |
| 作者：1111@126.com 于[127.0.0.1]、 [2015-2-26 15:34:30] 留言 | 删除留言 |
| 这是我的留言<br>这是我的留言                                      |      |
| 作者：1111@126.com 于[127.0.0.1]、 [2015-2-26 15:34:29] 留言 | 删除留言 |
| 这是我的留言<br>这是我的留言                                      |      |
| 作者：1111@126.com 于[127.0.0.1]、 [2015-2-26 15:34:25] 留言 | 删除留言 |
| 这是我的留言<br>这是我的留言                                      |      |

>>>

图 4-12 管理界面





## 第 5 章 浪漫满屋通信录系统

赢凡项目开发

通信录管理系统是每位用户管理通信录时不可缺少的工具，它的内容对于使用者来说是至关重要的，所以通信录管理系统应该能够为用户提供充足的信息和快捷的查询手段，大大地方便用户对通信录的管理。

在本章的内容中，将通过一个具体实例的实现过程，介绍以 C# 窗体程序实现个人通信录系统的基本流程。



### 赠送的超值电子书

- 041 定义和使用方法
- 042 方法的返回值
- 043 方法参数简介
- 044 变量作用域
- 045 静态方法与实例方法
- 046 使用值参数
- 047 使用引用参数
- 048 使用输出参数
- 049 使用参数数组
- 050 使用数组参数

## 5.1 体验语言之美

视频讲解 光盘：视频\第5章\体验语言之美.avi

程序员一般都能够很容易地用多种编码方式实现同一个功能，每种编码方式都有自己的特色。但在实际项目开发过程中，有的开发者开发的程序能够被领导和客户认可，有的则不能。

既然都是能够实现同样功能的程序，为什么受欢迎度不一样呢？

### 5.1.1 程序员经常忽视的问题

很多技术水平还不错的程序员可能很不屑于公司规定的一些编码规则，在平常的编码过程中不注意自己的编码风格。

例如，把项目中的变量或常量简单地命名为 aa、bb、cc 之类的形式。

在只有几行代码的程序中还好说，如果在成百、上千行代码中，每个变量和常量代表什么呢？眼前全是 aa、bb 之类的简单符号，编程者自己也会难以迅速理解。如果再过个一年半载，到后期维护时，更是早已将这些常量和变量的含义忘光了，给后期维护工作带来的麻烦简直是不可想象的。

编程语言本身往往是用法极为灵活的，但是，按照软件的工业化生产要求，却需要编程者遵循严格的规范。遵循规范的目的，一是为了保证代码的可读性，二是为了保证代码的可维护性。

就像我们上面举的例子，aa、bb 和 cc 之类的命名是不符合编码规范的。

在现实项目中，制订编码规范的出发点有以下 3 个：

- 使得代码统一、易于阅读，便于他人在多年后仍然能轻松地读懂并进行维护。
- 使得代码不受单一平台和编译器的制约，便于移植到其他平台或用其他编译器。
- 保证基本安全，避免代码漏洞。

开发高手和普通人的一条重要区别是，高手们的程序更加美观，一切都显得易读和易维护，编写的代码按部就班，令人赏心悦目。而那些不好的程序却显得杂乱无章，变量、常量和函数的命名毫无规则可循。

### 5.1.2 赢在程序自身——体现代码之美

遵循良好的编码规范，可以开发出赏心悦目的程序。其实，除了需要遵循编码规范外，要想体现出程序之美，开发者还需要注意如下几个方面。

#### (1) 代码简洁，避免冗余

将软件设计作为一门严谨的科学来对待，目的是开发出优雅简洁的代码。

程序结构要清晰，简单易懂，单个函数的程序行数不超过 100 行。过长的代码会影响理解，也会消耗架构者的敏捷性。

明确一个函数的目的是什么，在实现时要尽量简单，直截了当，代码精简，避免产生



任何垃圾程序。

另外，能用标准库函数和公共函数的地方要尽量使用，不要什么功能都自己去编写。

## (2) 遵循严格的规范

在编码过程中，要严格遵循开发语言的编码规范，做到可读性第一，效率第二。

例如：

- 每个源程序文件，都有文件头说明。
- 每个函数，都有函数头说明。
- 主要变量(结构、联合、类或对象)定义或引用时，注释能反映其含义。
- 常量定义要有相应说明。
- 处理过程的每个阶段都有相关的注释说明。
- 在典型算法前都有注释。
- 变量、常量和函数的命名要一目了然。
- 保持注释与代码完全一致，但一目了然的语句不加注释。
- 利用缩进来显示程序的逻辑结构，缩进量一致，并以 Tab 键为单位，缩进后的代码结构清晰，易于维护。

## (3) 健壮性和可扩展性

健壮性是指软件对于规范要求以外的输入能够判断出这个输入不符合规范要求，并能有合理的处理方式。

软件健壮性是一个比较模糊的概念，但却是非常重要的软件外部量度标准。软件设计的健壮与否，直接反映了分析设计和编码人员的水平。

可扩展性是指软件设计完要留有升级接口和升级空间。对扩展开放，对修改关闭。

## (4) 可靠性

可靠性意味着软件在测试运行过程中有避免发生故障的能力，且一旦发生故障后，具有解脱和排除故障的能力。软件可靠性和硬件可靠性的本质区别在于：后者为物理上的衰变和老化所致，而前者是由于设计和实现的错误所致。所以软件的可靠性必须在设计阶段就确定，在生产和测试阶段再考虑就困难了。

## (5) 适应性

适应性要求开发的程序能够在几乎所有的环境下成功运行，而不仅仅局限于在开发者的环境中运行。当今计算机环境千差万别，例如浏览器产品的类型繁多，版本也繁多，假如所开发的是 Web 程序，就需要确保自己的程序具有良好的适应性，能够在世界各地的不同计算机的浏览器中成功运行，并且不会造成兼容性问题。

# 5.2 新的项目

视频讲解 光盘：视频\第5章\新的项目.avi

本项目的客户是一家移动服务公司，要求用 C# + WPF 编程实现。图 5-1 是规划的项目进展流程。

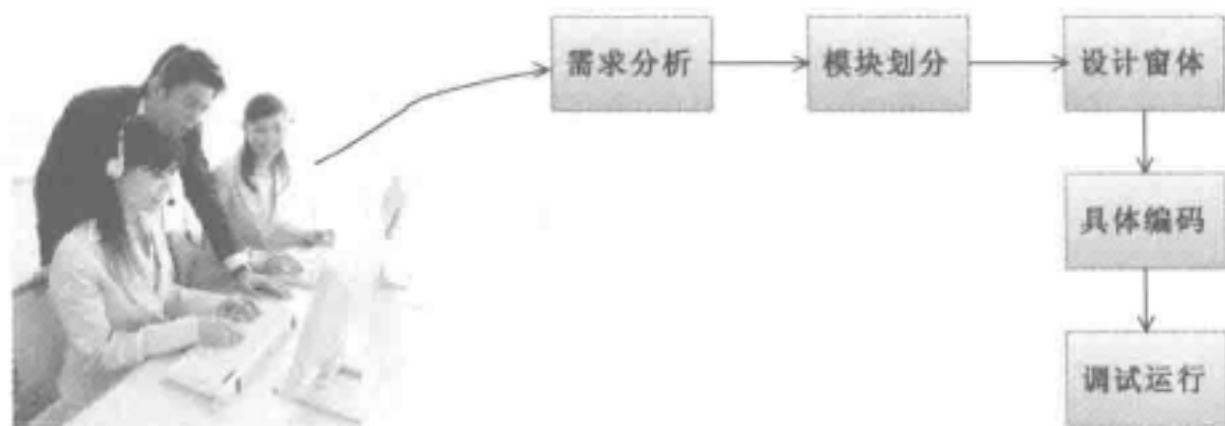


图 5-1 项目进展流程

### 5.2.1 系统分析

通信录的总体目标是为用户提供一个具有多媒体功能的通信簿，用户可以在该通信录中记录联系人的音视频信息、照片及文字信息等。软件将提供丰富的视觉特效、灵活的定制功能，以满足人们对于完美用户界面的需求。用户需要开发一个具有丰富功能的通信录程序，界面具有科幻色彩，能记录联系人的音频、视频照片等信息。

### 5.2.2 系统目标

通过系统分析并与企业管理人员再次探讨，最终确定系统的目标如下：

- 实现需求分析阶段客户提出的功能。
- 对于用户界面要能够定制显示方式，要具有动感效果，能够动态地切换界面。
- 不要违背用户习惯的操作方式，应做到既有新意又能尽快上手。
- 用户对界面的要求是重中之重，要求用户界面要美观、新颖、具有现代感，最好具有类似于 Windows Vista 的 3D Flip 效果。

另外，为了增加系统的美观性，尝试为每个联系人设置一幅图片，并设置音频和视频。图片通常是联系人的照片，而音频和视频通常保存联系人的声音和视频，这样，整个项目会变得更有趣，系统在使用时会更加直观。

最终，确定本系统需要具备如下所示的功能：

- 新增联系人功能：能够新增联系人，添加联系人的姓名、电子邮件及图像。
- 添加音视频文件：能够为联系人添加音频和视频文件。
- 查看联系人功能：能够以列表和卡片式的方式查看联系人信息。
- 设置通信录功能：指定通信录的相关选项。设置用户界面的显示方式等。

到此为止，需求分析阶段已经完成。因为本项目比较简单，所以只对系统分析和系统目标进行了讲解。在开发中、大型软件项目时，需要严格按照软件开发流程进行。

软件开发流程(Software Development Process)包括设计软件的功能和实现算法、软件的总体结构设计和模块设计、编程和调试、程序联调和测试，以及编写、提交程序。

(1) 相关系统分析员和用户初步了解需求，然后用 Word 列出要开发的系统的大功能模块，说明每个大功能模块中有哪些小功能模块，对于有些需求比较明确的功能，在这一步中可以初步定义好少量的界面。

(2) 系统分析员深入了解和分析需求，根据自己的经验和需求，用 Word 或相关的工具再做出一份文档系统的功能需求文档。这次的文档会清楚地列出系统大致的功能模块，大功能模块中有哪些小功能模块，并且还要列出相关的界面和界面功能。



(3) 系统分析员和用户再次确认需求。

(4) 概要设计：概要设计即系统设计，需要对软件系统的设计进行考虑，包括系统的基本处理流程、系统的组织结构、模块划分、功能分配、接口设计、运行设计、数据结构和出错处理设计等，为软件的详细设计提供基础。

(5) 详细设计：在概要设计的基础上，开发者需要进行软件系统的详细设计。在详细设计中，描述实现具体模块所涉及到的主要算法、数据结构、类的层次结构及调用关系，需要说明软件系统各个层次中的每一个程序(每个模块或子程序)的设计考虑，以便进行编码和测试。应当保证软件的需求完全分配给整个软件。详细设计应当足够详细，能够根据详细设计报告进行编码。

(6) 编码：在软件编码阶段，开发者根据《软件系统详细设计报告》中对数据结构、算法分析和模块实现等方面的设计要求，开始具体的编程工作，分别实现各模块的功能，从而实现对目标系统的功能、性能、接口、界面等方面的要求。

(7) 测试：测试编写好的系统。交给用户使用，用户使用后，逐个确认每个功能。

(8) 软件交付准备：在软件测试证明软件达到要求后，软件开发者应向用户提交开发的目标安装程序、数据库的数据字典、《用户安装手册》、《用户使用指南》、需求报告、设计报告、测试报告等双方合同约定的产物。《用户安装手册》应详细介绍安装软件对运行环境的要求、安装软件的定义和内容、客户端/服务器端及中间件的具体安装步骤、安装后的系统配置。《用户使用指南》应包括软件各项功能的使用流程、操作步骤、相应的业务介绍、特殊提示和注意事项等方面的内容，在需要时还应举例说明。

(9) 验收：用户验收。

## 5.3 功能模块划分

视频讲解 光盘：视频\第5章\功能模块划分.avi

系统本身的功能并不复杂，关键在于如何在用户界面上给用户一个创新，一种对于原来用户界面的突破。

针对本系统的功能结构，经过分析，得出如图 5-2 所示的模块结构。

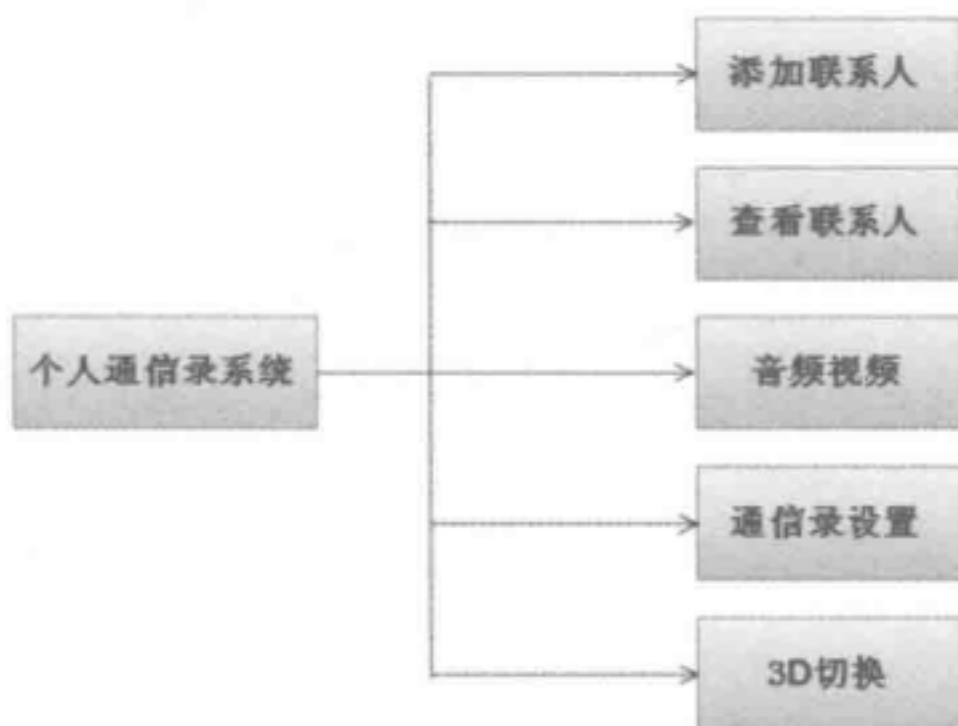


图 5-2 本系统的功能模块结构



## 5.4 设计窗体

视频讲解 光盘：视频\第5章\设计窗体.avi

主窗体是应用程序启动时由 Application 对象指定的一个窗体，它将负责管理整个应用程序的生与死。当应用程序启动时，最先开启主窗体；当主窗体关闭时，应用程序关闭。

为了创建多媒体通信录，首先使用 Visual Studio 新建一个 WPF 应用程序项目，如图 5-3 所示。并把项目命名为 Communication。



图 5-3 新建 WPF 项目

Visual Studio 将会自动新建一个名为 Window1.xaml 的应用程序主窗口和一个 App.xaml 的文件。App.xaml 是整个应用程序的全局应用程序类的派生子类，这是每个应用程序必须有且只能有一个的单件类。

### 5.4.1 设置和启动应用程序

当创建好项目后，还需要进行如下操作。

- (1) 移除自动生成的 Window1.xaml 文件，在解决方案资源管理器中新建一个 Windows 文件夹。
- (2) 创建一个名为 MainInterfaceWindow.xaml 的 WPF 窗口。
- (3) 打开文件 App.xaml，在 VS 提供的 XAML 代码编辑窗口中，将 StartupUri 指定为 MainInterface Window.xaml，这样就指定了 MyFriends 应用程序的主窗口。

上述功能的实现文件如下。

## 1. 文件 App.xaml

文件 App.xaml 用于指定主窗口，定义里面的按钮、窗口等控件的样式。具体实现代码如下所示：

```
<!--指定主窗口，定义各种应用程序事件-->
<Application
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="MyFriends.App"
  StartupUri="Windows/MainInterfaceWindow.xaml"
  Exit="Application_Exit"
  Startup="Application_Startup">
  <Application.Resources>
    <!--定义应用程序级别的资源字典-->
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="Resources/Templates.xaml"/>
        <ResourceDictionary Source="Resources/Styles.xaml"/>
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
</Application>
```

## 2. 文件 App.xaml.cs

App.xaml.cs 是一个处理文件，具体实现流程如下。

(1) 定义 Startup 事件，实现一些初始化的操作，该事件处理代码将为应用程序要使用的一些全局变量赋值，主要是通过将一些路径信息保存到全局应用程序属性集合中来实现的，对应的代码如下所示：

```
private void Application_Startup(object sender, StartupEventArgs e)
{
    //初始化全局应用程序属性
    Application.Current.Properties["SavedDetailsFileName"] = xmlFilename; //保存文件名
    Application.Current.Properties["FullXmlPath"] = //完整的XML路径
        Path.Combine(Environment.CurrentDirectory, xmlFilename);
    Application.Current.Properties["SaveFolder"] = //将要保存到的文件夹
        Environment.CurrentDirectory;
    Application.Current.Properties["SelectedDisplayStyle"] = //默认的显示风格
        DisplayStyle.GrowShrink;
    if (Directory.Exists(@"C:\WINDOWS\Web\Wallpaper"))
        //设置默认的选择图片路径的文件夹
        Application.Current.Properties["SelectedImagePath"] =
            @"C:\WINDOWS\Web\Wallpaper";
    else
        Application.Current.Properties["SelectedImagePath"] = @"C:\\";
    //设置免费的 Exceed 的 Grid 的授权序列号，需要将这里更改为自己的序列号
    Xceed.Wpf.DataGrid.Licenser.LicenseKey = "DGP30-E852N-G9C6E-DW5A";
    //返回一个 FriendsList 对象的实例
    FriendsList.Instance();
}
```

在上述代码中，为 Application 对象的 Properties 集合添加键值对，Properties 集合中保存了将用于保存联系人信息的 XML 文件名称、完整的 XML 文件路径。将要保存的文件夹、默认的显示风格以及默认的图片文件文件夹保存在全局属性集合中，以便于维护和修改。

为 Xceed 的 DataGrid 指定授权序列号, 并且实例化单件类 FriendsList, 使得启动时便能从 XML 文件加载联系人对象到内存集合中。

(2) 定义 Exit 事件, 在应用程序退出时触发, 该事件会保存所有的联系人信息到 XML 文件中, 对应的代码如下所示:

```
private void Application_Exit(object sender, ExitEventArgs e)
{
    try
    {
        //用 SaveOnExit 方法将当前的所有联系人保存到 XML 文件中
        XMLFileOperations.SaveOnExit();
    }
    catch
    {
        //如果保存失败, 显示失败信息
        MessageBox.Show("保存文件时失败!");
    }
}
```

在上述代码中, 通过调用 XMLFileOperations 类的 SaveOnExit 方法, 将当前的所有联系人列表保存到 XML 文件中去。XMLFileOperations 是本系统中集中操作 XML 文件的一个类, 它使用 XLINQ 来读取和保存 XML 文件、修改 XML 文件等。

## 5.4.2 设计用户界面

用户主界面是整个项目中稍稍复杂的一部分, 在用户主界面中, 需要实现导航按钮、3D 变换等用户界面功能。尽管 WPF 在用户界面制作方面提供了前所未有的改进, 但开发人员仍然要学习一些相关的知识, 积累经验。双击打开文件 MainInterfaceWindow.xaml, 首先为用户界面进行布局设计。

此处的设计思路如下: 整个界面由一个 Grid 组成, 在 Grid 内放一个 DockPanel 和一个子 Grid。DockPanel 控件用于设置主窗体的标题头, 另外一个 Grid 用于放置添加联系人和查看联系人的两个用户控件。最终期望的效果如图 5-4 所示。

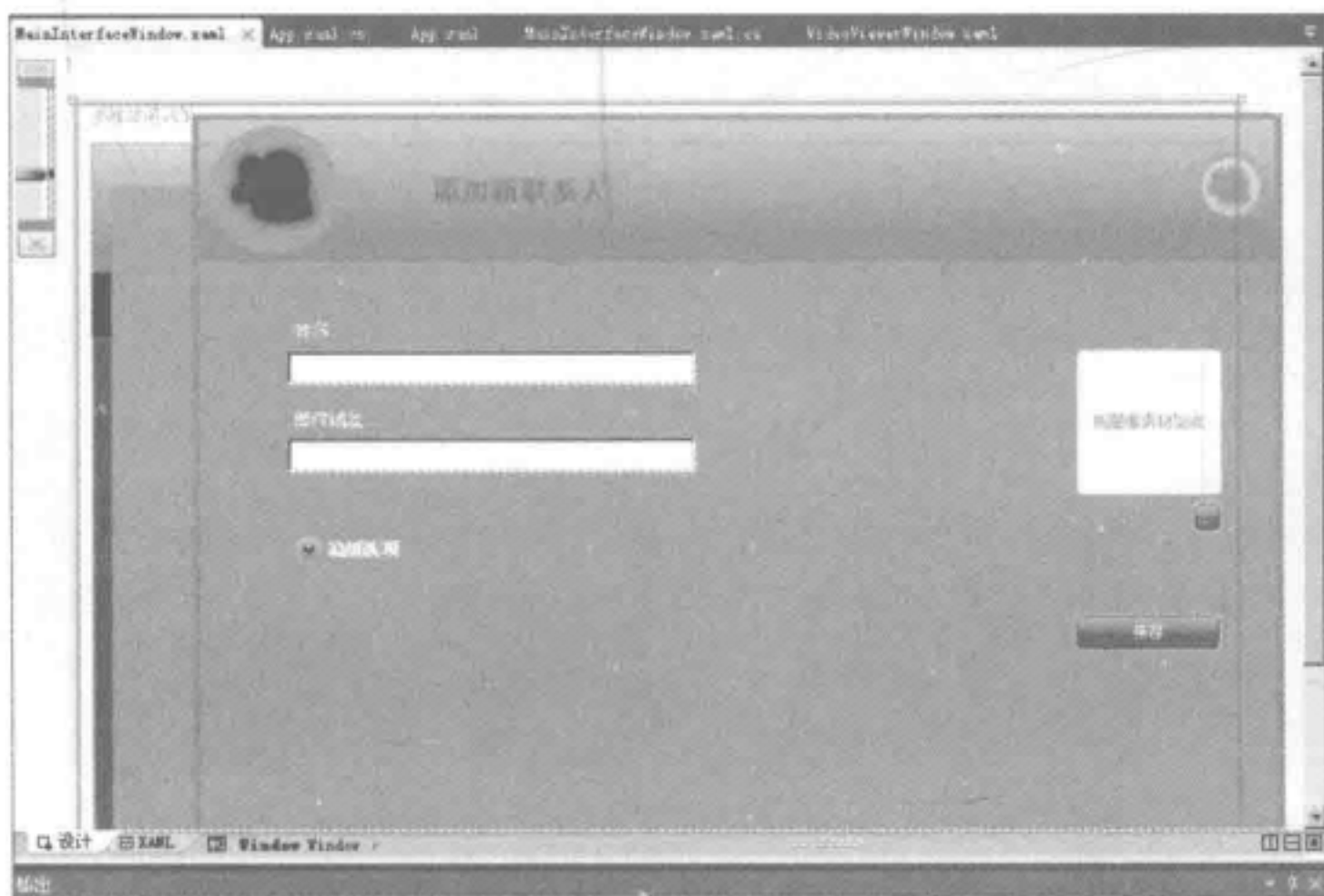


图 5-4 预期的用户界面效果



下面开始编写具体的实现代码。流程很简单，在此简单讲解。

(1) 实现 DockPanel 布局控件，对应代码如下所示：

```
<!--外层的 DockPanel，进行整个标题头的布局-->
<DockPanel Margin="0,0,0,0" LastChildFill="True" Background="#FF414141">
  <!--内层 DockPanel，用于进行标题头和导航按钮的布局-->
  <DockPanel VerticalAlignment="Top" Width="Auto"
    Height="135" DockPanel.Dock="Top" LastChildFill="False">
    <!--标题头画布-->
    <Canvas x:Name="canvasTop"
      Width="{Binding Path=ActualWidth,
        ElementName=GridOuter, Mode=Default}"
      Height="90" DockPanel.Dock="Top">
      <Canvas.Background>
        <!--定义标题头的渐变背景-->
        <LinearGradientBrush EndPoint="0.628,0.051"
          StartPoint="0.628,0.788">
          <GradientStop Color="#FFD0601D" Offset="0"/>
          <GradientStop Color="#FF99FF17" Offset="1"/>
        </LinearGradientBrush>
      </Canvas.Background>
      <!--显示标题头图片-->
      <Image Width="90" Height="90" Canvas.Left="5" Canvas.Top="0" />
      <!--显示多媒体通信录文字的图片-->
      <Image Width="250" Height="50" Canvas.Left="106" Canvas.Top="25"/>
    </Canvas>
    <!--导航的三个按钮所在的画布-->
    <Canvas x:Name="canvasBottom" Width="{Binding Path=ActualWidth,
      ElementName=GridOuter, Mode=Default}"
      Height="45" Background="#FF000000">
      <!--“添加联系人”按钮的定义，其样式定义在模板中-->
      <Button Width="35" x:Name="btnAddFriend" Click="btnAddFriend_Click"
        Height="35" Canvas.Left="14" Canvas.Top="5"
        Template="{DynamicResource GlassButton}"
        ToolTip="添加新联系人" Content="□" BorderThickness="1,1,1,1"
        FontFamily="Webdings" FontSize="23"
        FontWeight="Normal" Foreground="#FFFFFFFF" />
      <!--查看所有联系人按钮定义，其样式定义在模板中-->
      <Button Width="35" x:Name="btnViewAllFriends"
        Click="btnViewAllFriends_Click"
        Height="35" Content="□" Template="{DynamicResource GlassButton}"
        ToolTip="查看所有联系人" FontFamily="Webdings" FontSize="27"
        FontWeight="Normal" Foreground="#FFFFFFFF"
        Canvas.Left="68" Canvas.Top="5" />
      <!--“打开联系人选项”按钮的定义，其样式定义在模板中-->
      <Button Width="35" x:Name="btnOptions" Click="btnOptions_Click" Height="35"
        Content="@" Template="{DynamicResource GlassButton}"
        ToolTip="打开联系人选项窗口" FontFamily="Webdings" FontSize="27"
        FontWeight="Normal" Foreground="#FFFFFFFF"
        Canvas.Left="122" Canvas.Top="5"/>
    </Canvas>
  </DockPanel>
```

在上述代码中，外层的 DockPanel 内部，又嵌入了一个 DockPanel，嵌入的 DockPanel 用于实现标题和按钮的设置。

(2) 设置 Grid 控件 mainGrid，此控件将容纳增加联系人和查看联系人两个窗口。但是，在主窗口的声明中，增加联系人和查看联系人这两个用户控件是放在一个隐藏的名为

gridHolder 的控件中的，对应的实现代码如下所示：

```
<!--容纳 AddNewFriendControl 和 ViewAllUsersControl 控件的 Grid-->
<Grid x:Name="gridHolder" Visibility="Hidden" HorizontalAlignment="Center"
      VerticalAlignment="Center" Margin="0,0,0,0" Background="#6495ED">
    <!--引用定义在工程中的 AddNewFriendControl 控件-->
    <local:AddNewFriendControl x:Name="addFriendsControl"
        RenderTransformOrigin="0.5,0.5"
        Margin="55,-51,-55,-60" Background="#6495ED" Foreground="#6495ED"
        BorderBrush="#FF662B73" Loaded="addFriendsControl_Loaded" Height="611">
        <!--为控件添加变换特性-->
        <local:AddNewFriendControl.RenderTransform>
            <TransformGroup>
                <ScaleTransform ScaleX="1" ScaleY="1"/>
                <SkewTransform AngleX="0" AngleY="0"/>
                <RotateTransform Angle="0"/>
                <TranslateTransform X="0" Y="0"/>
            </TransformGroup>
        </local:AddNewFriendControl.RenderTransform>
    </local:AddNewFriendControl>
    <!--引用定义在工程中的 ViewAllUsersControl 控件-->
    <local:ViewAllUsersControl x:Name="viewAllUsersControl"
        HorizontalAlignment="Center"
        Width="750" Height="500"
        RenderTransformOrigin="0.5,0.5" Opacity="0.0"
        Loaded="viewAllUsersControl_Loaded">
        <!--为控件添加变换特性-->
        <local:ViewAllUsersControl.RenderTransform>
            <TransformGroup>
                <ScaleTransform ScaleX="0" ScaleY="0"/>
                <SkewTransform AngleX="0" AngleY="0"/>
                <RotateTransform Angle="0"/>
                <TranslateTransform X="0" Y="0"/>
            </TransformGroup>
        </local:ViewAllUsersControl.RenderTransform>
    </local:ViewAllUsersControl>
</Grid>
```

(3) 在文件 MainInterfaceWindow.xaml.cs 中设置操作处理事件，对应的代码如下：

```
public MainInterfaceWindow()
{
    InitializeComponent();
    CreateAlphaGrowArea(true); //创建缩放显示区域
    btnViewAllFriends.IsEnabled = true; //启用查看联系人按钮
    btnAddFriend.IsEnabled = false; //关闭添加联系人按钮
}
```

(4) 定义 CreateAlphaGrowArea 方法创建缩放区域，搜索 XAML 中指定名称的故事板，然后为故事板关联事件，对应的代码如下所示：

```
private void CreateAlphaGrowArea(bool initialCall)
{
    //初始化添加联系人控件
    addFriendsControl.ReInitialise();
    //设置当前的显示风格
    currentDisplayStyle = DisplayStyle.GrowShrink;
    mainGrid.Children.Clear(); //首先清除 Grid 中的控件
    //如果 GridOuter 中存在 gridHolder，先从 GridOuter 移除
    if (GridOuter.Children.Contains(gridHolder))
```



```

        GridOuter.Children.Remove(gridHolder);
//在mainGrid控件集合中添加gridHolder控件
if (!mainGrid.Children.Contains(gridHolder))
    mainGrid.Children.Add(gridHolder);
//让gridHolder控件得以显示
gridHolder.Visibility = Visibility.Visible;
//获取故事板并为故事板关联 Completed 事件
Grow_Hide_AddUserControlStoryboard =
    this.TryFindResource("GrowAndHideAddUserControl") as Storyboard;
Grow_Hide_AddUserControlStoryboard.Completed +=
    new EventHandler(Grow_Hide_AddUserControl_Completed);
Grow_Show_ViewUsersControlStoryboard =
    this.TryFindResource("GrowAndShowViewUsersControl") as Storyboard;
Shrink_Show_AddUserControlStoryboard =
    this.TryFindResource("ShrinkAndShowAddUserControl") as Storyboard;
Shrink_Show_AddUserControlStoryboard.Completed +=
    new EventHandler(Shrink_Show_AddUserControl_Completed);
Shrink_Hide_ViewUsersControlStoryboard =
    this.TryFindResource("ShrinkAndHideViewUsersControl") as Storyboard;
if (initialCall) //如果已经初始化, 则将当前显示选项设置为已显示
    currentDisplayOption = CurrentDisplayOption.AddIsShown;
}

```

(5) 定义 Completed 事件, 设置控件的显示和隐藏来实现切换界面的效果, 对应的代码如下所示:

```

void Shrink_Show_AddUserControl_Completed(object sender, EventArgs e)
{
    addFriendsControl.Visibility = Visibility.Visible; //显示添加联系人
    viewAllUsersControl.Visibility = Visibility.Hidden; //隐藏查看联系人
    currentDisplayOption = CurrentDisplayOption.AddIsShown; //设置显示状态
}
void Grow_Hide_AddUserControl_Completed(object sender, EventArgs e)
{
    addFriendsControl.Visibility = Visibility.Hidden; //隐藏添加联系人
    viewAllUsersControl.Visibility = Visibility.Visible; //显示查看联系人
    currentDisplayOption = CurrentDisplayOption.ViewIsShown; //设置显示状态
}

```

(6) 当单击页面顶部的导航按钮时, 将开始执行并完成切换的操作。“查看联系人”按钮与“添加联系人”按钮的处理代码如下所示:

```

//“查看联系人”按钮单击事件的处理代码
void btnViewAllFriends_Click(object sender, RoutedEventArgs e)
{
    viewAllUsersControl.Visibility = Visibility.Visible; //显示查看控件
    Grow_Hide_AddUserControlStoryboard.Begin(this); //开始动画
    Grow_Show_ViewUsersControlStoryboard.Begin(this); //开始动画
    btnAddFriend.IsEnabled = true; //启用“添加”按钮
    btnViewAllFriends.IsEnabled = false; //启用“查看”按钮
    viewAllUsersControl.DataBind(); //绑定联系人数据
}
//“添加联系人”按钮的单击事件处理代码
void btnAddFriend_Click(object sender, RoutedEventArgs e)
{
    addFriendsControl.Visibility = Visibility.Visible; //显示添加控件
    viewAllUsersControl.Visibility = Visibility.Visible; //显示查看控件
    Shrink_Show_AddUserControlStoryboard.Begin(this); //开始动画
    Shrink_Hide_ViewUsersControlStoryboard.Begin(this); //开始动画
}

```



```

    btnAddFriend.IsEnabled = false; //禁用“添加”按钮
    btnViewAllFriends.IsEnabled = true; //启用“查看”按钮
}

```

### 5.4.3 实现三维动画效果

当单击“选项”按钮，并在显示风格中设置为 3D 翻页效果时，主窗口的“添加联系人”按钮和“查看所有联系人”按钮将被禁用。在“添加联系人”控件和“查看所有联系人”控件的底部会增加一个黄色的按钮条，允许用户单击，来获得 3D 翻页转场效果。

(1) 通过“选项”按钮的单击事件，为 3D 翻页进行一些初始化的操作。

文件 MainInterfaceWindow.xaml.cs 中的初始化操作代码如下所示：

```

private void btnOptions_Click(object sender, RoutedEventArgs e)
{
    switch (currentDisplayOption)
    {
        case CurrentDisplayOption.AddIsShown:
            topleft = //根据当前的显示状态设置控件的初始位置
                addFriendsControl.PointToScreen(new Point(0, 0));
            break;
        case CurrentDisplayOption.ViewIsShown:
            topleft = //如果当前查看的联系人为显示状态，则将其定位到左上角
                viewAllUsersControl.PointToScreen(new Point(0, 0));
            break;
    }
    OptionsWindow optionsWin = new OptionsWindow(); //选项窗口
    optionsWin.Owner = this;
    optionsWin.CurrentDisplayStyle = currentDisplayStyle; //设置选项的当前显示风格
    optionsWin.WindowStartupLocation =
        WindowStartupLocation.CenterScreen; //窗口位置
    optionsWin.ShowDialog(); //显示选项按钮
    //获取定义在应用程序属性中的显示风格
    DisplayStyle newDisplayStyle =
        (DisplayStyle)Application.Current.Properties["SelectedDisplayStyle"];
    bool showingAddControl = false; //控件显示布尔值
    if (newDisplayStyle != currentDisplayStyle) //判断新的风格与当前风格是否一致
    {
        if (newDisplayStyle.Equals(DisplayStyle.ThreeDimension)) //如果新的风格为三维风格
        {
            Create3Area(); //用于创建三维动画区域
            btnViewAllFriends.IsEnabled = false; //禁用查看按钮
            btnAddFriend.IsEnabled = false; //禁用添加按钮
        }
        else //如果是动画风格
        {
            CreateAlphaGrowArea(false); //初始化动画方法
            switch (currentDisplayOption) //根据当前的显示状态设置控件的可见性和按钮的可用性
            {
                case CurrentDisplayOption.AddIsShown:
                    viewAllUsersControl.Visibility = Visibility.Visible;
                    showingAddControl = true;
                    break;
                case CurrentDisplayOption.ViewIsShown:
                    showingAddControl = false;
                    viewAllUsersControl.Visibility = Visibility.Visible;
                    break;
            }
        }
    }
}

```

```

        //根据是否显示, 设置按钮的启用或禁用
        btnViewAllFriends.IsEnabled = showingAddControl;
        btnAddFriend.IsEnabled = !showingAddControl;
    }
}

```

(2) 定义 Create3Area() 方法, 实现三维的初始化操作。先创建一个 ItemsControl 控件, 为其 ItemTemplate 赋一个定义在资源中的数据模板, 然后添加一个单独的项, 最后将 ItemsControl 控件添加到容器 Grid 中。对应的代码如下所示:

```

//创建三维动画区域
private void Create3Area()
{
    currentDisplayStyle = DisplayStyle.ThreeDimension; //设置当前的显示风格
    //将控件移动到 Grid 中
    if (mainGrid.Children.Contains(gridHolder))
        mainGrid.Children.Remove(gridHolder);
    if (!GridOuter.Children.Contains(gridHolder))
        GridOuter.Children.Add(gridHolder);
    gridHolder.Visibility = Visibility.Collapsed; //折叠控件
    //取消关联委托, 在 3D 模式时将不使用动画
    if (Grow_Hide_AddUserControlStoryboard != null)
        Grow_Hide_AddUserControlStoryboard.Completed -=
            //移除故事板 Completed 事件
            new EventHandler(Grow_Hide_AddUserControl_Completed);
    if (Shrink_Show_AddUserControlStoryboard != null)
        Shrink_Show_AddUserControlStoryboard.Completed -=
            //移除故事板 Completed 事件
            new EventHandler(Shrink_Show_AddUserControl_Completed);
    //现在创建 3D 动画区域
    items3d = new ItemsControl(); //实例化一个 ItemsControl 控件
    //设置控件的对齐依赖属性
    items3d.SetValue(Grid.HorizontalAlignmentProperty, HorizontalAlignment.Center);
    items3d.SetValue(Grid.VerticalAlignmentProperty, VerticalAlignment.Center);
    //设置控件的 ItemTemplate 模板为资源 flipItemTemplate 模板
    items3d.ItemTemplate = this.TryFindResource("flipItemTemplate") as DataTemplate;
    //随便添加一项
    items3d.Items.Add("I care");
    mainGrid.Children.Clear(); //清除容器 Grid
    mainGrid.Children.Add(items3d); //将 ItemsControl 控件添加进去
}

```

#### 5.4.4 遍历窗体可视化树

在 WPF 中, 窗体上的所有 UI 元素都属于 Windows 逻辑树。通过可视化树, 开发人员可以获取对于正确的 Item 的引用, 并且直接改变其属性。3D 效果实际上是 DataTemplate 的一部分, 被作为一个 ItemsControl 中的单个 Item。当把显示风格设置为 3D 模式时, 将会触发 AddNewFriendControl 的 SizeChanged 事件, 通过此事件来更新位于 3D DataTemplate 中的内容。文件 MainInterfaceWindow.xaml.cs 中的对应实现代码如下所示:

```

void AddNewFriendControl_SizeChanged(object sender, SizeChangedEventArgs e)
{
    addfriendsControl3D = sender as AddNewFriendControl; //获取对象引用
    addfriendsControl3D.ReInitialise(); //重新初始化最后一联系人内容
    //获取 ViewAllUsersControl 时有些技巧,
}

```



```

//需要在 DataTemplate 中查找, 意味着需要遍历视觉树
DependencyObject item = null;
//因为 DataItem 中只有一个 Item, 作为一个技巧方法来应用自定义的 3D 模板
foreach (object dataitem in items3d.Items)
{
    //获取 ItemsControl 中的 UIElement
    item = items3d.ItemContainerGenerator.
        ContainerFromItem(dataitem);
    int count = VisualTreeHelper.GetChildrenCount(item);
    for (int i=0; i<count; i++)
    {
        DependencyObject itemFetched = VisualTreeHelper.GetChild(item, i);
        //查找 Grid, 找出哪一个允许查找相关的宿主
        //ViewAllUsersControl 的 ContentPresenter
        if (itemFetched is Grid)
        {
            //查找 backContent 中的 ViewAllUsersControl, 设置其高度并重新绑定数据
            ContentPresenter cp =
                (itemFetched as Grid).FindName("backContent") as ContentPresenter;
            DataTemplate myDataTemplate = cp.ContentTemplate;
            ViewAllUsersControl viewUsers =
                (ViewAllUsersControl)myDataTemplate.FindName(
                    "viewFriendsControl3d", cp);
            viewUsers.Height = (sender as AddNewFriendControl).Height;
            viewUsers.DataBind();
            return;
        }
    }
}
}

```

对于企业来说, WPF 实现了改进的客户关系和不同的应用程序。通过使用能够快速提供更好的视觉效果、独特的用户体验的技术, 来建立与客户的密切关系, 使企业可以建立稳定的数字客户关系和独特的品牌化机会。而且, 由于 WPF 是窗体、文档、视频、三维以及其他功能的综合, 因此企业可以创建持久的用户体验解决方案, 并集成到客户的日常活动中。对于开发人员和设计人员, WPF 提供了统一的 UI 平台, 因此他们只需学习一个模式, 就可以获得无限可能的 UI 体验。对于 .NET 开发人员, 其框架是熟悉的, 并且它最终将减少提供最佳用户体验和通信逻辑所需的代码行数。对于设计人员, WPF 提供的平台可消除内容、媒体和应用程序之间的边界。最重要的是, WPF 可以使开发人员和设计人员同步紧密地合作, 来快速提供不同的连通体验。

### 5.4.5 添加联系人

“添加联系人”窗口是一个用户控件, 提供了添加通信录的入口。

“添加联系人”窗口包含一些有趣的功能, 例如, 添加图片时实现了漂亮的图片选择对话框。添加联系人的可折叠的高级功能面板, 使用户可以拖动音频或视频文件, 让联系人资料更加丰富和吸引人。

#### 1. 基本的用户界面功能

要实现“添加联系人”用户控件, 应先创建一个 Controls 文件夹, 右键单击该文件夹, 添加一个新的 WPF 用户控件, 命名为 AddNewFriendControl.xaml。



“添加联系人”用户控件的主体布局由一个 Grid 布局控件组成，内部包含一个 DockPanel 控件，在 DockPanel 内部使用了两个 Canvas 画布，一个用于设置标题栏，一个用于放置实际的添加联系人内容。

(1) 先看控件的声明部分，对应的代码如下所示：

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="MyFriends.AddNewFriendControl"
  x:Name="Control"
  Width="750" Height="500" Background="{x:Null}"
  AllowDrop="True">
</UserControl>
```

(2) 在用户控件的最外部，通过一个 Border 控件来创建一个圆角矩形的外边框，使用户控件更具吸引力。接下来创建一个 Canvas，用于设置用户控件的标题栏。

对应的代码如下所示：

```
<Grid x:Name="LayoutRoot" Opacity="1">
  <!--在整个控件的外部，用一个 Border 设置边框-->
  <Border HorizontalAlignment="Stretch"
    Margin="0,0,0,0" Width="Auto"
    Background="#6495ED"
    BorderBrush="#FFD0601D"
    BorderThickness="5,5,5,5"
    CornerRadius="5,5,5,5">
    <!--添加一个用于进行用户界面布局的 DockPanel 控件-->
    <DockPanel Width="Auto" Height="Auto" LastChildFill="True">
      <!--添加一个用于定义标题栏的 Canvas-->
      <Canvas Margin="0,-1,0,0" x:Name="cansTop"
        VerticalAlignment="Top" Width="Auto"
        Height="100" DockPanel.Dock="Top">
        <!--使用属性元素语法定义渐变色-->
        <Canvas.Background>
          <LinearGradientBrush EndPoint="0.628,0.051"
            StartPoint="0.628,0.788">
            <GradientStop Color="#FFD0601D" Offset="0"/>
            <GradientStop Color="#FFFB917" Offset="1"/>
          </LinearGradientBrush>
        </Canvas.Background>
        <!--添加左侧的用于修饰用的红色椭圆-->
        <Ellipse Fill="#6495ED" Stroke="#6495ED"
          StrokeThickness="5" Width="90"
          Height="90" Panel.ZIndex="1"
          Canvas.Left="7" Canvas.Top="4"/>
        <!--添加右侧的用于修饰用的红色椭圆-->
        <Ellipse Fill="#FFFEFEFE" Stroke="#FFFF6717"
          StrokeThickness="5" Width="50"
          Height="50" Panel.ZIndex="0"
          Canvas.Left="685" Opacity="0.5"
          Canvas.Top="23"/>
        <!--添加左侧的图标-->
        <Image Width="75.196" Height="66.02"
          Panel.ZIndex="1" Source="..\Images\myFriends.png"
          Canvas.Left="14" Canvas.Top="13"/>
        <!--添加右侧的图标-->
        <Image Width="37.196" Height="35.549"
          Panel.ZIndex="1" Source="..\Images\myFriends.png"/>
```

```

        Opacity="0.4" Canvas.Left="690" Canvas.Top="30"/>
        <!--添加中间的添加联系人图像-->
        <Image Width="230.222" Height="37"
            Canvas.Left="104.778" Canvas.Top="31"
            Source="..\Images\AddFriendWords.png"/>
    </Canvas>

```

## 2. 输入窗体

创建了标题栏之后，接下来，创建用户输入的窗体。

从文档大纲视图中可以看到，用户输入栏包括了用于输入名字和 E-mail 的 TextBox 和 Label 控件、用于允许用户添加图片的 Image 控件、用于保存和选择新图像的 Button 控件，以及一个用于显示高级内容的 Expander 控件。

(1) 输入用户名及电子邮件的是两个 TextBox 控件，分别使用两个 Label 控件向用户标识其要输入的内容。位于其下面的是一个 Expander 控件，它提供了联系人多媒体信息的输入，对应的实现代码如下所示：

```

<!--定义用户高级选项折叠区域-->
<Expander Header="高级选项" x:Name="expAdvancedOptions"
    Foreground="#FFFFFFFF"
    Style="{DynamicResource ExpanderOrangeStyle}"
    Canvas.Left="63" Canvas.Top="186" Width="522"
    Height="197" FontWeight="Bold">
    <!--使用 StackPanel 进行布局-->
    <StackPanel Orientation="Vertical" Margin="0,20,0,0">
        <StackPanel Orientation="Horizontal" HorizontalAlignment="Left">
            <!--添加一个圆角的 Border 控件-->
            <Border Width="60" Height="60" Background="#FFFFFFFF"
                BorderBrush="{x:Null}" CornerRadius="5,5,5,5"
                HorizontalAlignment="Left">
                <!--使用 Grid 控件来接受拖放操作-->
                <Grid x:Name="videoGrid" Drop="videoGrid_Drop">
                    <TextBlock HorizontalAlignment="Center"
                        VerticalAlignment="Center"
                        Text="设置一个视频文件" Width="61"
                        TextAlignment="Center"
                        TextWrapping="WrapWithOverflow"
                        FontFamily="Arial" FontSize="11"
                        FontWeight="Normal" Foreground="#FF000000"/>
                    <!--使用 MediaElement 来播放多媒体文件-->
                    <MediaElement x:Name="videoSrc"
                        LoadedBehavior="Manual"
                        Stretch="Fill" Margin="5,5,5,5"/>
                </Grid>
            </Border>
            <!--播放视频按钮-->
            <Button x:Name="btnVideoPlay" ToolTip="播放"
                Template="{DynamicResource GlassButton}"
                Width="20" Height="20" Content="4"
                Canvas.Left="683" Canvas.Top="166"
                Foreground="#FF54FB0C" FontSize="15"
                Margin="20,0,0,0" FontFamily="Webdings"
                Click="btnVideoPlay_Click"/>
            <!--停止播放视频按钮-->
            <Button x:Name="btnVideoStop" ToolTip="停止"
                Template="{DynamicResource GlassButton}"
                Width="20" Height="20" Content="5"/>
        </StackPanel>
    </StackPanel>

```



```

Canvas.Left="683" Canvas.Top="166"
Foreground="#FFFFFFFF" FontSize="12"
Margin="20,0,0,0" FontFamily="Webdings"
Click="btnVideoStop_Click"/>
<!--放置文本-->
<TextBlock Margin="10,0,0,0" Text="视频文件"
TextAlignment="Justify"
TextWrapping="WrapWithOverflow"
FontFamily="Arial" FontSize="11"
FontWeight="Normal"
Foreground="#FFFFFFFF"
VerticalAlignment="Center"/>
</StackPanel>

```

(2) 选择联系人图片按钮的布局与选择多媒体文件的代码比较相似，并且也允许用户从资源管理器中拖动图像，对应的实现代码如下所示：

```

...
<StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="0,10,0,0">
  <Border Width="60" Height="60" Margin="0,0,0,0"
    Background="#FFFFFFFF" BorderBrush="{x:Null}"
    CornerRadius="5,5,5,5" HorizontalAlignment="Left">
    <Grid x:Name="musicGrid" Drop="musicGrid_Drop">
      <TextBlock x:Name="txtMusic"
        HorizontalAlignment="Center"
        VerticalAlignment="Center"
        Text="设置一个音频" Width="62"
        TextAlignment="Center" TextWrapping="WrapWithOverflow"
        FontFamily="Arial" FontSize="11" FontWeight="Normal"
        Foreground="#FF000000" Visibility="Visible"/>
      <MediaElement x:Name="musicSrc"
        LoadedBehavior="Manual" Stretch="Fill"/>
      <Image x:Name="imgMusic"
        Source="..\Images\music.png"
        HorizontalAlignment="Center"
        VerticalAlignment="Center"
        Width="39" Height="49"
        Visibility="Hidden"/>
    </Grid>
  </Border>
  <Button x:Name="btnMusicPlay" ToolTip="播放"
    Template="{DynamicResource GlassButton}"
    Width="20" Height="20" Content="4"
    Canvas.Left="683" Canvas.Top="166"
    Foreground="#FF54FB0C" FontSize="15"
    Margin="20,0,0,0" FontFamily="Webdings"
    Click="btnMusicPlay_Click"/>
  <Button x:Name="btnMusicStop" ToolTip="停止"
    Template="{DynamicResource GlassButton}" Width="20" Height="20" Content="3"
    Canvas.Left="683" Canvas.Top="166" Foreground="#FFFFFFFF" FontSize="12"
    Margin="20,0,0,0" FontFamily="Webdings" Click="btnMusicStop_Click"/>
  <TextBlock Margin="10,0,0,0" Text="音频文件" TextAlignment="Justify"
    TextWrapping="WrapWithOverflow" FontFamily="Arial" FontSize="11"
    FontWeight="Normal" Foreground="#FFFFFFFF" VerticalAlignment="Center"/>
</StackPanel>
<!--选择图片区域-->
<Border Width="100" Height="100" Background="#FFFFFFFF"
  BorderBrush="{x:Null}"
  CornerRadius="5,5,5,5"
  Canvas.Left="603" Canvas.Top="60">

```



```

<!--定义一个允许接受拖入操作的 Grid-->
<Grid x:Name="imgGrid" Drop="imgGrid_Drop">
    <!--用于显示文字信息的 TextBlock 控件-->
    <TextBlock Margin="5,5,5,5"
        HorizontalAlignment="Center"
        VerticalAlignment="Center"
        Text="拖图像素材到此" Width="80"
        TextAlignment="Center"
        TextWrapping="WrapWithOverflow"
        FontFamily="Arial" FontSize="11"
        FontWeight="Normal"/>
    <!--用于显示图像的 Image 控件-->
    <Image x:Name="photoSrc" Margin="5,5,5,5" Stretch="Fill" />
</Grid>
</Border>
<!--定义选择图像的按钮-->
<Button x:Name="btnChooseNewImage"
    ToolTip="选择一幅插图"
    Template="{DynamicResource GlassButton}"
    Width="20" Height="20" Content="..."
    Canvas.Left="683" Canvas.Top="166"
    Foreground="#FFFFFFBFB"
    Click="btnChooseNewImage_Click"
    FontSize="9"/>
<!--用于保存联系人信息的按钮-->
<Button x:Name="btnSave" ToolTip="保存"
    Template="{DynamicResource GlassButton}"
    Width="101" Height="25" Content="保存"
    Foreground="#FFFFFFBFB" FontSize="11"
    Canvas.Left="601" Canvas.Top="243"
    Click="btnSave_Click"/>
</Canvas>

```

## 5.4.6 实现多媒体

多媒体功能就是指用户可以通过拖动添加多媒体文件，播放或暂停多媒体文件。要想添加音视频文件，需在资源管理器中选中一个多媒体文件，比如 WMV 或 MP3 格式的文件，拖动到相应的音视频区域上面，将触发其容器 Grid 的 Drop 事件。

(1) 看视频的处理过程，当拖动一个 WMV 到视频窗口上时，其触发的 Drop 事件处理代码在文件 AddNewFriendControl.xaml.cs 中实现，对应的代码如下所示：

```

private void videoGrid_Drop(object sender, DragEventArgs e)
{
    //获取拖放的文件名数组
    string[] fileNames = e.Data.GetData(DataFormats.FileDrop, true) as string[];
    string[] allowableFiles = {".wmv", ".avi"};
    if (fileNames.Length > 0)
    {
        FileInfo f = new FileInfo(fileNames[0]);
        //如果文件类型为 WMV 或 AVI，则将文件路径赋给 MediaElement 对象
        if (allowableFiles.Contains(f.Extension.ToLower()))
        {
            friendContent.VideoUrl = fileNames[0];
            videoSrc.Source = null;
            videoSrc.Source = new Uri(friendContent.VideoUrl);
            MessageBox.Show("已经成功的添加了一个视频\r\n" + "可以通过所提供的按钮来控制视频!");
        }
    }
}

```

```

}
//确保已经处理了事件, 那么将不会调用基类的方法处理拖动操作
e.Handled = true;
}

```

(2) 音频部分的拖放操作与视频类似, 在 Grid 的 Drop 事件中, 通过判断拖入的是否是 WMA 或 MP3 文件, 将文件路径赋给 MediaElement 对象, 对应的代码如下所示:

```

private void musicGrid_Drop(object sender, DragEventArgs e)
{
    //获取拖放操作的文件名数组
    string[] fileNames = e.Data.GetData(DataFormats.FileDrop, true) as string[];
    string[] allowableFiles = {".wma", ".mp3"};
    if (fileNames.Length > 0) //如果文件名存在
    {
        FileInfo f = new FileInfo(fileNames[0]);
        //判断指定的文件格式是否是所允许放置的格式
        if (allowableFiles.Contains(f.Extension.ToLower()))
        {
            friendContent.MusicUrl = fileNames[0];
            musicSrc.Source = new Uri(friendContent.MusicUrl);
            //拖放操作后, 将文件路径赋给 MediaElement 并设置文本和图片的显示
            txtMusic.Visibility = Visibility.Hidden;
            imgMusic.Visibility = Visibility.Visible;
            MessageBox.Show(
                "已经成功地添加了一个音频文件\r\n" + "可以使用所提供的按钮来控制音频");
        }
    }
    //确保已经处理了事件, 那么将不会调用基类的方法处理拖动操作
    e.Handled = true;
}

```

(3) 可以使用按钮来控制播放与暂停, 这是通过调用 MediaElement 对象的相关方法来实现的, 对应的实现代码如下所示:

```

private void btnVideoPlay_Click(object sender, RoutedEventArgs e)
{
    videoSrc.Play(); //播放视频
}
private void btnVideoStop_Click(object sender, RoutedEventArgs e)
{
    videoSrc.Stop(); //停止视频播放
}
private void btnMusicPlay_Click(object sender, RoutedEventArgs e)
{
    musicSrc.Play(); //播放音频
}
private void btnMusicStop_Click(object sender, RoutedEventArgs e)
{
    musicSrc.Stop(); //停止音频播放
}


```

### 5.4.7 添加图片

在“添加联系人”窗口中, 可以拖动一幅图片到添加图片区域, 也可以单击按钮选择一幅图片。

(1) 拖动图片的操作与拖动音频和视频的代码非常相似, 对应的实现代码如下所示:

```
private void imgGrid_Drop(object sender, DragEventArgs e)
{
    //获取拖放的文件名
    string[] fileNames = e.Data.GetData(DataFormats.FileDrop, true) as string[];
    if (fileNames.Length > 0)
    {
        //将拖放的文件保存到 FriendContent 的 PhotoUrl 中
        friendContent.PhotoUrl = fileNames[0];
        //将拖放的文件作为 Image 控件的路径显示
        photoSrc.Source = new BitmapImage(new Uri(friendContent.PhotoUrl));
    }
    //确保已经处理了事件，那么将不会调用基类的方法处理拖动操作
    e.Handled = true;
}
```

(2) 也可以单击按钮选择图片，当单击按钮时，将弹出一个选择图片的对话框，对应的实现代码如下所示：

```
private void btnChooseNewImage_Click(object sender, RoutedEventArgs e)
{
    Point topleft = this.PointToScreen(new Point(0, 0)); //获取屏幕左上角的相对坐标
    //获取显示风格
    DisplayStyle newDisplayStyle =
        (DisplayStyle)Application.Current.Properties["SelectedDisplayStyle"];
    //如果是三维模式，则高度偏移 20，否则不偏移
    double heightOffset = newDisplayStyle==DisplayStyle.ThreeDimension? 20 : 0;
    AddFriendImageWindow addImageWindow = new AddFriendImageWindow(); //打开选择图片窗口
    (addImageWindow as Window).Height = this.Height + heightOffset;
    (addImageWindow as Window).Width = this.Width;
    (addImageWindow as Window).Left = topleft.X;
    (addImageWindow as Window).Top = topleft.Y;
    addImageWindow.ShowDialog(); //设置了其位置和大小后，显示出来
    if (!string.IsNullOrEmpty(addImageWindow.SelectedImagePath))
    {
        //获取选择的图片，赋给 Image 进行显示
        friendContent.PhotoUrl = addImageWindow.SelectedImagePath;
        photoSrc.Source = new BitmapImage(new Uri(friendContent.PhotoUrl));
    }
}
```

## 5.4.8 保存联系人资料

当用户单击“保存联系人”按钮时，会将用户的输入保存到名为 MyFriends.xml 的 XML 文件中。在保存联系人信息时，首先获取指定的文件路径，判断文件是否存在指定的位置。如果存在，则追加数据；如果不存在，则创建一个新的 XML 文件再进行数据保存。

(1) 在文件 AddNewFriendControl.xaml.cs 中，文件名存在时的处理代码如下所示：

```
private void btnSave_Click(object sender, RoutedEventArgs e)
{
    //获取保存的文件名
    string xmlFilename =
        (string)Application.Current.Properties["SavedDetailsFileName"];
    //获取完整的 XML 文件路径
    string fullXmlPath = Path.Combine(Environment.CurrentDirectory, xmlFilename);
    bool allRequiredFieldsFilledIn = true; //判断所需要的字段是否填充
    allRequiredFieldsFilledIn =
```



```

IsEntryValid(txtFriendName) && IsEntryValid(txtEmail);
//判断 E-mail 是否正确
allRequiredFieldsFilledIn = IsEmailValid(txtEmail.Text);
if (allRequiredFieldsFilledIn) //如果所填的资料都正确
{
    if (File.Exists(fullXmlPath)) //如果也存在文件名
    {
        try
        {
            //如果当前没有 XML 且没有联系人在内存, 追加到文件
            if (FriendsList.Instance().Count == 0)
            {
                //使用对象初始化语法初始化并给 Friend 对象赋值
                Friend friend = new Friend
                {
                    ID = Guid.NewGuid(),
                    Name = friendContent.FriendName,
                    Email = friendContent.FriendEmail,
                    PhotoUrl = friendContent.PhotoUrl,
                    VideoUrl = friendContent.VideoUrl,
                    MusicUrl = friendContent.MusicUrl
                };
                //调用 XMLFileOperation 的 AppendToFile 方法追加文件
                XMLFileOperations.AppendToFile(fullXmlPath, friend);
                FriendsList.Instance().Add(friend); //添加到联系人列表中
                RaiseEvent(new RoutedEventArgs(FriendAddedEvent)); //引发路由事件
                friendContent.Reset();
                this.Reset();
                MessageBox.Show("成功保存联系人");
            }
            //否则只是更新内存中的联系人集合中的单个联系人
            //在应用程序关闭时将写入到磁盘
            else
            {
                FriendsList.Instance().Add(new Friend
                {
                    ID = Guid.NewGuid(),
                    Name = friendContent.FriendName,
                    Email = friendContent.FriendEmail,
                    PhotoUrl = friendContent.PhotoUrl,
                    VideoUrl = friendContent.VideoUrl,
                    MusicUrl = friendContent.MusicUrl
                });
                RaiseEvent(new RoutedEventArgs(FriendAddedEvent));
                friendContent.Reset();
                this.Reset();
                MessageBox.Show("成功保存联系人");
            }
        }
        catch
        {
            MessageBox.Show("更新联系人错误");
        }
    }
}

```

(2) 在文件 AddNewFriendControl.xaml.cs 中, 文件名不存在时的处理代码如下所示:

```

else //如果不存在 XML 文件名, 则创建一个新的文件并写入
{
    try
    {

```

```

        Friend friend = new Friend
        { //初始化 Friend 对象并赋值
            ID = Guid.NewGuid(),
            Name = friendContent.FriendName,
            Email = friendContent.FriendEmail,
            PhotoUrl = friendContent.PhotoUrl,
            VideoUrl = friendContent.VideoUrl,
            MusicUrl = friendContent.MusicUrl
        };
        //调用 CreateInitialFile 写入一个新的文件
        XMLFileOperations.CreateInitialFile(fullXmlPath, friend);
        FriendsList.Instance().Add(friend); //添加到联系人集合
        RaiseEvent(new RoutedEventArgs(FriendAddedEvent)); //触发事件
        friendContent.Reset(); //重置对象
        this.Reset(); //重置控件
        MessageBox.Show("成功保存联系人");
    }
    catch (Exception ex)
    {
        MessageBox.Show("保存联系人信息时产生错误");
    }
}
else
{
    MessageBox.Show("需要填充所有的字段, 或者是验证是否输入错误", "错误",
        MessageBoxButton.OK, MessageBoxImage.Error);
}
}

```

(3) “保存联系人”按钮调用了几个自定义的方法来实现其功能, 一个是验证输入框是否输入了值, 一个是验证电子邮件地址是否正确, 另外一个为重置控件的 Reset 方法。对应的实现代码如下所示:

```

private bool IsEntryValid(TextBox txtBox) //判断用户姓名字段是否输入为空, 否则变换背景色
{
    txtBox.Background = string.IsNullOrEmpty(txtBox.Text) ? Brushes.Red : Brushes.White;

    //返回是否为空的布尔值
    return !string.IsNullOrEmpty(txtBox.Text);
}

private void Reset() //重置控件的值, 使其返回初始化的空状态
{
    this.txtFriendName.Text = string.Empty;
    this.txtEmail.Text = string.Empty;
    this.photoSrc.Source = null;
    this.videoSrc.Source = null;
    this.musicSrc.Source = null;
    txtMusic.Visibility = Visibility.Visible;
    imgMusic.Visibility = Visibility.Hidden;
}

private bool IsEmailValid(string email)
{
    bool isValid = false;

    //指定验证的正则表达式
    string pattern = @"^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+\.[a-zA-Z]{2,5})$";
}

```

```

Regex regEx = new Regex(pattern); //实例化 Regex 对象
isValid = regEx.IsMatch(email); //进行验证工作

//根据验证的结果设置电子邮件框的背景色
txtEmail.Background = !isValid? Brushes.Red : Brushes.White;

//返回验证结果的布尔值
return isValid;
}

```

(4) 在添加联系人的窗体中, 使用了 FriendContent 单件类来临时地保存联系人信息。

这里之所以使用了单件模式, 与主窗口的 3D 动画有关, 在主窗体中, 分别放置了 AddNewFriendControl 控件和 ViewAllUsersControl 控件, 而在用于 3D 动画的 ItemsControl 的 DataTemplate 中, 又包含了 AddNewFriendControl 和 ViewAllUsersControl 这两个用户控件。为了确保数据在两个拷贝间的同步, 引入了单件模式。

FriendContent 类的代码如下所示:

```

public class FriendContent //用于临时保存联系人信息的单件类
{
    public string FriendName { get; set; } //名字
    public string FriendEmail { get; set; } //E-mail
    public string PhotoUrl { get; set; } //照片路径
    public string VideoUrl { get; set; } //视频路径
    public string MusicUrl { get; set; } //音乐路径
    private static FriendContent instance; //实例变量

    private FriendContent() //私有构造函数, 防止用户直接实例化
    {
    }

    public void Reset() //调用 Reset 方法清空字段
    {
        FriendName = string.Empty;
        FriendEmail = string.Empty;
        PhotoUrl = string.Empty;
        VideoUrl = string.Empty;
        MusicUrl = string.Empty;
    }

    public static FriendContent Instance() //实例化 FriendContent 的公共静态方法
    {
        if (instance == null)
        {
            instance = new FriendContent();
        }
        return instance;
    }
}

```

## 5.5 系统测试

视频讲解 光盘: 视频\第5章\系统测试.avi

运行本项目后, 将首先按照默认样式显示, 如图 5-5 所示。



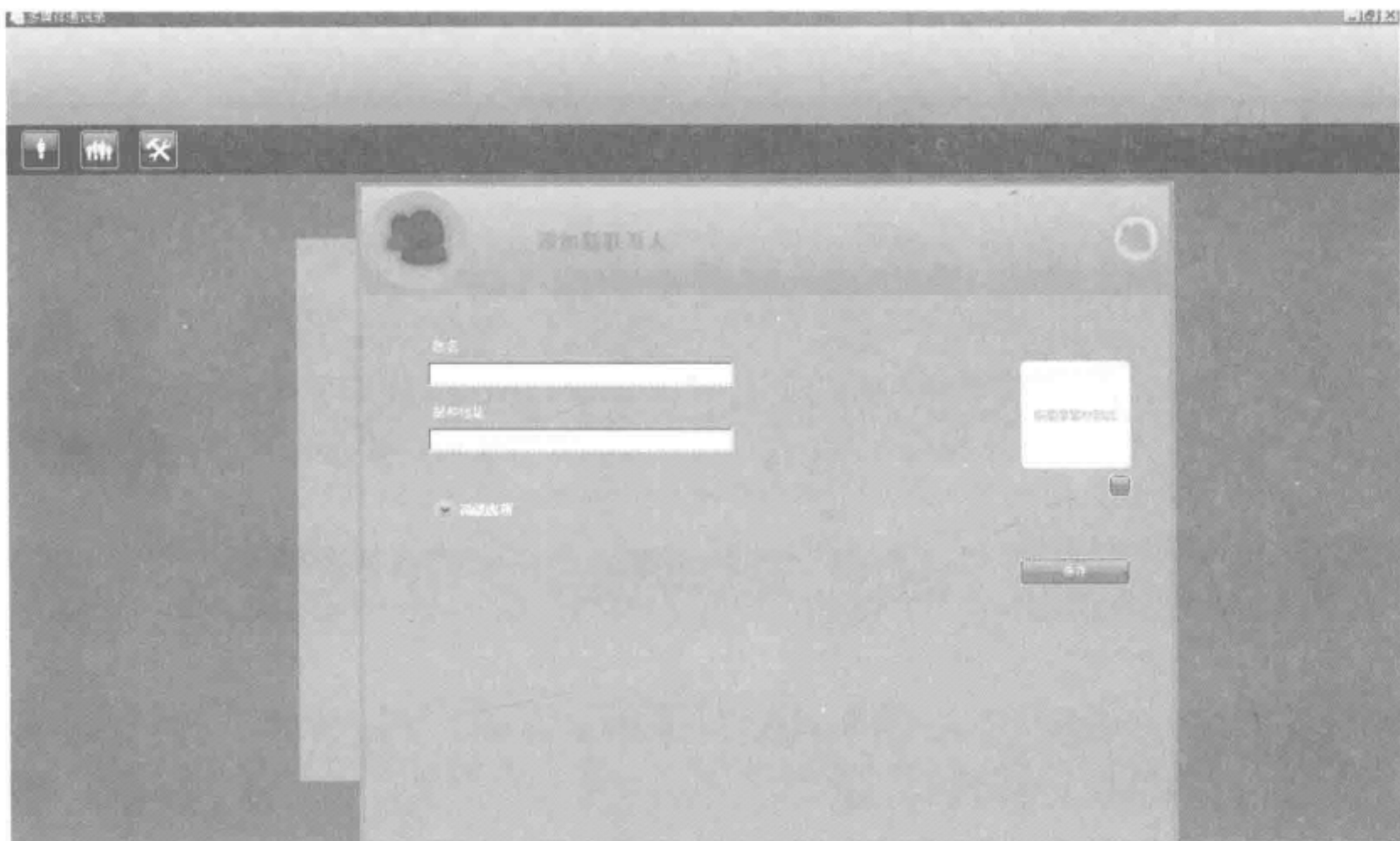


图 5-5 初始效果

高级选项界面的效果如图 5-6 所示。

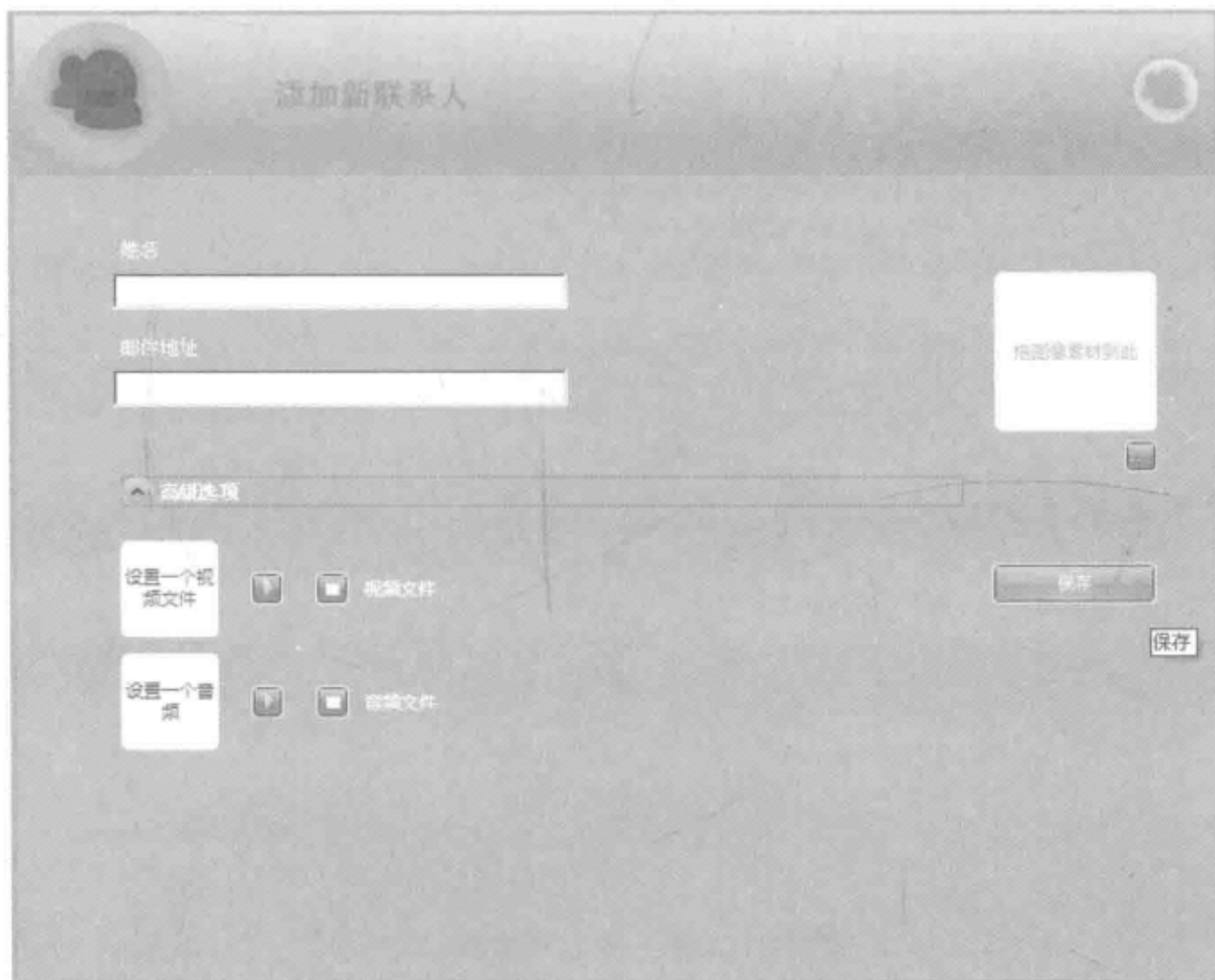


图 5-6 高级选项界面的效果

选择图片界面的效果如图 5-7 所示。



图 5-7 选择图片界面的效果

查看所有联系人界面的效果如图 5-8 所示。

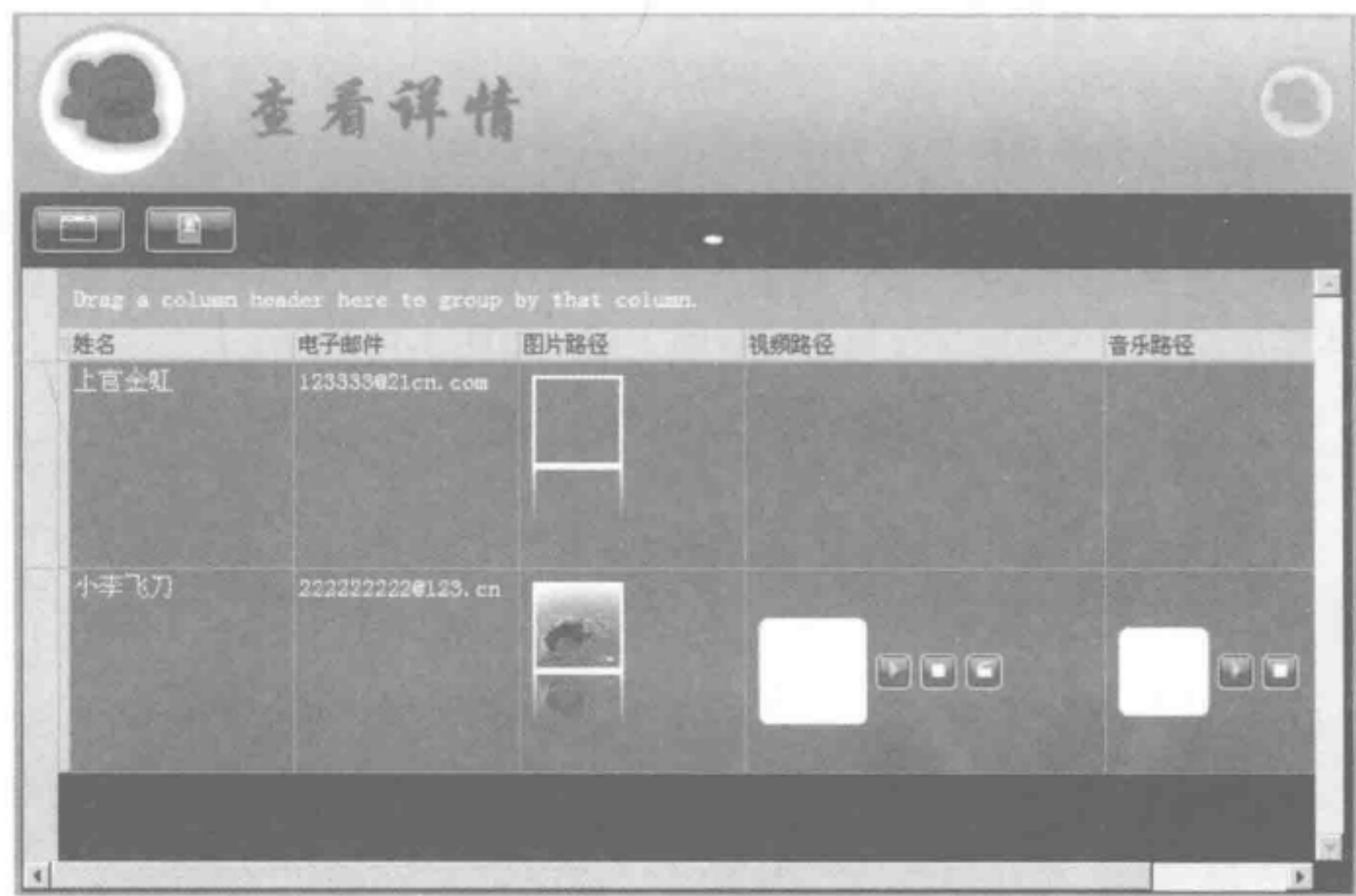


图 5-8 查看所有联系人界面的效果

联系人选项窗口界面的效果如图 5-9 所示。

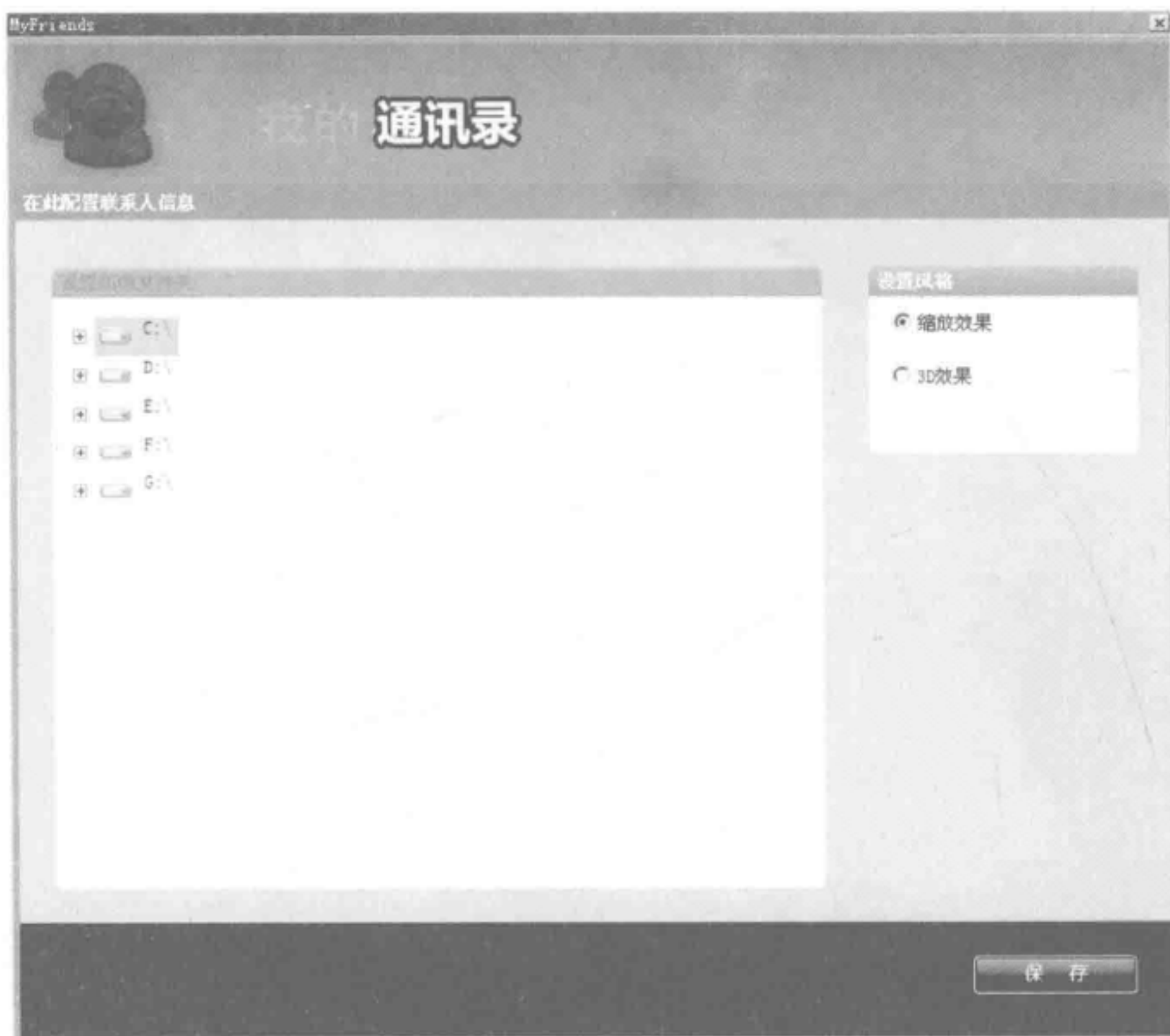


图 5-9 联系人选项窗口界面的效果

到此为止，整个通信录系统的实现过程全部讲解完毕。通过本项目的具体实现，可以对开发实现流程有一个全新的认识。

很多初学者在遇到项目时，总是在粗略估计后，立即投入到代码编写工作中，这样，后期往往会出现很多错误。对于小项目来说，修改的工作量也许不是很多。但想象一下如果在大项目中，会是很恐怖的。所以提前搞好规划很重要。通过这个项目，可以总结出此项目的基本实现流程，里面标注了每个过程的工期，如图 5-10 所示。

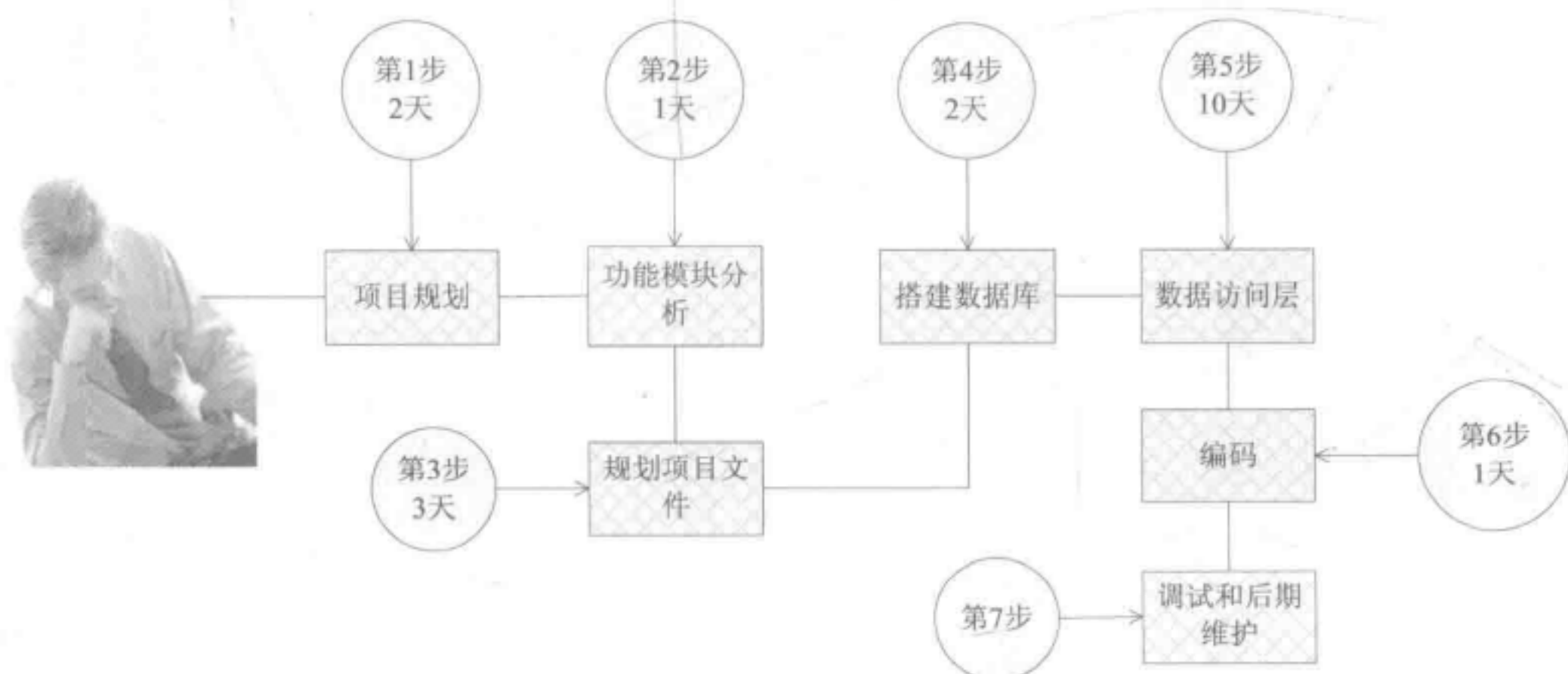


图 5-10 本项目的基本流程



## 第 6 章 在线点歌系统

高州项目开发

在现代都市的夜生活中，KTV 和练歌房随处可见。因为它们能够帮助人们解除工作疲劳，所以深受广大消费者的喜爱。作为软件开发工作者来说，开发一个方便管理的点歌系统，会有很好的市场前景。

本章讲解的这个点歌系统是作者本人开发的一个颇为自豪的系统，因为这个系统由始至终严格遵循了面向对象技术，而且充分考虑了可扩充性和可升级性的原则，准备了大量的接口，便于后期的升级工作。读者可以以此为基础进行扩展，迅速打造一个适合自己项目的点歌系统。希望读者重点关注这部分内容，因为这些内容正是初学者们在学习过程中经常遇到的最大困惑。



### 赠送的超值电子书

- 051 结构方法
- 052 方法重载介绍
- 053 方法重载的具体应用
- 054 使用方法 Main()
- 055 递归
- 056 C++中的方法重载和 C#中的方法重载
- 057 方法和内存分配
- 058 参数的传递问题
- 059 何时用静态函数，何时用实例函数
- 060 建议优先考虑将基类型作为参数传递

## 6.1 架构中的可扩展性

视频讲解 光盘：视频\第6章\架构中的可扩展性.avi

作为系统设计人员或系统架构师来说，除了关注产品的功能和性能外，还应多考虑系统结构的可扩展性。在程序开发领域中，影响一款软件的质量的属性有如下两类。

- 功能性的质量属性：正确性、健壮性和可靠性。
- 非功能的质量属性：性能、易用性、清晰性、安全性、可扩展性、兼容性和可移植性。

无论是作为一名普通的程序员，还是项目经理或系统架构师，都要确保自己的软件具有可扩展性。只有这样，才能使自己的作品满足客户的不同需求。在本节的内容中，将与读者一起，探讨设计可扩展性程序的技巧。

### 6.1.1 软件的发展是一个不断完善的过程

对于任何一款软件来说，在最开始做架构设计时，都是针对某一方面问题而推出的。但是，随着时间的推移和用户需求的变化，这款软件必须经过升级、改造之后才能继续存在于市场中。在现实世界中，大型的著名软件都提供了升级功能，以便及时地改善这款软件的功能。软件升级，通常是指软件前期版本发行后，又通过对程序的修改或加入新的功能后，再以补丁的形式发布的方式。当有重大功能变化时，软件通常会以全新版本的形式发布出来。

软件为什么需要可扩展呢？主要原因有如下两点：

- 软件系统的功能往往是变化的，而且新增需求越来越多。
- 实现这些新增需求时，不可能推倒重新设计(否则代价太大了，除非是很小的系统)，而且希望代价越小越好。

例如 QQ 由最初的简单文字聊天，发展到了现在的视频聊天、语音聊天、QQ 空间、QQ 邮箱、QQ 微博等集成功能。

### 6.1.2 赢在项目本身——让程序具有更好的可扩展性

针对软件更新和升级的需求，要求架构师和开发者们的作品具有可扩展性。可扩展性是指软件扩展新功能的容易程度。可扩展性越好，表示软件适应“变化”的能力越强。

Bertrand Meyer 在 1988 年提出了著名的“开放-封闭”原则(The Open-Closed Principle, OCP)，此原则提出了如下两条准则。

- **扩展是开放的：**模块的行为是可以扩展的，当用户的需求发生变化时，可以对模块进行扩展，使其具有满足用户新需求的行为，即可以改变模块的功能。
- **更改是封闭的：**对模块行为进行扩展时，不必改动原有模块的源代码或者二进制代码(但可以新增)。模块的二进制可执行文件，无论是 DLL 文件还是 EXE 文件，都无须改动。

由此可见，“开放-封闭”原则是说软件实体(类、模块、函数等)应该可以扩展，但是



不可修改。这其实说明了两个特征，一个是“对于扩展是开放的(Open for extension)”，另一个是“对于更改是封闭的(Closed for modification)”。

在做任何系统时，都不能指望需求在一开始就完全确定。怎样的设计才能面对需求的改变却可以保持相对稳定，从而使得系统可以在第一个版本以后不断推出新的版本呢？“开放-封闭”原则就是这个问题的答案，软件设计要容易维护又不容易出问题的最好方法就是多扩展、少修改。

在现实项目开发过程中，无法在初期架构设计时就需求的各种变化完全考虑到。只能尽量将这个类设计得足够好。等来了新的需求，再通过增加一些类来完成，原来的代码能不动则不动。

绝对的对修改关闭是不可能的。无论模块是多么的“封闭”，都会存在一些无法对之封闭的变化。既然不可能完全封闭，设计人员就必须对他设计的模块应该对哪种变化封闭做出选择。他必须先猜测出最有可能发生的变化种类，然后构造抽象来隔离那些变化。

很多时候，我们无法猜测到需求的变更，但我们可以在发生小变化时，及早地想办法应对发生更大变化的可能。也就是等到变化发生时立即采取行动。正所谓“不要在同一个地方摔两次跤”。在我们最初编写代码时，假设变化不会发生。当发生变化时，我们就创建抽象来隔离以后发生的同类变化。

比如一个加法程序，可以很快地在一个 client 类中完成。但此时需求变化，需要增加减法功能。于是发现，增加功能需要修改原来这个类，这就违背了“开放-封闭”原则。于是考虑重构程序，增加一个抽象的运算类，通过继承、多态等面向对象的手段来隔离加法、减法与 client 类的耦合，这样还可以应对以后的变化。如果还需要增加乘除功能，就不需要更改 client 以及加减法的类了，而是增加乘除子类即可。即面对需求，对程序的改动是通过增加新代码进行的，而不是更改现有的代码。这就是“开放-封闭”原则的中心思想。

由此可见，“开放-封闭”原则是面向对象设计的核心所在。遵循这个原则，可以带来面向对象技术所声称的巨大好处，也就是可维护、可扩展、可复用、灵活性好。开发人员应该仅对程序中呈现出频繁变化的那些部分做出抽象，而对于应用程序中的每个部分都刻意地进行抽象就不是一个好主意，所以拒绝不成熟的抽象和进行抽象本身同样重要。

## 6.2 系统分析

视频讲解  光盘：视频\第6章\系统分析.avi

在本节的内容中，将首先讲解点歌系统的市场背景和模块划分，为步入后面的具体编码工作打下基础。这一部分内容十分重要，读者们一定不要忽视，因为这部分工作决定了本项目运营的成败。

### 6.2.1 背景介绍

由于计算机硬件技术取得了长足的进步，绚丽的多媒体应用已经走进了普通大众的生活中。随着近些年网络技术的发展，多媒体应用已经由单机走向了网络。视频播放也采用了网络数字流，即视频点播系统。这种系统以快速、灵活的特点，逐渐得到了各领域用户



的青睐，并逐渐成为时尚潮流。VOD 技术已经逐渐被用于政府、教育、智能小区、宾馆、KTV 等领域。

目前，市场上已有存在不少 KTV 软件，经过考察和比较，发现目前已有的系统存在如下所示的问题：

- 成本偏高，市场竞争要求尽可能地降低成本。
- 系统可靠性不高，客户端对服务器的依赖性过高。
- 软件易用性不好，上手慢。
- 软件稳定性不够高，容易发生死机等现象。
- 软件有些方面设计得不合理，例如对最常用的消除原声操作复杂，应该做到自动实现。

### 6.2.2 需求分析

XX 集团是本地的一家大型 KTV 连锁机构，在本市拥有 8 家 KTV，3000 多个包间。为应对市场发展需求，与国际娱乐潮流接轨，XX 集团决定开发一款自主品牌的点歌系统。这样不但可以扩大对自主品牌的宣传，而且长期下来，能节省购买第三方平台的费用。

经过集团内部对当前市场的分析，要求点歌系统具备下面的功能：

- 选择歌星点歌。
- 根据拼音点歌。
- 根据歌名点歌。

XX 集团的预算为 15~20(万元)，工期要求为两个月。

### 6.2.3 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，XX 软件开发公司项目部特意编制了一份可行性研究报告，具体内容如下所示。

#### 1. 引言

##### (1) 编写目的

为了给企业的决策层提供是否进行项目实施的参考依据，现以文件的形式分析项目的风险、项目需要的投资和效益。

##### (2) 背景

XX 娱乐集团是本地的一家大型 KTV 连锁机构，在本市拥有 8 家 KTV，3000 多个包间。为应对市场发展需求和与国际娱乐潮流接轨，XX 集团决定开发一个自主品牌的点歌系统。现委托我公司开发来开发，项目名称暂定为：XX 点歌系统。

#### 2. 可行性研究的前提

##### (1) 要求

要求系统具有选择歌星点歌、根据拼音点歌、根据歌名点歌等功能。

##### (2) 目标

系统主要目标是可以方便消费者快速点播自己喜欢的歌曲，打造一个愉悦的欢唱氛围。

### (3) 条件、假定和限制

要求整个项目在立项后的两个月内交付用户使用。系统分析人员需要 3 天内到位, 用户需要 2 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 5 天时间确认需求分析。那么程序开发人员需要在 52 天的时间内进行系统设计、程序编码、系统测试和程序调试工作。其间还包括了员工每周的休息时间。

### (4) 评价尺度

根据企业的要求, 系统应能按照规定正确地根据使用者的要求提供点歌功能。因为系统的信息数量需求不大, 系统应能快速、有效地对数据库数据进行操作。

## 3. 投资及效益分析

### (1) 支出

由于系统规模比较小, 而客户要求的项目周期不是很短(两个月), 因此公司决定只安排 3 人投入到其中。公司将为此支付 6 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 2 万元的资金, 累计项目投入需要 10 万元资金。

### (2) 收益

XX 公司提供项目资金 15~20 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, 公司最低可以获得 5 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 3 个人、50 天最低获利 5 万元, 比较可观。在公司发展上可以储备项目开发的经验和资源。因此认为该项目可以开发。

## 6.2.4 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, 结合单位的实际情况, 设计项目计划书如下。

### 1. 引言

#### (1) 编写目的

为了保证项目开发人员按时保质地完成预订目标, 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。

#### (2) 背景

XX 点歌系统是由 XX 娱乐集团委托我公司开发的一款点播软件, 项目周期为两个月。项目背景规划如表 6-1 所示。

表 6-1 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           |
|---------|---------|-------|------------------|
| XX 点歌系统 | XX 娱乐集团 | 吴总    | 项目开发部门<br>项目测试部门 |

## 2. 概述

### (1) 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。XX 点歌系统的项目目标如下：消费者可以使用此系统方便点歌，点歌时可以根据个人喜好的模式进行，例如歌名点歌、歌星点歌、拼音字母点歌。

### (2) 应交付成果

在项目开发完成后，交付内容有编译后的社区视频监控系统、系统数据库文件和系统使用说明书。系统安装后，进行系统无偿维护与服务 6 个月，超过 6 个月则进行有偿维护与服务。

### (3) 项目开发环境

操作系统为 Windows XP、Windows 2000、Windows 2003 或 Windows 7，数据库采用 Access 2007，开发工具为 Visual Studio。

### (4) 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用户进行验收，验收的主要依据为需求规格说明书。

## 3. 项目团队组织

### (1) 组织结构

为了完成点歌系统的项目开发，公司组建一个临时的项目团队，由项目经理、系统分析员、软件工程师和测试人员构成，组织结构如图 6-1 所示。

### (2) 人员分工

为了明确项目团队中每个人的任务，现制定人员分工表，如表 6-2 所示。

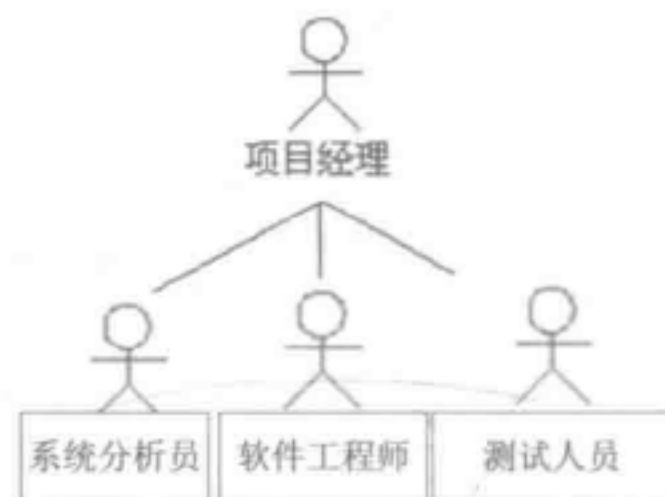


图 6-1 项目团队的组织结构

表 6-2 人员分工表

| 姓名 | 技术水平      | 所属部门  | 角 色   | 工作描述                                                |
|----|-----------|-------|-------|-----------------------------------------------------|
| 吴某 | MBA       | 项目开发部 | 项目经理  | 负责项目的审批、决策的实施以及前期分析、策划、项目开发进度的跟踪、项目质量的检查以及系统功能分析与设计 |
| 刘某 | 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                                           |
| 王某 | 初级系统测试工程师 | 项目测试部 | 测试人员  | 对软件进行测试、编写软件测试文档                                    |



## 6.3 系统模块划分

视频讲解 光盘：视频\第6章\系统模块划分.avi

虽然客户的最初要求并不高，但是整个实现过程却并不简单。其实，对于一款点歌系统来说，其基本的功能模块并不难掌握。经过自己去对 KTV 系统做体验，可以迅速做出分析并得出结论，一个典型点歌系统的构成模块如下所示。

### (1) 歌星点歌

根据歌星的名字查询并选择歌曲。

### (2) 数字点歌

根据歌曲的名称和编号进行查询并选择歌曲。

### (3) 拼音点歌

根据歌曲名称的每一个汉字拼音的首字母来选择歌曲。

### (4) 歌名点歌

根据歌曲的名称来检索歌曲。

### (5) 登录验证

管理员可以登录系统，来对曲库进行管理，只有合法的用户才能使用点歌系统。

上述功能模块的具体运行流程如图 6-2 所示。



图 6-2 点歌系统的运行流程

## 6.4 设计数据库

视频讲解 光盘：视频\第6章\设计数据库.avi

在设计数据库时，一定要考虑客户的因素，例如系统的维护性和造价。就本项目而言，因为客户要求整个维护工作要尽量简单，并且要求造价低，所以在此可以选择使用一款轻量级的数据库产品。在众多数据库产品中，建议使用微软的 Access。

这里,肯定很多读者会提出疑问:SQL Server Express 和 Access 都是免费的,都可以作为轻量级项目的数据库,为什么本项目要选择使用 Access 呢?

的确,SQL Server Express 和 Access 都是微软的产品,并且两者都是免费的。读者要想了解在此选择 Access 的原因,不妨先对两者进行一个详细的对比。

#### (1) SQL Server Express

该工具用于小型应用程序,其数据库引擎是 Microsoft 的 SQL Server 数据库引擎的一部分。该版本 2005 支持很多完整 SQL Server 版的高级功能,如存储过程、视图、函数、CLR 集成、打印及 XML 支持等。然而,它仅仅是一个数据库引擎,而不像 Access 那样集成了接口开发工具。任何前台应用程序的开发都需要开发程序来处理,例如免费的 C# Express 工具。此外,微软还创建一个很好的 SQL Server Management Studio 的 Express 版本,这个版本可以管理 SQL Server Express 数据库引擎。

虽然 SQL Server Express 是免费的,但如果想用它实现一个解决方案,则需要提前注册该产品。SQL Server Express 只是 SQL Server 的精简版本,并不包含所有的内置接口设计工具,因此,在使用它来解决各类问题时,往往要比使用 SQL Server 更加复杂。

#### (2) Microsoft Access

如果开发的应用程序非常小,例如是简单的登录系统,此时可以选择 Access。

Access 拥有内置的窗体、报表及其他功能项。可以使用它为后台数据库表格构建用户接口。Access 的大部分可编程对象都拥有一个很好的向导,这对初学者来说十分方便。最重要的是,用它开发一个小系统的时间相当短。

与 SQL Server Express 相比,Access 的突出优势是它包含在 Microsoft Office 的产品系列中,这样,其开发成本相对于 SQL Server Express 将有显著的降低。另外,Access 更加容易使用,便于菜鸟级用户的使用和维护。

综上所述,对于大多数项目来说,如果你的应用程序非常小,并且同一时刻只要求很少用户访问,使用 Access 将是一个不错的选择,并且 Access 在降低成本方面也表现得更加出色。当程序涉及到的数据量较大,并且同一时刻访问的用户较多时,建议选择 SQL Server Express。其实,无论选择它们中的哪一款,都必须弄清楚自身的开发经验。建议在面对不是很复杂的项目时,或者在一个系统的应用初期,都首先选择使用 Access,在后期,随着数据量的增多,再升级为 SQL Server。因为两者都是微软的产品,所以相互之间的转换工作非常简单。

再回到本项目,因为本点歌系统是为 KTV 服务的,客户特意要求维护简单和造价低,所以本项目选择使用 Access 即可满足基本要求。

### 6.4.1 数据库概念结构设计

决定使用 Access 后,首先创建一个名为“db\_KTV.mdb”的数据库,然后根据系统需求开始规划出整个系统需要的实体。

根据前面的模块划分介绍,总结出本项目的实体有:明星信息实体、歌曲信息实体、歌曲类型实体、管理员信息实体。

#### (1) 明星信息实体的 E-R 图如图 6-3 所示。

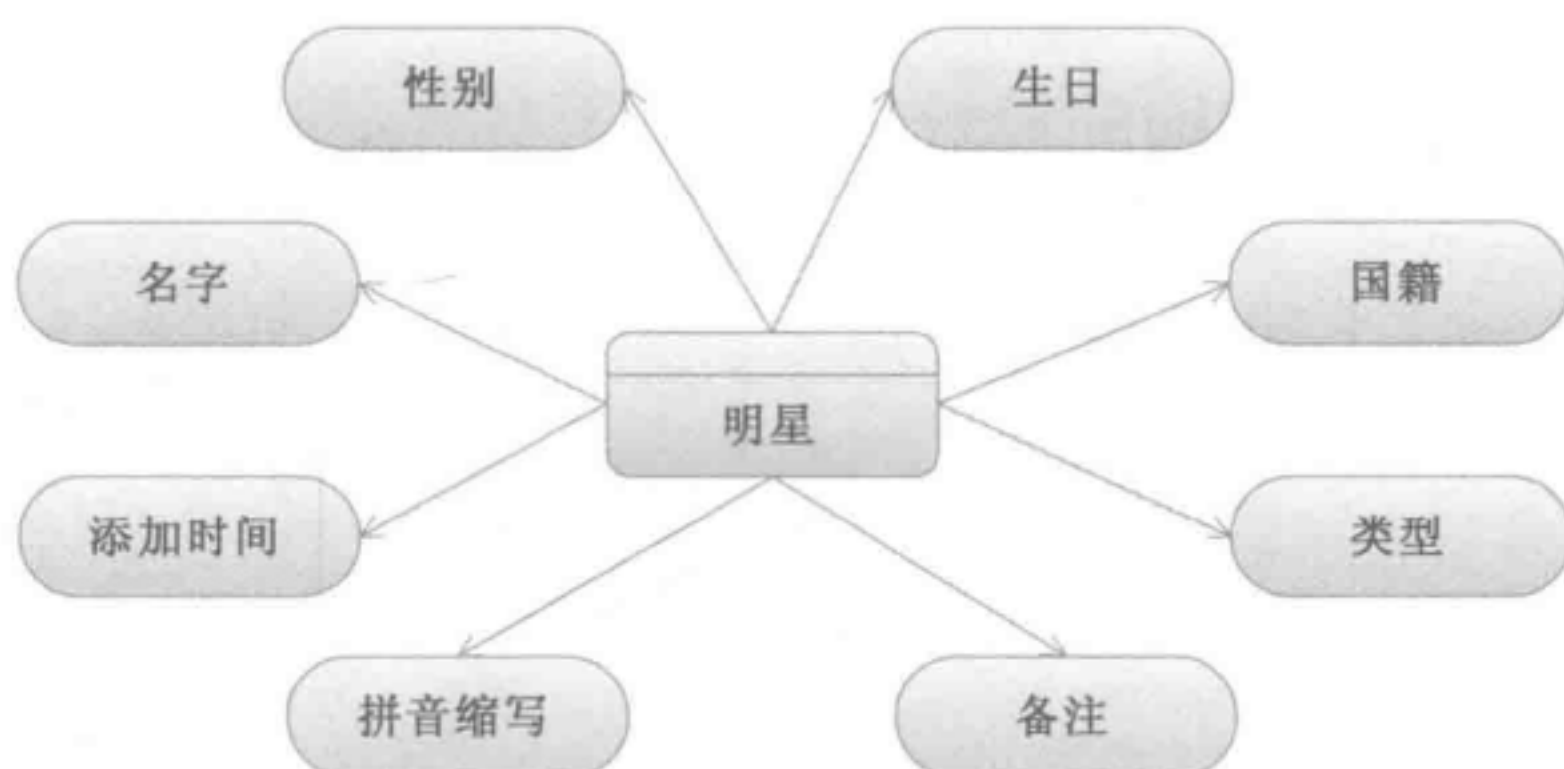


图 6-3 明星信息实体的 E-R 图

(2) 歌曲信息实体的 E-R 图如图 6-4 所示。



图 6-4 歌曲信息实体的 E-R 图

(3) 管理员信息实体的 E-R 图如图 6-5 所示。



图 6-5 管理员信息实体的 E-R 图

(4) 歌曲类型信息实体的 E-R 图如图 6-6 所示。



图 6-6 歌曲类型信息实体的 E-R 图



### 6.4.2 数据库逻辑结构的设计

在接下来的内容中，开始讲解 Access 数据库 db\_KTV.mdb 中各个表的具体逻辑结构。

(1) 表 tb\_authorinfo 用于保存明星的基本信息，具体设计结构如图 6-7 所示。

| 字段名称           | 数据类型  | 说明   |
|----------------|-------|------|
| authorId       | 文本    | 编号   |
| authorName     | 文本    | 明星名字 |
| authorSex      | 文本    | 明星性别 |
| authorbirthday | 日期/时间 | 出生日期 |
| authorGenre    | 文本    | 国籍   |
| authorcompany  | 文本    | 所属公司 |
| authorRecma    | 文本    | 备注   |
| authorzjm      | 文本    | 拼音码  |
| RdateTime      | 日期/时间 | 录入时间 |

图 6-7 表 tb\_authorinfo

(2) 表 tb\_computer 用于保存系统管理员的信息，具体设计结构如图 6-8 所示。

| 字段名称         | 数据类型 | 说明  |
|--------------|------|-----|
| cmp_ID       | 文本   | 编号  |
| cmp_name     | 文本   | 用户名 |
| cmp_Paww     | 文本   | 密码  |
| cmp_DataTime | 文本   | 时间  |
| cmp_Falg     | 文本   | 标记  |

图 6-8 表 tb\_computer

细心的读者应该会发现，在图 6-8 给出的管理员信息表 tb\_computer 中，有一个名为 cmp\_Flag 的字段。这个字段与项目本身并没有任何关系，在表面看来是多余的。其实，这个字段是有意而为之的，原因是出于系统安全方面的考虑。因为数据库技术是动态网站的根本，正是因为它的重要性，所以数据库的安全性就成为我们当务之急要解决的问题。

在数据库安全问题上，多数都采用用户标识机制。用户标识是指用户向系统出示自己的身份证明，最简单的方法是输入用户 ID 和密码。标识机制用于唯一标志进入系统的每个用户的身份，因此必须保证标识的唯一性。而鉴别是指系统检查和验证用户的身份证明，用于检验用户身份的合法性。标识和鉴别功能保证了只有合法的用户才能存取系统中的资源。由于数据库用户的安全等级是不同的，因此分配给他们的权限也是不一样的，数据库系统必须建立严格的用户认证机制。身份的标识和鉴别是 DBMS 对访问者授权的前提，并且通过审计机制，使 DBMS 有保留追究用户行为责任的证据的能力。功能完善的标识与鉴别机制也是访问控制机制有效实施的基础，特别是在一个开放的多用户系统的网络环境中，识别与鉴别用户是构筑 DBMS 安全防线的一个重要环节。

(3) 表 tb\_dictionary 用于保存系统歌曲类型的信息，具体设计结构如图 6-9 所示。

| 字段名称     | 数据类型 | 说明     |
|----------|------|--------|
| codeID   | 文本   | 编号     |
| codName  | 文本   | 信息类别名称 |
| codeReam | 文本   | 备注     |

图 6-9 表 tb\_dictionary

(4) 表 tb\_musicinfo 用于保存系统的歌曲信息，具体设计结构如图 6-10 所示。

|                |       |        |
|----------------|-------|--------|
| Music_code     | 文本    | 歌曲编号   |
| MusicC_name    | 文本    | 歌曲名称   |
| Music_author   | 文本    | 作者     |
| Music_Kind     | 文本    | 歌曲类型   |
| Music_chinse   | 文本    | 语种     |
| Music_filepath | 文本    | 文件名称路径 |
| Music_Ping     | 文本    | 拼音     |
| Music_date     | 日期/时间 | 日期     |
| Music_falg     | 数字    | 删除标记   |

图 6-10 表 tb\_musicinfo

## 6.5 设计公共类

视频讲解 光盘：视频\第6章\设计公共类.avi

作为整个项目的核心和基础，本项目的公共类分为如下 3 个部分。

- (1) 数据库连接。
- (2) 歌曲信息参数。
- (3) 操作歌曲信息的方法。

在接下来的内容中，将详细讲解设计公共类的具体流程。

### 6.5.1 数据库连接

定义 getConnection 类，此类封装了连接数据库的方法，因为只是对数据库的操作，所以需要引入一些与数据库相关的命名空间。实现文件是 getConnection.cs，对应的代码如下所示：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.OleDb;
using System.Windows.Forms;

namespace KTV.KTVclass
{
    class getConnection
    {
        public OleDbConnection OledCon()
        {
            //创建连接数据库的字符串
            string reportPath = Application.StartupPath.Substring(0,
                Application.StartupPath.Substring(0,
                    Application.StartupPath.LastIndexOf(@"\")).LastIndexOf(@"\"));
            reportPath += @"DataBase\db_KTV.mdb";
            string ConStr = "Provider=Microsoft.Jet.OLEDB.6.0;Data source=" + reportPath;
            //创建 OleDbConnection 对象
            OleDbConnection con = new OleDbConnection(ConStr);
            //con.Open();
            return con;
        }
    }
}
```

## 6.5.2 歌曲信息参数

在文件 tb\_musicinfo.cs 中定义类 tb\_musicinfo, 此类是歌曲信息实体类, 功能是传递歌曲信息表有关的参数实体。文件 tb\_musicinfo.cs 的实现代码如下所示:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace KTV.KTVclass
{
    public class tb_musicinfo
    {
        //歌曲编号
        private string Music_code;
        public string strMusic_code
        {
            get{ return Music_code; }
            set{ Music_code=value; }
        }
        //歌曲名称
        private string MusicC_name;
        public string strMusicC_name
        {
            get{ return MusicC_name; }
            set{ MusicC_name=value; }
        }
        //演唱歌手
        private string Music_author;
        public string strMusic_author
        {
            get{ return Music_author; }
            set{ Music_author=value; }
        }
        //歌曲类别
        private string Music_Kind;
        public string strMusic_Kind
        {
            get{ return Music_Kind; }
            set{ Music_Kind=value; }
        }
        //语言
        private string Music_chinse;
        public string strMusic_chinse
        {
            get{ return Music_chinse; }
            set{ Music_chinse=value; }
        }
        //歌曲文件路径
        private string Music_filepath;
        public string strMusic_filepath
        {
            get{ return Music_filepath; }
            set{ Music_filepath=value; }
        }
        //歌曲名缩写
        private string Music_Ping;
        public string strMusic_Ping
        {

```



```

        get{ return Music_Ping; }
        set{ Music_Ping=value; }
    }
    //录入时间
    private DateTime Music_date;
    public DateTime daMusic_date
    {
        get{ return Music_date; }
        set{ Music_date=value; }
    }
    //删除标记
    private int Music_falg;
    public int intMusic_falg
    {
        get{ return Music_falg; }
        set{ Music_falg=value; }
    }
}

```

从上述代码的实现过程来看,类 `tb_musicinfo` 只是起了一个重复描述数据库中歌曲信息表的作用。其实,我们完全不用特意编写这个类,直接在需要时读取数据库中的歌曲信息字段即可,所以很多初学者认为这是多此一举的。相信在很多初学者眼中,对于本项目中的歌曲信息操作,都会认为只须直接连接数据库并进行数据读取即可,不用必须使用 `tb_musicinfo` 这个实体类从中间“过滤”一遍。遗憾的是,这只是大多数初学者的幼稚想法而已。对于一门面向对象的编程语言,如果想让自己的程序更具有可扩展性和健壮性,就应尽量使用基于多层架构的实体类。但是用的时候也并不是漫无目的,随便使用。对于数据库项目来说,通常关系数据库中的每一个表都可以抽象为一个类,数据表中的每条数据都可以看作是该类的一个实例。假如用户表 `Users` 有三个字段: `id(varchar2)`、`name(varchar2)`、`age(number)`,用户理所当然要抽象成一个类(这是基于面向对象设计的),而该类的数据成员应该是 `Users` 表中的三个字段。至于用户类的方法,总少不了添加新用户、修改用户信息、查询用户等。但是这个时候,问题也随之而来,如果有  $N$  个表,就需要对这  $N$  个表进行操作,写  $N$  个类。虽然它们的数据成员(其实就是数据表字段)不一样,但是却有着一些相同的方法(增、删、改、查),这是不是要给每一个类一一地写这些实现呢?如果  $N > 10$ ,你是否会感觉很累呢?以后如果还有类似的项目,你又要重新再写  $N$  个类,如果这样的话,程序员会疯掉的。所以在写程序之前,一定要仔细地想一想,怎样编写代码才能表现得更加精巧和更加容易复用。在此建议将经常用到的并且需要跨层处理的表设计为实体类,这样就可以将这个表中的数据当作对象来用,负责各层之间的数据传输。这样我们不经意间,就实现了数据的表映射处理。对于初学者来说,建议平时多练习实体类的知识,因为它涉及了映射、数据封装、设计模式的知识,对于向深层次的发展和套用有很大的帮助。在此重点讲解上述实体类 `tb_musicinfo` 的目的,是让读者养成使用实体类的习惯。

### 6.5.3 歌曲信息操作处理

定义 `tbMusicinfoMethod` 类,用于实现对系统内歌曲信息的各种操作处理。主要包括对歌曲信息的添加、修改和删除等操作,每种操作是通过对应的方法实现的。实现文件是 `tbMusicinfoMethod.cs`,下面开始讲解此文件的具体实现流程。

(1) 引入命名空间，声明 3 个操作数据库的公共对象。对应代码如下所示：

```
using System;
using System.Collections.Generic;
using System.Text;

using System.Windows.Forms;
using System.Data.OleDb;
//using System.Configuration;
namespace KTV.KTVclass
{
    public class tbMusicinfoMethod
    {
        OleDbCommand oledcmd = null;
        OleDbConnection oledCon = null;
        OleDbDataReader oleRed = null;
```

(2) 定义方法 tbMusicinfoAdd(tb\_musicinfo tb\_aut)，实现向系统内添加新的歌曲信息。返回值是 int 类型，返回值是 1 时表示添加成功，返回 0 表示添加失败。对应代码如下所示：

```
public int tbMusicinfoAdd(tb_musicinfo tb_aut)
{
    int intResult = 0;
    try
    {
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();
        string strAdd = "insert into tb_musicinfo values ( ";
        strAdd += "'" + tb_aut.strMusic_code + "','" + tb_aut.strMusicC_name + "','" +
        strAdd += "'" + tb_aut.strMusic_author + "','" + tb_aut.strMusic_Kind + "','" +
        strAdd += "'" + tb_aut.strMusic_chinse + "','" + tb_aut.strMusic_filepath + "','" +
        strAdd += "'" + tb_aut.strMusic_Ping + "','" + tb_aut.daMusic_date + "','" +
        strAdd += "'" + tb_aut.intMusic_falg + "'" + ")";
        oledcmd = new OleDbCommand(strAdd, oledCon);
        intResult = oledcmd.ExecuteNonQuery();
        return intResult;
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.Message.ToString());
        return intResult;
    }
}
```

(3) 定义方法 tbMusicinfoID()，功能是自动生成歌曲的编号。返回值是 int 类型，值是数据库内歌曲表中的最大编号值加 1。方法 tbMusicinfoID() 的实现代码如下所示：

```
public int tbMusicinfoID()
{
    int intResult = 0;
    try
    {
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();
        string strAdd = "select Max(Music_code) from tb_musicinfo";
        oledcmd = new OleDbCommand(strAdd, oledCon);
        oleRed = oledcmd.ExecuteReader();
        oleRed.Read();
```

```

if (oleRed.HasRows)
{
    if (oleRed[0].ToString() == "")
    { intResult = 1; }
    else
    {
        intResult = Convert.ToInt32(oleRed[0].ToString()) + 1;
    }
}
return intResult;
}
catch (Exception ee)
{
    MessageBox.Show(ee.Message.ToString());
    return intResult;
}
}

```

(4) 定义方法 `tbMusicinfoFill(object obj)`，功能是将歌曲表中的所有数据填充到指定的 `ListView` 控件中。方法 `tbMusicinfoFill(object obj)` 的实现代码如下所示：

```

public void tbMusicinfoFill(object obj)
{
    try
    {
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();
        string strAdd = "select * from tb_musicinfo ";
        oledcmd = new OleDbCommand(strAdd, oledCon);
        oleRed = oledcmd.ExecuteReader();
        ListView lv = (ListView)obj;
        lv.Items.Clear();
        while (oleRed.Read())
        {
            ListViewItem lvl = new ListViewItem(oleRed[0].ToString());
            lvl.SubItems.Add(oleRed[1].ToString());
            lvl.SubItems.Add(oleRed[2].ToString());
            lvl.SubItems.Add(oleRed[3].ToString());
            lvl.SubItems.Add(oleRed[4].ToString());
            lvl.SubItems.Add(oleRed[5].ToString());
            lv.Items.Add(lvl);
        }
        oleRed.Close();
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.Message.ToString());
    }
}

```

(5) 定义方法 `tbMusicinfoFillReder(string obj)`，功能是根据歌曲信息编号查询对应的歌曲信息。方法 `tbMusicinfoFillReder(string obj)` 的实现代码如下所示：

```

public OleDbDataReader tbMusicinfoFillReder(string obj)
{
    try
    {
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
    }
}

```



```

oledCon.Open();
string strAdd = "select * from tb_musicinfo where Music_code='" + obj + "'";
oledcmd = new OleDbCommand(strAdd, oledCon);
oleRed = oledcmd.ExecuteReader();
return oleRed;
}
catch (Exception ee)
{
    MessageBox.Show(ee.Message.ToString());
    return oleRed;
}
}

```

(6) 定义方法 `tbFill(object obj, string strResult, int intFalg)`, 功能是根据拼音、数字、歌星名、歌曲名等条件来查询对应的歌曲信息, 并将结果显示在 `ListView` 控件中。有如下 3 个参数。

- `obj`: `ListView` 控件的实例。
- `strResult`: 表示查询条件。
- `intFalg`: 表示查询语句。

方法 `tbFill(object obj, string strResult, int intFalg)` 的实现代码如下所示:

```

public int tbFill(object obj, string strResult, int intFalg)
{
    int intResult = 0;
    try
    {
        string strSelect = null;
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();
        switch (intFalg)
        {
            case 1: //数字
                strSelect = "select * from tb_musicinfo where Music_code like '%" + strResult + "%'";
                break;
            case 2: //拼音
                strSelect = "select * from tb_musicinfo where Music_Ping like '%" + strResult + "%'";
                break;
            case 3: //明星
                strSelect = "select * from tb_musicinfo where Music_author like '%" + strResult + "%'";
                break;
            case 4: //歌名
                strSelect = "select * from tb_musicinfo where MusicC_name like '%" + strResult + "%'";
                break;
        }
        oledcmd = new OleDbCommand(strSelect, oledCon);
        oleRed = oledcmd.ExecuteReader();

        ListView lv = (ListView)obj;
        lv.Items.Clear();
        while (oleRed.Read())
        {
            ListViewItem lvl = new ListViewItem(oleRed[0].ToString());
            lvl.SubItems.Add(oleRed[1].ToString());
        }
    }
}

```

```

        lv1.SubItems.Add(oleRed[2].ToString());
        lv1.SubItems.Add(oleRed[3].ToString());
        lv.Items.Add(lv1);
        intResult++;
    }
    oleRed.Close();
    return intResult;
}
catch (Exception ee)
{
    MessageBox.Show(ee.Message.ToString());
    return intResult;
}
}

```

(7) 定义方法 `tbFillName(string strResult)`，功能是根据歌曲信息编号查询对应的歌曲的路径信息。方法 `tbFillName(string strResult)` 的实现代码如下所示：

```

public string tbFillName(string strResult)
{
    string Result = null;
    try
    {
        string strSelect = null;
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();
        strSelect = "select Music_filepath from tb_musicinfo where Music_code= '"
            + strResult + "'";
        oledcmd = new OleDbCommand(strSelect, oledCon);
        oleRed = oledcmd.ExecuteReader();
        oleRed.Read();
        if(oleRed.HasRows)
        {
            Result = oleRed[0].ToString();
        }
        oleRed.Close();
        return Result;
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.Message.ToString());
        return Result;
    }
}

```

## 6.6 设计窗体

视频讲解  光盘：视频\第 6 章\设计窗体.avi

在设计本项目的窗体时，我们需要根据预先规划的功能模块进行布局。在本项目中，共涉及 8 个窗体，接下来，将逐一展示这 8 个窗体的界面效果。

### 1. 登录界面

登录界面实现用户登录验证，我的设计方案如图 6-11 所示。



图 6-11 登录界面窗体

## 2. 后台主界面

管理员登录后，首先显示默认的后台主界面，设计方案如图 6-12 所示。

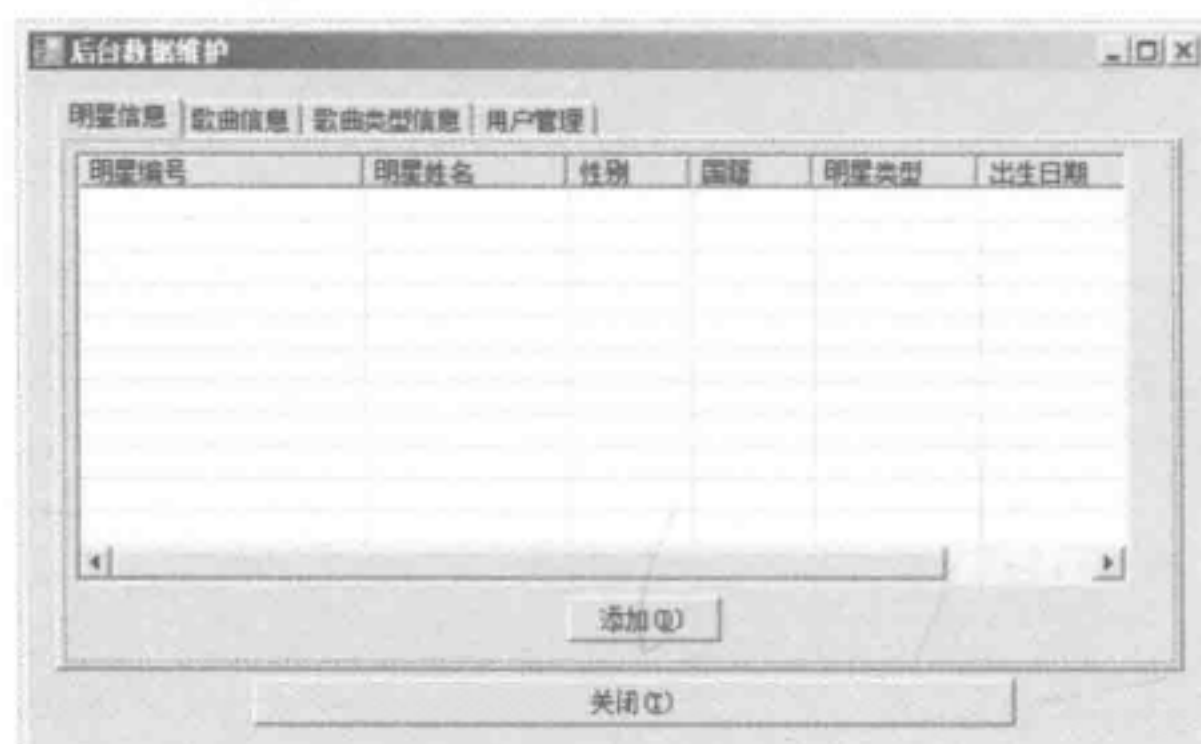


图 6-12 后台主界面窗体

## 3. 点歌系统主界面

用户登录后，首先显示默认的点歌系统主界面，设计方案如图 6-13 所示。

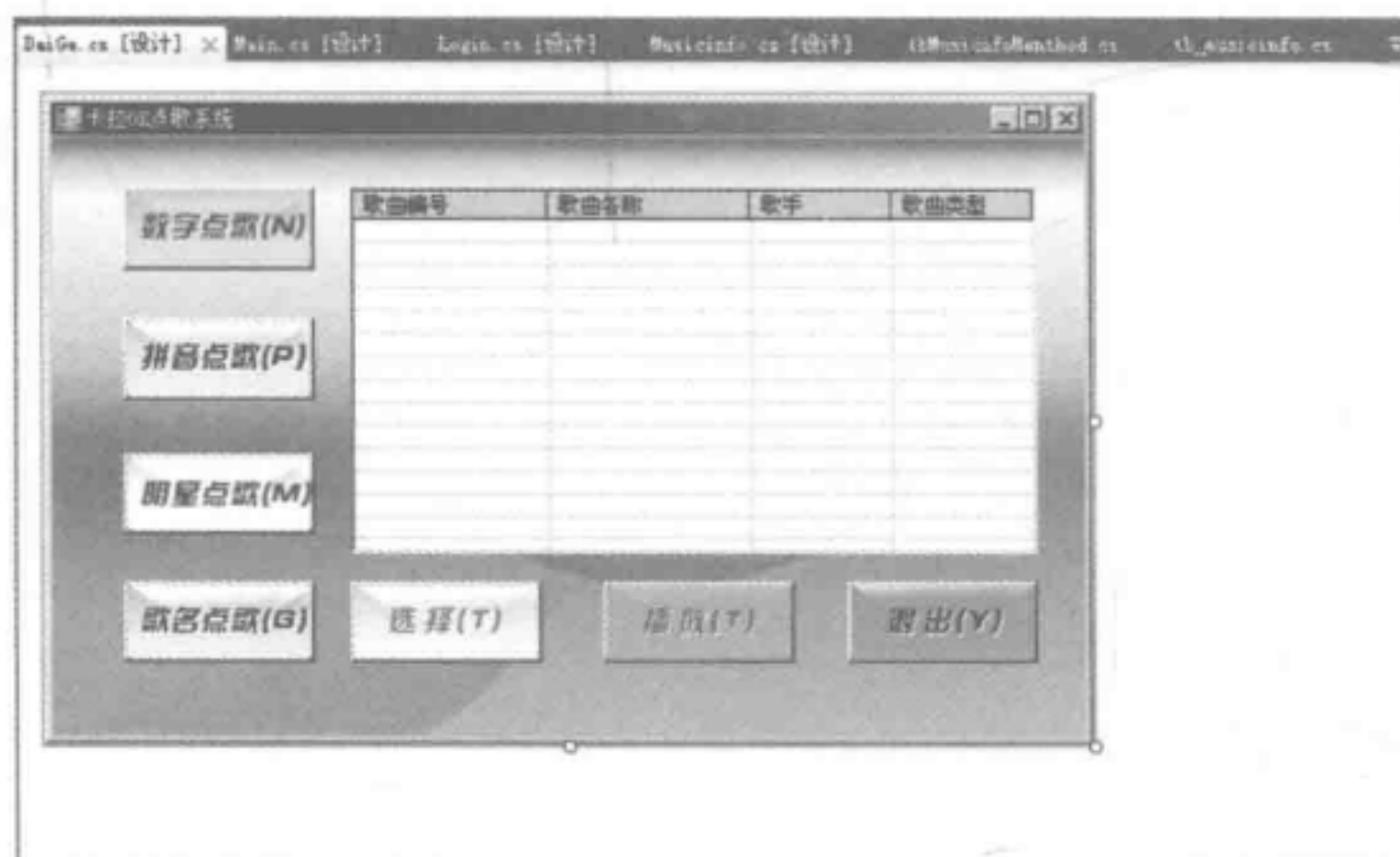


图 6-13 点歌系统主界面窗体



## 6. 歌曲类型界面

歌曲类型界面用于显示系统的歌曲类型，设计方案如图 6-14 所示。



图 6-14 歌曲类型界面窗体

## 5. 添加明星界面

添加明星界面用于添加明星信息，设计方案如图 6-15 所示。



图 6-15 添加明星界面窗体

## 6. 歌曲信息界面

歌曲信息界面用于添加、显示歌曲的信息，设计方案如图 6-16 所示。

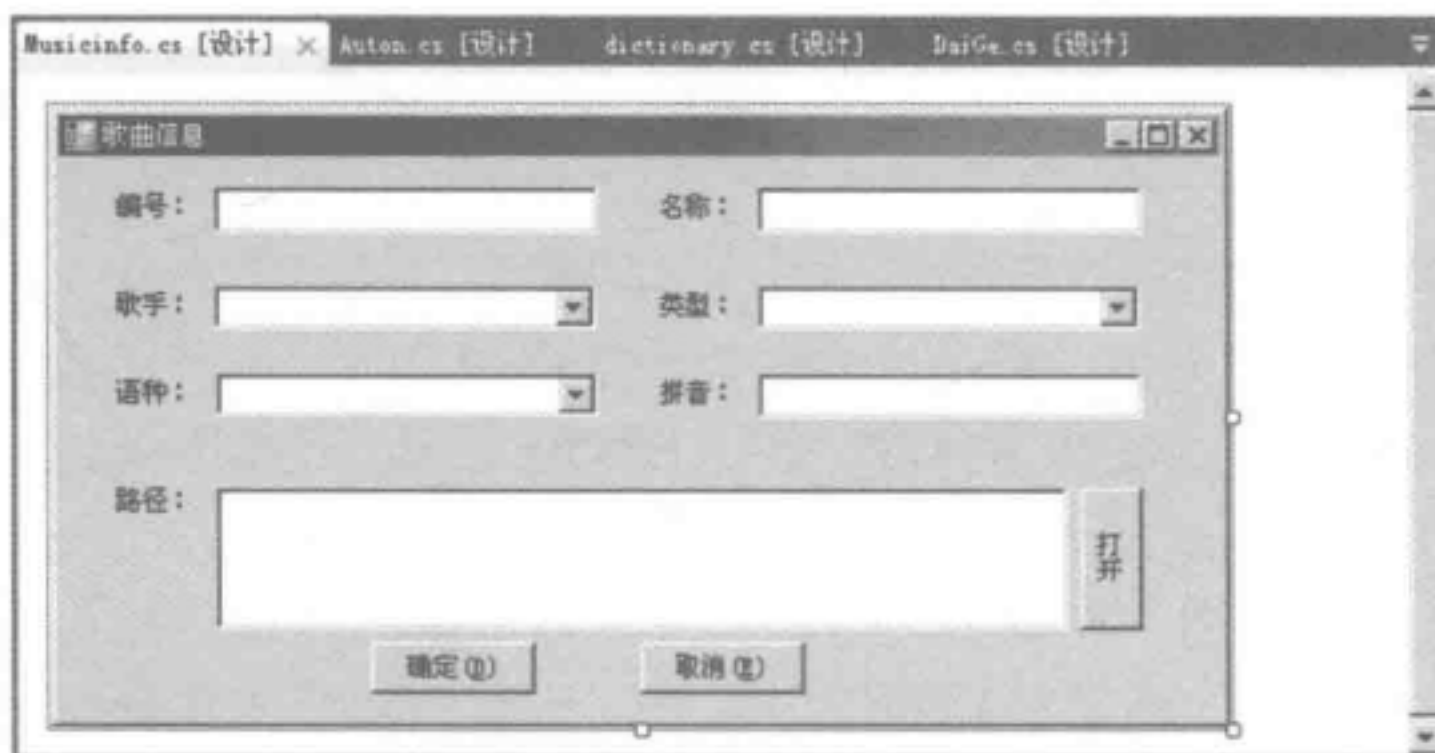


图 6-16 歌曲信息界面窗体

## 7. 查询信息界面

查询信息界面用于查询系统内的歌曲信息，设计方案如图 6-17 所示。



图 6-17 查询信息界面窗体

## 8. 播放界面

播放界面用于播放歌曲信息，设计方案如图 6-18 所示。

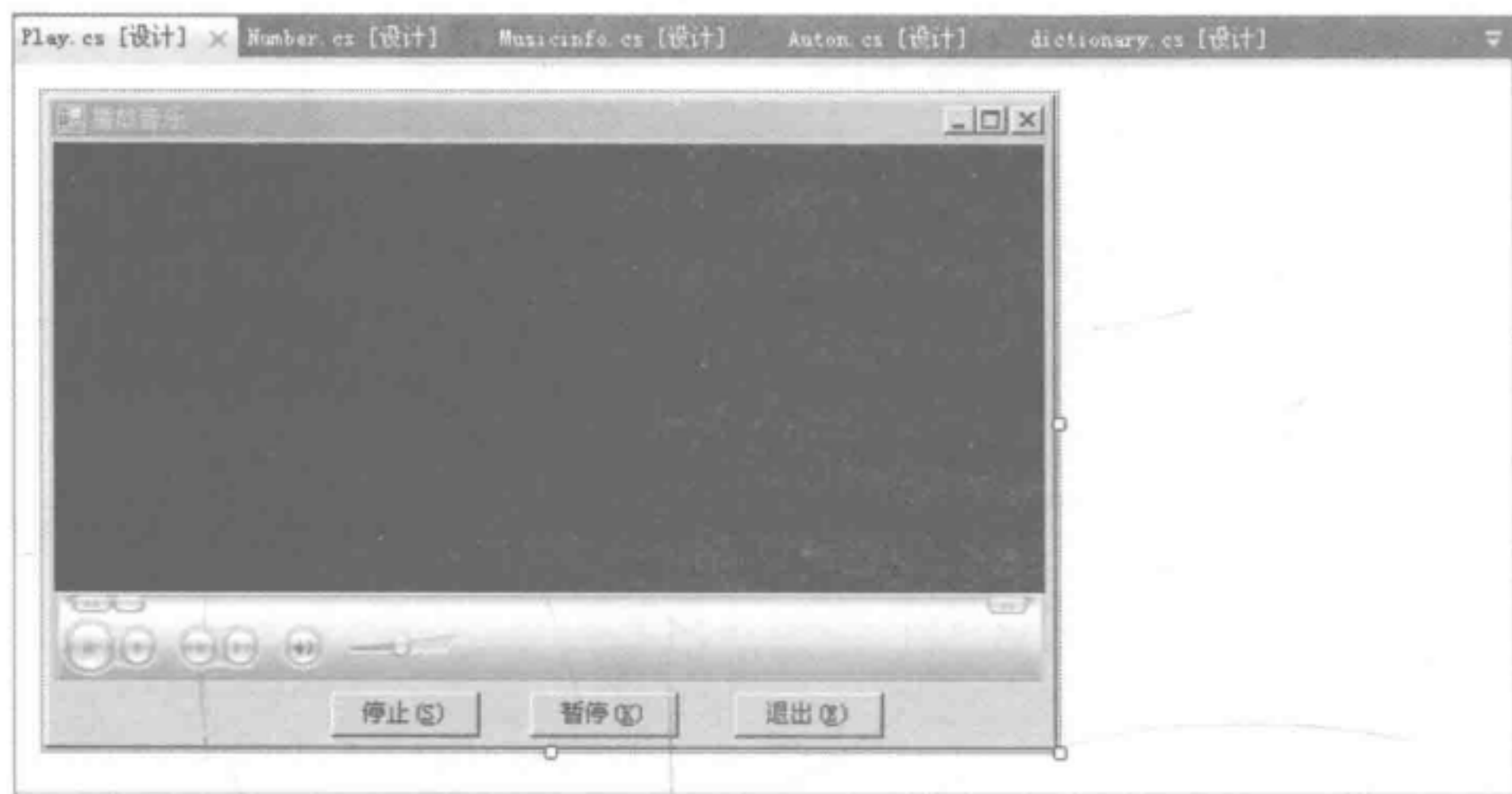


图 6-18 播放界面窗体

到此为止，我们的窗体设计工作就介绍完毕了。其实，作为一个窗体项目，窗体的设计在项目刚刚规划完成之时就已经有了一个大体轮廓了。具体怎么设计，需要从多个因素进行考虑。就拿本 C# 项目来说，因为经常需要在不同的窗体之间传递数据，所以很多人认为在使用 Visual Studio 2013 设计窗体时，在具体规划之前，一定要考虑窗体之间的数据传递问题。其实，对于这一说法，是存在着很大争议的，我们认为，在刚开始设计窗体的时候先无须考虑这方面的问题，在规划之初，首先重点考虑的是功能和实用性，也就是平常说的用户体验。只要确保实现了最佳的用户体验，后面的数据传递问题可以有多重解决方案。并且对于这个问题，微软已经考虑得很周到了，使用专业开发工具 Visual Studio 2013 可以很轻松地解决。

## 6.7 具体编码工作

视频讲解 光盘：视频\第6章\具体编码工作.avi

经过前面内容的介绍，整个项目的基础工作全部结束了。在接下来的内容中，将详细讲解本项目的具体编码过程，详细剖析各个模块的具体实现。希望读者认真阅读，掌握这个项目的技术精髓。

### 6.7.1 登录验证模块

(1) 对于 C# 项目来说，登录验证的原理很简单，具体说明如下。

① 用一个表单供用户输入登录数据。

② 获取用户的登录数据后，与数据库内的合法用户数据进行比较。如果完全一致，则登录系统，如果不一致，则不能登录系统。

(2) 接下来开始介绍编写登录验证模块处理事件代码的具体过程。

① 在登录界面中，当用户单击“登录”按钮后，会执行对应的处理事件 `btnOK_Click(object sender, EventArgs e)`。实现文件是 `Login.cs`，对应代码如下所示：

```
private void btnOK_Click(object sender, EventArgs e)
{
    tb_computer computer = new tb_computer();
    if(txtUser.Text == "")
    {
        MessageBox.Show("登录名称不能为空!");
        txtUser.Focus();
        return;
    }
    if(txtPwd.Text == "")
    {
        MessageBox.Show("登录密码不能为空!");
        txtPwd.Focus();
        return;
    }
    if(cmbLogin.Text == "")
    {
        MessageBox.Show("请选择登录界面");
        cmbLogin.Focus();
        return;
    }
    computer.strcmp_name = txtUser.Text;
    computer.strcmp_Paww = txtPwd.Text;
    if (computer.tb_computerLogin(computer, 2) == 1)
    {
        if (cmbLogin.Text == "后台数据维护")
        {
            Main frm = new Main();
            frm.Show();
            this.Hide();
        }
        if (cmbLogin.Text == "系统点歌")
        {
```



```

        DaiGe daige = new DaiGe();
        daige.Show();
        this.Hide();
    }
}
else
{
    MessageBox.Show("登录失败!");
    txtPwd.Text = "";
    txtUser.Text = "";
    cmbLogin.Text = "";
    txtUser.Focus();
}
}

```

② 在登录界面中，当用户单击“取消”按钮后，会执行对应的处理事件 `btnEsce_Click(object sender, EventArgs e)`。实现文件是 `Login.cs`，对应代码如下所示：

```

private void btnEsce_Click(object sender, EventArgs e)
{
    this.Close();
}

```

上述登录实现是一个典型的登录验证模块，在软件项目中，这个模块十分具有代表性。对于本项目来说，因为使用的是 `Access` 数据库，所以一旦遭到黑客攻击，登录信息就很容易被窃取。此时读者可能会有一个疑问：既然登录信息这么重要，本项目可以使用 `MD5` 加密技术吗？当然可以！我们可以考虑采用 `MD5` 加密技术对系统进行升级。

对于初学者来说，不要对 `MD5` 技术敬而远之，其实非常简单。在用户设置登录密码时，我们可以使用 `MD5` 进行加密，然后在数据库中存储的便是 `MD5` 加密后的数据。此时即使黑客获取了数据库，打开后看见的也是加密信息，而不能得到正确的密码。在网络中有很多针对 `C#` 语言的 `MD5` 加密函数，例如下面的就比较具有代表性：

```

/// <summary>
/// MD5 函数
/// </summary>
/// <param name="str">原始字符串</param>
/// <returns>MD5 结果</returns>
public static string MD5(string str)
{
    byte[] b = Encoding.Default.GetBytes(str);
    b = new MD5CryptoServiceProvider().ComputeHash(b);
    string ret = "";
    for (int i=0; i<b.Length; i++)
        ret += b[i].ToString("x").PadLeft(2, '0');
    return ret;
}

```

可以直接将上述 `MD5` 加密函数用在我们的项目中。

## 6.7.2 后台维护模块

因为当前新歌层出不穷，网络歌曲更是一个比一个火。所以设置一个管理模块势在必行，这样能及时添加新的歌曲。本模块主要实现对如下信息的管理：

- 歌曲信息。

- 歌曲类型。
- 用户管理。

本模块的实现文件是 Main.cs, 下面开始讲解它的实现流程。

(1) 引入命名空间, 这样可以使用 Access 数据库并传递参数。对应的代码如下所示:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Configuration;
using System.Data.OleDb;           //数据库需要的命名空间
using KTV.KTVclass;               //实体类需要的命名空间
```

(2) 在窗体加载事件中调用方法类中的方法来填充 ListView 控件。代码如下所示:

```
public Main()
{
    InitializeComponent();
}
frmdictionaryMethod frmDictyin = new frmdictionaryMethod();
tb_authorinfoMethod tbAuto = new tb_authorinfoMethod();
tbMusicinfoMethod tbMuseic = new tbMusicinfoMethod();
tb_computer computer = new tb_computer();
private void frmMain_Load(object sender, EventArgs e)
{
    frmDictyin.dictionaryFind("2", lvDitcy); //绑定控件数据
    tbAuto.tb_authorinfoFill("2", lvAuton); //绑定控件数据
    tbMuseic.tbMusicinfoFill(lvMuscie);
    computer.tbMusicinfoFill(LvUser);
}
```

上述代码执行后, 会在 ListView 中加载数据, 并且定义了点击不同按钮的事件处理程序。这种做法在窗体项目中还还好一些, 如果是在一个 Web 项目中, 往往同时存在很多个其他不同的控件, 这时候, 加载事件的顺序就会有所不同, 这个时候的加载顺序是因项目而异的。在 C# 项目中, 在 Page 执行中, 将按照如下顺序激活事件:

```
Page.PreInit → Page.Init → Page.InitComplite → Page.PreLoad → Page.Load
→ Page.LoadComplete → Page.PreRender
→ Page.PreRenderComplete
```

如果 Web 页面继承自另外一个页面, 例如继承关系是 BasePage: System.Web.UI.Page, 则 BasePage 和最终 Page 的事件激活顺序是:

```
UI.PreInit → Page.PreInit → UI.Init → Page.Init
→ UI.InitComplite → Page.InitComplite
→ UI.PreLoad → Page.PreLoad
→ UI.Load → Page.Load
→ UI.LoadComplete → Page.LoadComplete
→ UI.PreRender → Page.PreRender
→ UI.PreRenderComplete → Page.PreRenderComplete
```

如果使用了 MasterPage, 则 MasterPage 中的事件和 ContentPage 中的事件将按照下面的顺序触发:



```

ContentPage.PreInit → Master.Init → ContentPage.Init
→ ContentPage.InitComplite → ContentPage.PreLoad
→ ContentPage.Load → Master.Load
→ ContentPage.LoadComplete → ContentPage.PreRender
→ Master.PreRender → ContentPage.PreRenderComplete
    
```

如果 ContentPage 继承 BasePage 的页面事件，则触发顺序如下所示：

```

UI.PreInit → ContentPage.PreInit → Master.Init → UI.Init
→ ContentPage.Init → UI.InitComplite
→ ContentPage.InitComplite → UI.PreLoad
→ ContentPage.PreLoad → UI.Load → ContentPage.Load
→ Master.Load → UI.LoadComplete → ContentPage.LoadComplete
→ UI.PreRender → ContentPage.PreRender
→ Master.PreRender → UI.PreRenderComplete
→ ContentPage.PreRenderComplete
    
```

(3) 当单击 btnAut 按钮时，会弹出一个添加明星信息的窗体，对应代码如下所示：

```

private void btnAut_Click(object sender, EventArgs e)
{
    //添加明星
    Auton frmAuAdd = new Auton(1, "");
    frmAuAdd.Owner = this;
    frmAuAdd.ShowDialog();
}
    
```

(4) 单击 btnMuserAdd 按钮时，会弹出一个添加歌曲信息的窗体，代码如下所示：

```

//歌曲添加
private void btnMuserAdd_Click(object sender, EventArgs e)
{
    Musicinfo frmInfo = new Musicinfo(1, "");
    frmInfo.Owner = this;
    frmInfo.ShowDialog();
}
    
```

(5) 当单击 lvMuscie 按钮时会记录选择歌曲信息的编号，通过这个编号，可以修改歌曲的信息。对应的代码如下所示：

```

public string strMuseName = null;
//歌曲
private void lvMuscie_MouseClick(object sender, MouseEventArgs e)
{
    strMuseName = lvMuscie.SelectedItems[0].SubItems[0].Text;
}
    
```

(6) 当单击 lvDitcy 按钮时，会记录选择歌曲类型信息的编号，通过这个编号，可以修改歌曲的类型信息。对应的代码如下所示：

```

public string strName = null;
//歌曲类别
private void lvDitcy_Click(object sender, EventArgs e)
{
    strName = lvDitcy.SelectedItems[0].SubItems[0].Text; //当前选择的值
}
    
```

(7) 当单击 button1 按钮时，会删除系统内某歌曲类型的信息，对应的代码如下所示：

```

private void button1_Click_1(object sender, EventArgs e)
{
    
```



```

if (strName == null)
{
    MessageBox.Show("请选择要删除的内容!");
    return;
}
else
{
    //删除歌曲信息
    frmDictyin.dictionaryDelete(strName);
    MessageBox.Show("删除成功!");
    frmDictyin.dictionaryFind("2", lvDitcy);
}
}

```

(8) 定义方法 fillScoure(), 功能是将系统内的用户信息从数据库中检索出来, 并显示在 TextBox 控件中。对应的代码如下所示:

```

public void fillScoure()
{
    OleDbDataReader dr = computer.tbFill(strUser);
    dr.Read();
    if (dr.HasRows)
    {
        txtUser.Text = dr[1].ToString();
        txtUser.Enabled = false;
    }
}

```

(9) 当单击 LvUser 按钮时, 会记录选择的用户, 并调用 fillScoure() 方法来查询用户信息。对应的代码如下所示:

```

public string strUser = null;
private void LvUser_Click(object sender, EventArgs e)
{
    strUser = LvUser.SelectedItems[0].SubItems[0].Text;
    if (strUser != null)
    { fillScoure(); }
}

```

### 6.7.3 明星管理模块

在本模块中, 管理员能够对系统内的明星信息进行管理。本模块的实现文件是 Auton.cs, 下面开始讲解其实现流程。

(1) 重写窗体的初始方法, 设置全局变量接收控件窗体的操作参数。对应的代码如下所示:

```

public partial class Auton : Form
{
    public Auton()
    {
        InitializeComponent();
    }
    public Auton(int intcunt, string strId)
    {
        InitializeComponent();
        intFalg = intcunt;
    }
}

```

```

        strgetId = strId;
    }
    public int intFalg = 0;
    public string strgetId = null;

```

(2) 在加载窗体时, 根据操作标识执行对应的操作, 并给窗体数据控件赋值。对应的代码如下所示:

```

frmdictionaryMethod frmDictiyon = new frmdictionaryMethod();
tb_authorinfo tbAu = new tb_authorinfo();
tb_authorinfoMethod tbAuMethod = new tb_authorinfoMethod();
private void frmAuton_Load(object sender, EventArgs e)
{
    frmDictiyon.dictionaryFind("1", cmbauthorcompany);

    if (intFalg == 1)
    {
        txtauthorId.Text = tbAuMethod.gettb_authorinfoID();
    }
    if (intFalg == 2)
    {
        getFill();
    }
}

```

(3) 定义方法 getFill(), 功能是根据条件将数据从数据库中查询出来并显示在窗体中。对应的代码如下所示:

```

public void getFill()
{
    OleDbDataReader dr = tbAuMethod.AuthFind(strgetId);
    dr.Read();
    if (dr.HasRows)
    {
        txtauthorId.Text = dr[0].ToString();
        txtauthorName.Text = dr[1].ToString();
        cmbauthorSex.Text = dr[2].ToString();
        daAuthorbirthday.Value = Convert.ToDateTime(dr[3].ToString());
        comboBox2.Text = dr[4].ToString();
        cmbauthorcompany.Text = dr[5].ToString();
        txtauthorRecma.Text = dr[6].ToString();
        txtauthorzjm.Text = dr[7].ToString();
    }
    dr.Close();
}

```

(4) 单击 btnSur 按钮后, 根据操作标记执行对应的操作处理。对应的代码如下所示:

```

private void btnSure_Click(object sender, EventArgs e)
{
    if (txtauthorName.Text == "")
    {
        MessageBox.Show("姓名不能为空");
        return;
    }
    tbAu.intauthorId = txtauthorId.Text;
    tbAu.strauthorName = txtauthorName.Text;
    tbAu.strauthorSex = cmbauthorSex.Text;
    tbAu.daauthorbirthday = daAuthorbirthday.Value;
    tbAu.strauthorGenre = comboBox2.Text;
    tbAu.strauthorcompany = cmbauthorcompany.Text;
}

```

```
tbAu.strauthorRecma = txtauthorRecma.Text;
tbAu.strauthorzjm = txtauthorzjm.Text;
tbAu.daRdateTime = DateTime.Now;
if (intFalg == 1)
{
    if (tbAuMethod.AuthAdd(tbAu) == 1)
    {
        MessageBox.Show("添加成功!");
        Main frm = (Main)this.Owner;
        tbAuMethod.tb_authorinfoFill("2", frm.lvAuton);
        intFalg = 0;
        this.Close();
    }
    else
    {
        MessageBox.Show("添加失败");
        intFalg = 0;
        this.Close();
    }
}
}
```

(5) 定义 GetCodstring(string UnName)方法, 功能是实现汉字拼音的简码处理。对应的代码如下所示:

```
public static string GetCodstring(string UnName)
{
    int i = 0;
    ushort key = 0;
    string strResult = string.Empty;

    //创建两个不同的 encoding 对象
    Encoding unicode = Encoding.Unicode;
    //创建 GBK 码对象
    Encoding gbk = Encoding.GetEncoding(936);
    //将 unicode 字符串转换为字节
    byte[] unicodeBytes = unicode.GetBytes(UnName);
    //再转化为 GBK 码
    byte[] gbkBytes = Encoding.Convert(unicode, gbk, unicodeBytes);
    while (i < gbkBytes.Length)
    {
        //如果为数字\字母\其他 ASCII 符号
        if (gbkBytes[i] <= 127)
        {
            strResult = strResult + (char)gbkBytes[i];
            i++;
        }
        #region 否则生成汉字拼音简码, 取拼音首字母
        else
        {
            key = (ushort)(gbkBytes[i] * 256 + gbkBytes[i + 1]);
            if (key >= '\uB0A1' && key <= '\uB0C4')
            {
                strResult = strResult + "A";
            }
            else if (key >= '\uB0C5' && key <= '\uB2C0')
            {
                strResult = strResult + "B";
            }
        }
    }
}
```



```
else if (key >= '\uB2C1' && key <= '\uB4ED')
{
    strResult = strResult + "C";
}
else if (key >= '\uB4EE' && key <= '\uB6E9')
{
    strResult = strResult + "D";
}
else if (key >= '\uB6EA' && key <= '\uB7A1')
{
    strResult = strResult + "E";
}
else if (key >= '\uB7A2' && key <= '\uB8C0')
{
    strResult = strResult + "F";
}
else if (key >= '\uB8C1' && key <= '\uB9FD')
{
    strResult = strResult + "G";
}
else if (key >= '\uB9FE' && key <= '\uBBF6')
{
    strResult = strResult + "H";
}
else if (key >= '\uBBF7' && key <= '\uBFA5')
{
    strResult = strResult + "J";
}
else if (key >= '\uBFA6' && key <= '\uC0AB')
{
    strResult = strResult + "K";
}
else if (key >= '\uC0AC' && key <= '\uC2E7')
{
    strResult = strResult + "L";
}
else if (key >= '\uC2E8' && key <= '\uC4C2')
{
    strResult = strResult + "M";
}
else if (key >= '\uC4C3' && key <= '\uC5B5')
{
    strResult = strResult + "N";
}
else if (key >= '\uC5B6' && key <= '\uC5BD')
{
    strResult = strResult + "O";
}
else if (key >= '\uC5BE' && key <= '\uC6D9')
{
    strResult = strResult + "P";
}
else if (key >= '\uC6DA' && key <= '\uC8BA')
{
    strResult = strResult + "Q";
}
else if (key >= '\uC8BB' && key <= '\uC8F5')
{
    strResult = strResult + "R";
}
else if (key >= '\uC8F6' && key <= '\uCBF9')
```

```

    {
        strResult = strResult + "S";
    }
    else if (key >= '\uCBFA' && key <= '\uCDD9')
    {
        strResult = strResult + "T";
    }
    else if (key >= '\uCDDA' && key <= '\uCEF3')
    {
        strResult = strResult + "W";
    }
    else if (key >= '\uCEF4' && key <= '\uD188')
    {
        strResult = strResult + "X";
    }
    else if (key >= '\uD1B9' && key <= '\uD4D0')
    {
        strResult = strResult + "Y";
    }
    else if (key >= '\uD4D1' && key <= '\uD7F9')
    {
        strResult = strResult + "Z";
    }
    else
    {
        strResult = strResult + "?";
    }
    i = i + 2;
}
#endregion
} //end while
return strResult;
}

```

(6) 在 txtauthorName 文本框的 TextChanged 事件中调用 GetCodstring 方法, 用于获取明星的姓名的拼音简码; 当按下 btnEsce 按钮时, 取消整个操作。对应的代码如下所示:

```

private void txtauthorName_TextChanged(object sender, EventArgs e)
{
    if (txtauthorName.Text != "")
    {
        txtauthorzjm.Text = GetCodstring(txtauthorName.Text);
    }
}

private void btnEsce_Click(object sender, EventArgs e)
{
    this.Close();
}

```

(7) 当单击 lvAuton 按钮时, 记录已经选择明星信息的编号。对应的代码如下所示:

```

public string strNameAuton = null;
//明星
private void lvAuton_Click(object sender, EventArgs e)
{
    strNameAuton = lvAuton.SelectedItems[0].SubItems[0].Text; //当前选择的值
}

```

到此为止, 整个明星管理模块介绍完毕, 整个模块的核心内容是各个处理函数。

C#项目中的模块功能是通过一个个函数来实现的，并且这些函数可以用不同的方式来表现，例如，有时通过事件处理程序的方式实现，有时通过构造方法的方式实现，这两种方式有什么区别吗？

在此，以本章项目为素材进行比较，在后台维护模块中，将主要功能代码都定义在了事件处理程序中了。而在设计的明星模块中，将主要功能都定义在了构造方法中了。虽然这两种方式都能实现我们需要的功能，但是两者是有区别的。要想探寻这两种方式的具体差异，就得从构造函数的特点说起了。

构造函数是一种特殊的方法，主要用来在创建对象时初始化对象，即为对象成员变量赋初始值，总与 new 运算符一起使用在创建对象的语句中，特别是，一个类中有多个构造函数时，可以根据其参数个数的不同或参数类型的不同来区分它们，即构造函数的重载。

首先，窗体的 Load 事件是在窗体加载时执行的，构造函数里的代码是这个类的一个实例被创建时，也就是实例化一个类的对象时调用的。真要分出个它们的执行顺序的话，建议读者亲自设置断点进行跟踪，这样可以加深理解。

#### 6.7.4 系统点歌模块

作为 KTV 点歌系统，点歌功能当然是整个项目的核心模块。去过 KTV 的人肯定不会陌生，我们只需选择一手歌曲，系统就会从曲库里调出此首歌曲并播放。

本项目点歌模块的功能是，当用户登录后，可以点选自己需要的歌曲。本模块的实现文件是 DaiGe.cs，是由一些按钮来实现具体功能的，各个按钮的具体说明如下。

- btnNumber: 实现数字点歌。
- btnPing: 实现拼音点歌。
- btnAutor: 实现明星点歌。
- btnName: 实现歌名点歌。
- btnEsce: 实现退出系统。
- btnSelect: 选择某首歌曲后的处理事件。

文件 DaiGe.cs 的具体实现代码如下所示：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using KTV.KTVclass;
namespace KTV
{
    public partial class DaiGe : Form
    {
        public DaiGe()
        {
            InitializeComponent();
        }
        private void tabPage1_Click(object sender, EventArgs e)
        {
        }
    }
}
```



```
//数字点歌
private void btnNumber_Click(object sender, EventArgs e)
{
    Number frm1 = new Number(1) ;
    frm1.Owner = this;
    frm1.ShowDialog();
}
//拼音点歌
private void btnPing_Click(object sender, EventArgs e)
{
    Number frm2 = new Number(2) ;
    frm2.Owner = this;
    frm2.ShowDialog();
}
//明星点歌
private void btnAutor_Click(object sender, EventArgs e)
{
    Number frm3 = new Number(3) ;
    frm3.Owner = this;
    frm3.ShowDialog();
}
//歌名点歌
private void btnName_Click(object sender, EventArgs e)
{
    Number frm4 = new Number(4) ;
    frm6.Owner = this;
    frm6.ShowDialog();
}
private void frmDaiGe_Load(object sender, EventArgs e)
{
}
private void btnEsce_Click(object sender, EventArgs e)
{
    DialogResult diaol = MessageBox.Show("是否要退出系统!", "提示",
        MessageBoxButtons.YesNo, MessageBoxIcon.Information);
    if (diaol == DialogResult.Yes)
    {
        Application.Exit();
    }
}
tbMusicinfoMethod tbMend = new tbMusicinfoMethod();
private void btnSelect_Click(object sender, EventArgs e)
{
    if (stringName != null)
    {
        stringName = tbMend.tbFillName(stringName);
        MessageBox.Show("选择歌曲<<" + strigName2
            + ">>完成, 单击《播放》按钮, 播放歌曲!", "提示");
    }
    else
    {
        MessageBox.Show("请选择要播放的歌曲!", "提示");
    }
}
private void btnPlay_Click(object sender, EventArgs e)
{
    if (stringName != null)
    {
        Play frm = new Play(stringName);
        frm.Owner = this;
        frm.ShowDialog();
    }
}
```

```

        stringName = null;
        // lvPlay.SelectedItems[0].Selected = false;
    }
    else
    {
        MessageBox.Show("请选择要播放的歌曲!", "提示");
    }
}
string stringName = null;
string strigName2 = null;
private void lvPlay_Click(object sender, EventArgs e)
{
    stringName = lvPlay.SelectedItems[0].SubItems[0].Text;
    strigName2 = lvPlay.SelectedItems[0].SubItems[1].Text;
}
}
}

```

### 6.7.5 歌曲信息模块

系统中肯定会有海量的歌曲信息，国内的、国外的都有，不同风格、不同组合，总之信息量巨大。用户进入 KTV 之后会选歌，但是，所选择的歌曲究竟是不是自己需要的呢？这就需要我们添加一个歌曲信息功能，能让用户及时了解所选歌曲的信息。

本项目中，歌曲信息模块的功能是，实现对系统内歌曲信息的显示处理。本模块的实现文件是 Musicinfo.cs，是由一些按钮和文本框实现具体功能的，下面介绍其实现流程。

(1) 载入初始信息，根据操作分别判断执行修改还是删除操作。对应的代码如下所示：

```

namespace KTV
{
    public partial class Musicinfo : Form
    {
        public Musicinfo()
        {
            InitializeComponent();
        }
        public Musicinfo(int intid, string strName)
        {
            InitializeComponent();
            intFalg = intid;
            strId = strName;
        }
        public int intFalg = 0;
        public string strId = null;
        tb_musicinfo tbMusice = new tb_musicinfo();
        tbMusicinfoMethod tbMuseNeth = new tbMusicinfoMethod();
        tb_authorinfoMethod AuMethod = new tb_authorinfoMethod();
        frmdictionaryMethod diction = new frmdictionaryMethod();
        private void frmMusicinfo_Load(object sender, EventArgs e)
        {
            AuMethod.tb_authorinfoFill("1", cmbMusic_author); //填充明星 ComBox 控件
            diction.dictionaryFind("1", cmbMusic_Kind); //填充类型 ComBox 控件
            if (intFalg == 1) //添加
            {
                txtMusic_code.Text = tbMuseNeth.tbMusicinfoID().ToString();
            }
        }
    }
}

```

(2) 定义函数 btnAdd\_Click(object sender, EventArgs e), 功能是实现歌曲信息的添加处理。对应的代码如下所示:

```
//添加
private void btnAdd_Click(object sender, EventArgs e)
{
    if (txtMusicC_name.Text == "")
    {
        MessageBox.Show("请输入歌曲名称");
        txtMusicC_name.Focus();
        return;
    }
    if (cmbMusic_author.Text == "")
    {
        MessageBox.Show("请输入歌手名称");
        return;
    }
    if (txtMusic_filepath.Text == "")
    {
        MessageBox.Show("请输入歌曲文件路径");
        txtMusic_filepath.Focus();
        return;
    }
    tbMusice.strMusic_code = txtMusic_code.Text;
    tbMusice.strMusicC_name = txtMusicC_name.Text;
    tbMusice.strMusic_author = cmbMusic_author.Text;
    tbMusice.strMusic_Kind = cmbMusic_Kind.Text;
    tbMusice.strMusic_chinse = cmbMusic_chinse.Text;
    tbMusice.strMusic_filepath = txtMusic_filepath.Text;
    tbMusice.strMusic_Ping = txtMusic_Ping.Text;

    tbMusice.daMusic_date = DateTime.Now;
    tbMusice.intMusic_falg = 0;
    Main frm = (Main)this.Owner;
    if (intFalg == 1)
    {
        if (tbMuseNeth.tbMusicinfoAdd(tbMusice) == 1)
        {
            MessageBox.Show("添加成功!");

            tbMuseNeth.tbMusicinfoFill(frm.lvMuscie);
            intFalg = 0;
            this.Close();
        }
        else
        {
            MessageBox.Show("添加失败");
            tbMuseNeth.tbMusicinfoFill(frm.lvMuscie);
            intFalg = 0;
            this.Close();
        }
    }
    if (intFalg == 2)
    {
        if (tbMuseNeth.tbMusicinfoUpdate(tbMusice) == 1)
        {
            MessageBox.Show("修改成功!");
            tbMuseNeth.tbMusicinfoFill(frm.lvMuscie);
            intFalg = 0;
            this.Close();
        }
    }
}
```



```

    }
}
}

```

(3) 定义函数 GetCodstring(string UnName), 功能是获取汉子的拼音编码。对应的代码如下所示:

```

private void txtMusicC_name_TextChanged(object sender, EventArgs e)
{
    if (txtMusicC_name.Text != "")
    {
        txtMusic_Ping.Text=GetCodstring(txtMusicC_name.Text);
    }
}
public static string GetCodstring(string UnName)
{
    int i = 0;
    ushort key = 0;
    string strResult = string.Empty;

    //创建两个不同的 encoding 对象
    Encoding unicode = Encoding.Unicode;
    //创建 GBK 码对象
    Encoding gbk = Encoding.GetEncoding(936);
    //将 Unicode 字符串转换为字节
    byte[] unicodeBytes = unicode.GetBytes(UnName);
    //再转化为 GBK 码
    byte[] gbkBytes = Encoding.Convert(unicode, gbk, unicodeBytes);
    while (i < gbkBytes.Length)
    {
        //如果为数字\字母\其他 ASCII 符号
        if (gbkBytes[i] <= 127)
        {
            strResult = strResult + (char)gbkBytes[i];
            i++;
        }
        #region 否则生成汉字拼音简码, 取拼音首字母
        else
        {
            key = (ushort)(gbkBytes[i] * 256 + gbkBytes[i + 1]);
            if (key >= '\uB0A1' && key <= '\uB0C4')
            {
                strResult = strResult + "A";
            }
            else if (key >= '\uB0C5' && key <= '\uB2C0')
            {
                strResult = strResult + "B";
            }
            else if (key >= '\uB2C1' && key <= '\uB4ED')
            {
                strResult = strResult + "C";
            }
            else if (key >= '\uB4EE' && key <= '\uB6E9')
            {
                strResult = strResult + "D";
            }
            else if (key >= '\uB6EA' && key <= '\uB7A1')
            {
                strResult = strResult + "E";
            }
        }
    }
}

```

```

else if (key >= '\uB7A2' && key <= '\uB8C0')
{
    strResult = strResult + "F";
}
else if (key >= '\uB8C1' && key <= '\uB9FD')
{
    strResult = strResult + "G";
}
else if (key >= '\uB9FE' && key <= '\uBBF6')
{
    strResult = strResult + "H";
}
else if (key >= '\uBBF7' && key <= '\uBFA5')
{
    strResult = strResult + "J";
}
else if (key >= '\uBFA6' && key <= '\uC0AB')
{
    strResult = strResult + "K";
}
else if (key >= '\uC0AC' && key <= '\uC2E7')
{
    strResult = strResult + "L";
}
else if (key >= '\uC2E8' && key <= '\uC4C2')
{
    strResult = strResult + "M";
}
else if (key >= '\uC4C3' && key <= '\uC5B5')
{
    strResult = strResult + "N";
}
else if (key >= '\uC5B6' && key <= '\uC5BD')
{
    strResult = strResult + "O";
}
else if (key >= '\uC5BE' && key <= '\uC6D9')
{
    strResult = strResult + "P";
}
else if (key >= '\uC6DA' && key <= '\uC8BA')
{
    strResult = strResult + "Q";
}
else if (key >= '\uC8BB' && key <= '\uC8F5')
{
    strResult = strResult + "R";
}
else if (key >= '\uC8F6' && key <= '\uCBF9')
{
    strResult = strResult + "S";
}
else if (key >= '\uCBFA' && key <= '\uCDD9')
{
    strResult = strResult + "T";
}
else if (key >= '\uCDDA' && key <= '\uCEF3')
{
    strResult = strResult + "W";
}
else if (key >= '\uCEF4' && key <= '\uD188')

```

```

        {
            strResult = strResult + "X";
        }
        else if (key >= '\uD1B9' && key <= '\uD4D0')
        {
            strResult = strResult + "Y";
        }
        else if (key >= '\uD4D1' && key <= '\uD7F9')
        {
            strResult = strResult + "Z";
        }
        else
        {
            strResult = strResult + "?";
        }
        i = i + 2;
    }
    #endregion
} //end while
return strResult;
}

//打开文件路径
private void btnOpen_Click(object sender, EventArgs e)
{
    openPath.Filter = "(*.wav)|*.wav|(*.mp3)|*.mp3|(*.avi)|*.avi";
    openPath.FileName = "";
    if (openPath.ShowDialog() == DialogResult.OK)
    {
        txtMusic_filepath.Text = openPath.FileName;
    }
}

```

## 6.7.6 播放歌曲模块

播放歌曲模块的功能是，当用户选择一首歌曲后，播放这首歌曲。本模块的实现文件是 Play.cs，对应的实现代码如下所示：

```

namespace KTV
{
    public partial class Play : Form
    {
        public Play()
        {
            InitializeComponent();
        }

        public Play(string strPaht)
        {
            InitializeComponent();
            strPath = strPaht;
        }

        public string strPath = null;
        private void button1_Click(object sender, EventArgs e)
        {
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
        }
    }
}

```



```

    {
        this.axWindowsMediaPlayer1.Ctlcontrols.stop();
    }

    private void frmPlay_Load(object sender, EventArgs e)
    {
        //播放文件
        this.axWindowsMediaPlayer1.URL = strPath;
    }

    private void btnExce_Click(object sender, EventArgs e)
    {
        this.axWindowsMediaPlayer1.Ctlcontrols.stop();
        this.Close();
    }

    private void btnZan_Click(object sender, EventArgs e)
    {
        if (btnZan.Text == "暂停(&K)")
        {
            this.axWindowsMediaPlayer1.Ctlcontrols.pause();
            btnZan.Text = "继续(&K)";
        }
        else
        {
            this.axWindowsMediaPlayer1.Ctlcontrols.play();
            btnZan.Text = "暂停(&K)";
        }
    }

    private void axWindowsMediaPlayer1_Enter(object sender, EventArgs e)
    {
    }
}

```

到此为止，本项目的设计和编码工作全部结束。

(1) 在 C#窗体程序中，当启动 Windows Form 应用程序时，会以下列顺序引发主要表单的启动事件：

- System.Windows.Forms.Control.HandleCreated。
- System.Windows.Forms.Control.BindingContextChanged。
- System.Windows.Forms.Form.Load。
- System.Windows.Forms.Control.VisibleChanged。
- System.Windows.Forms.Form.Activated。
- System.Windows.Forms.Form.Shown。

(2) 当应用程序关闭时，会以下列顺序引发主要表单的关闭事件：

- System.Windows.Forms.Form.Closing。
- System.Windows.Forms.Form.FormClosing。
- System.Windows.Forms.Form.Closed。
- System.Windows.Forms.Form.FormClosed。
- System.Windows.Forms.Form.Deactivate。

## 6.8 项目调试

在 Visual Studio .NET 中打开项目后，在“解决方案资源管理器中”查看文件目录，发现与最初规划的完全一致，具体如图 6-19 所示。



图 6-19 Visual Studio 2013 中的项目文件结构

其中包含了两个程序文件夹，具体说明如下所示。

(1) Database

该文件夹用于保存系统的 Access 数据库文件。

(2) KTVclass

该文件夹用于保存系统的公共类文件。

系统的用户登录界面效果如图 6-20 所示。

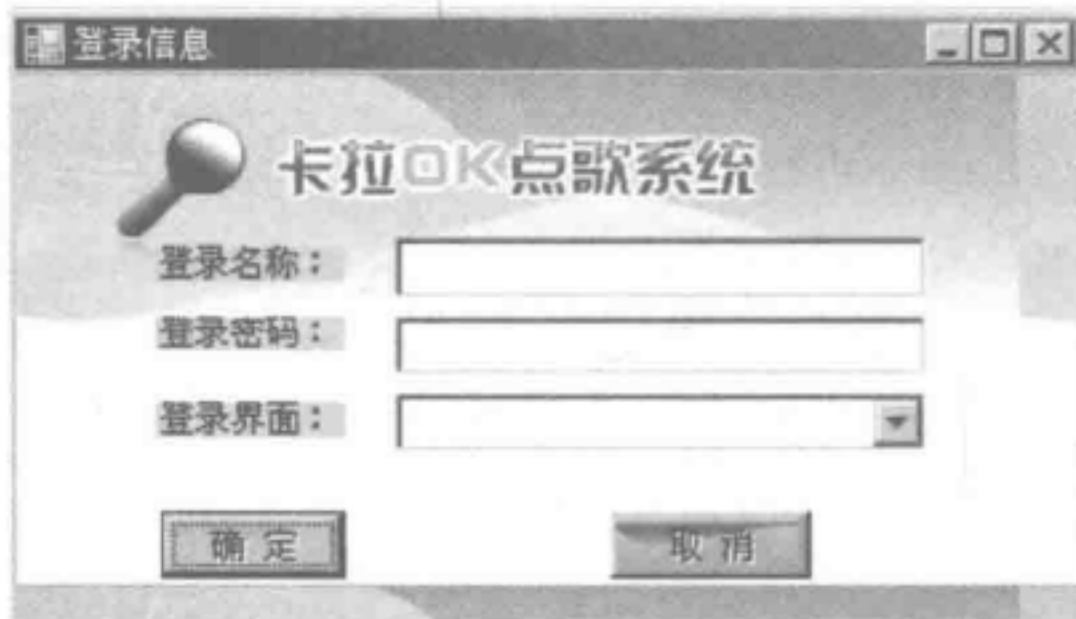


图 6-20 系统登录界面的效果

后台维护界面的效果如图 6-21 所示。



图 6-21 后台维护界面的效果

前台点歌界面的效果如图 6-22 所示。



图 6-22 前台点歌界面的效果

歌曲播放界面的效果如图 6-23 所示。

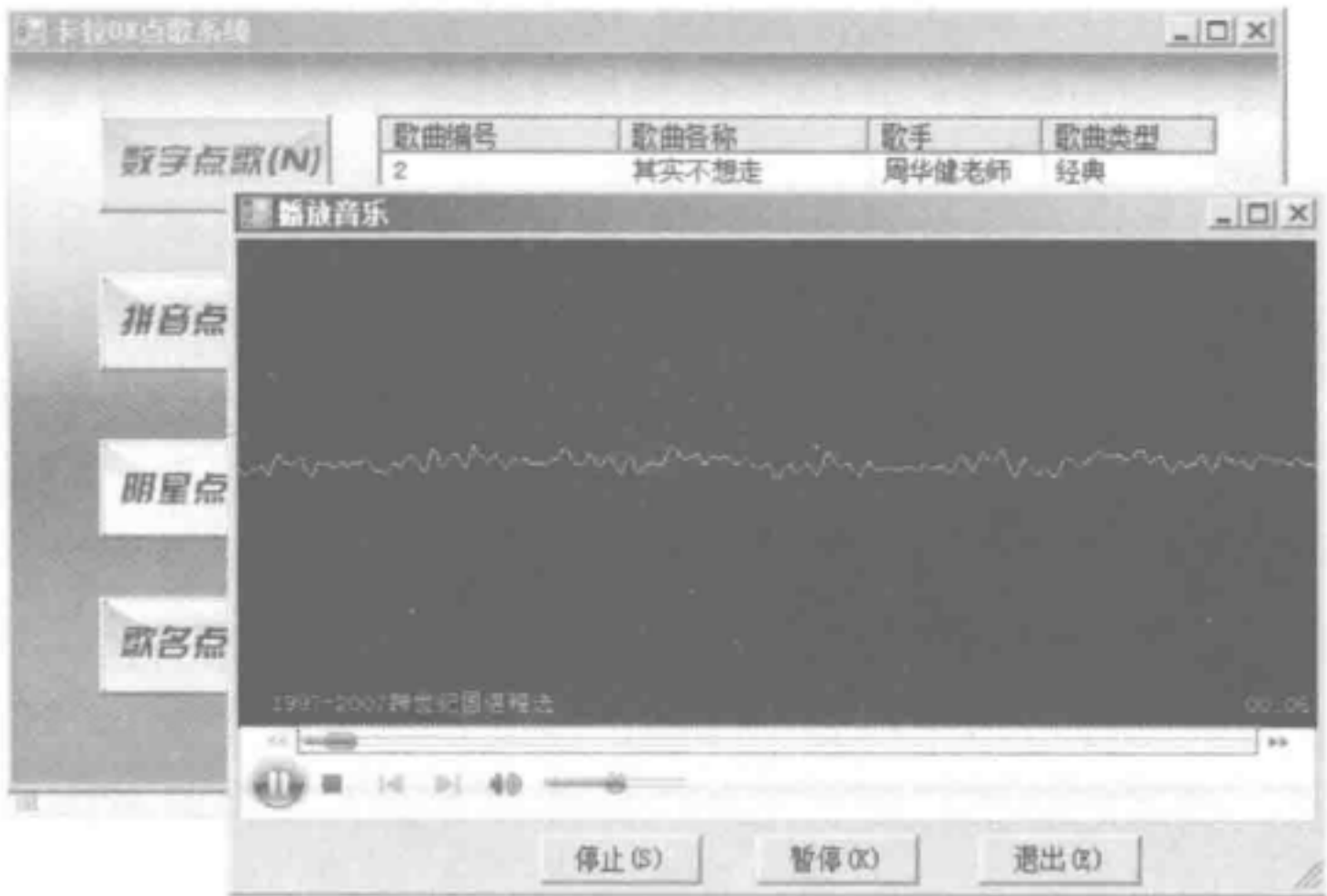


图 6-23 歌曲播放界面的效果



## 6.9 系统升级

视频讲解 光盘：视频\第6章\系统升级.avi

到目前为止，本系统能够正常运行，可以满足客户当前的需求。但是，一个项目在规划设计之初有一个背景，而随着时间的推移，背景也会随之变化，这个时候，就需要对软件项目进行升级，以适应新的环境要求。而作为开发团队来说，在规划项目时就需要考虑到以后的系统升级问题。在本节的内容中，将详细讲解对本点歌系统升级的知识。

### 6.9.1 功能升级——升级前的思考

(1) 在具体升级之前，读者应考虑如下 3 个问题：

- 本系统升级的可行性，具体工作量有多大。
- 当前的模块功能是否科学。
- 当前的配置是否可以再升级，具体怎么升级。

(2) 要想回答上述 3 个问题，得从本项目的系统分析谈起。无论是作为一名富有经验的项目经理，还是作为一名架构师，在项目伊始，都需要考虑到如下两个问题。

① 系统的可扩展性。合格的项目程序需要以面向对象为基础，并遵循模块化设计原则，确保在不改变整体结构的前提下可以灵活地添加新的功能模块。

② 配置升级。随着系统的发展和客户业务的发展变化，配置硬件也需要随之升级。合格的项目程序需要以面向对象为基础，并遵循模块化设计原则，确保只需修改最少的代码，即可实现配置的升级。例如，实现 Access 数据库向 SQL Server 数据库或 Oracle 数据库的升级。

(3) 接下来开始解答上面提出的 3 个问题。

① 本系统升级可行性，具体工作量有多大。

因为本项目是基于面向对象构建的，自始至终遵循了模块化开发原则，所以升级工作比较简单，只须添加新的独立模块即可。

② 当前的模块功能是否科学。

对于 KTV 点歌系统来说，与其他类型的系统是有区别的。因为每天都会涌现出不少新歌和娱乐圈新人，所以无论是歌曲信息还是明星信息，数量只会越来越多，而不是越来越少。所以在项目规划伊始，只设置了歌曲添加功能和明星添加功能，并没有设置对应的修改功能和删除功能。但是，作为一个软件项目，最基本的要求就是无限扩展和使用正常。

- 无限扩展：是指遵循面向对象思想，确保项目易于科学升级，有关这一点，已经在整个项目的实现过程中得到了体现，在此不再讲解。
- 使用正常：就是能够执行所有可能的操作，并且在操作时不会出现任何异常。

很显然，现在的项目无法满足上述“使用正常”的要求。例如，可能有的维护人员想删除某位不喜欢的歌星或歌曲信息，也可能某位管理员离职之后，为了安全起见，需要修改管理密码，等等。这些都需要项目具备修改和删除功能。基于此，我们可以进一步重新规划本项目，重新规划后的项目运行流程如图 6-24 所示。

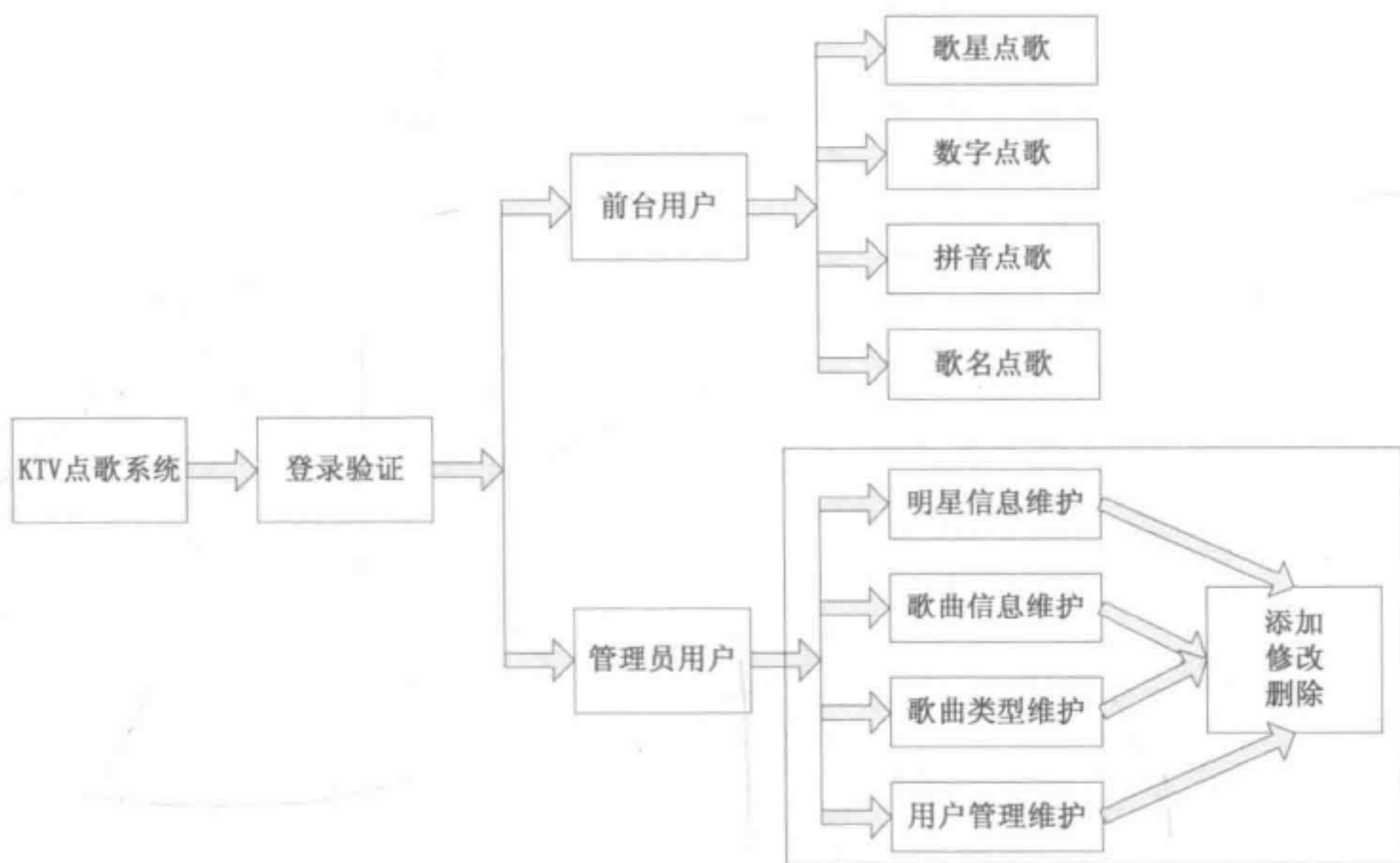


图 6-24 重新规划后的运行流程

与原来的流程图 6-2 相比，变化主要体现在图 6-24 中矩形框中的部分，即为管理员用户新增了信息修改和删除功能。上述新增功能在技术上将十分容易实现，只需遵循面向对象的原则，在原有项目文件的基础上编写新的修改和删除处理函数即可。

### ③ 当前的配置是否可以再升级，具体怎么升级？

配置升级比较简单，只是数据库的选择问题。本项目现在使用的是 Access 数据库，可以考虑向更专业、更高级的 SQL Server 或 Oracle 升级。

## 6.9.2 增加维护歌曲信息模块

我们知道，对数据库数据的操作是通过 SQL 语句实现的，在前面的添加歌曲信息模块中，是通过方法 `tbMusicinfoAdd` 实现的，此方法使用 SQL 中的“insert into”语句，向数据库表 `tb_musicinfo` 中添加歌曲信息。同样道理，歌曲的修改和删除操作也可以使用 SQL 语句来实现。并且遵循模块化设计原则，可以单独编写对应的操作方法。在具体实现时，是不是需要单独保存在一个新文件中呢？当然不是，因为系统遵循了面向对象的原则，将所有与歌曲操作相关的方法放在了类 `tbMusicinfoMethod` 中，所以我们只需将新编写的维护歌曲信息的方法保存在这个类中即可。

在原有的基础上重新定义类 `tbMusicinfoMethod`，用于实现对系统内歌曲信息的各种操作处理。升级工作主要完成对歌曲信息的修改和删除操作，每种操作是通过对应的方法实现的。实现文件是 `tbMusicinfoMethod.cs`，下面开始针对此文件的具体升级流程。

(1) 定义方法 `tbMusicinfoUpdate(tb_musicinfo tb_aut)`，功能是单击“修改”按钮对系统库内的歌曲信息进行修改。返回值是 `int` 类型，返回值是 1 时，表示修改成功，返回 0 则表示修改失败。对应的代码如下所示：

```
public int tbMusicinfoUpdate(tb_musicinfo tb_aut)
{
    int intResult = 0;
```



```

try
{
    getConnection getCon = new getConnection();
    oledCon = getCon.OledCon();
    oledCon.Open();
    string strAdd = "update tb_musicinfo set ";
    strAdd += "MusicC_name='" + tb_aut.strMusicC_name + "',";
    strAdd += "Music_author='" + tb_aut.strMusic_author + "',Music_Kind='"
        + tb_aut.strMusic_Kind + "',";
    strAdd += "Music_chinse='" + tb_aut.strMusic_chinse + "',Music_filepath='"
        + tb_aut.strMusic_filepath + "',";
    strAdd += "Music_Ping='" + tb_aut.strMusic_Ping + "',Music_date='"
        + tb_aut.daMusic_date + "',";
    strAdd += "Music_falg='" + tb_aut.intMusic_falg + "' where Music_code='"
        + tb_aut.strMusic_code + "'";

    oledcmd = new OleDbCommand(strAdd, oledCon);
    intResult = oledcmd.ExecuteNonQuery();
    return intResult;
}
catch (Exception ee)
{
    MessageBox.Show(ee.Message.ToString());
    return intResult;
}
}

```

(2) 新定义方法 `tbMusicinfoDelete(string tb_aut)`，功能是单击“删除”按钮时删除系统内不要的歌曲信息。此方法的返回值是 `int` 类型，如果返回值为 1，表示删除成功，如果返回 0，则表示删除失败。方法 `tbMusicinfoDelete(string tb_aut)` 的实现代码如下所示：

```

public int tbMusicinfoDelete(string tb_aut)
{
    int intResult = 0;
    try
    {
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();
        string strAdd = "delete * from tb_musicinfo where ";
        strAdd += "Music_code='" + tb_aut + "'";
        oledcmd = new OleDbCommand(strAdd, oledCon);
        intResult = oledcmd.ExecuteNonQuery();
        return intResult;
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.Message.ToString());
        return intResult;
    }
}

```

上述各种维护操作都涉及了对数据库数据的操作处理，细心的读者应该发现，上述代码中有很多类似的语句，此时肯定会问：难道所有的数据库项目都十分相似，都离不开查询、添加、删除和修改这些操作范畴吗？事实确实如此，无论是 Web 项目还是窗体项目，只要使用了数据库存储技术，都离不开对数据库中数据的查询、添加、修改和删除操作。唯一的区别只是用什么具体方法来实现这些操作，例如用不用存储过程。这些查询、添加、



删除和修改操作一般是基于 SQL 语言实现的, SQL 为我们提供了一整套的对数据库进行操作的方案。读者只要掌握了 SQL 语言的知识, 对数据库操作的问题便会迎刃而解。所以在此建议读者, 如果对数据库项目有困惑, 建议多了解一些 SQL 语言的知识。

### 6.9.3 增加维护明星信息模块

明星信息模块的升级工作与歌曲信息模块的升级工作类似, 在前面的添加明星信息模块是通过方法 AuthAdd 实现的, 此方法使用 SQL 中的“insert into”语句向数据库表 tb\_authorinfo 中添加添加明星信息。

同样道理, 明星信息的修改和删除操作可以使用 SQL 语句来实现。并且遵循模块化设计原则, 可以单独编写对应的操作方法。在具体实现时, 只需将所有与明星操作相关的方法放在类 tb\_authorinfoMethod 中, 并将新编写的维护明星信息方法保存在这个类中即可。

修改本模块的实现文件 tb\_authorinfoMethod.cs, 下面开始讲解具体的升级流程。

(1) 添加修改明星信息的方法 AuthUpdate, 具体代码如下所示:

```
public int AuthUpdate(tb_authorinfo tb_aut)
{
    int intResult = 0;
    try
    {
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();

        string strAdd = "update tb_authorinfo set ";
        strAdd += "authorName='" + tb_aut.strauthorName + "',authorSex='"
            + tb_aut.strauthorSex + "',authorbirthday='" + tb_aut.daauthorbirthday + "',";
        strAdd += "authorGenre='" + tb_aut.strauthorGenre + "',authorcompany='"
            + tb_aut.strauthorcompany + "',authorRecma='" + tb_aut.strauthorRecma + "',";
        strAdd += "authorzjm='" + tb_aut.strauthorzjm + "',RdateTime='"
            + tb_aut.daRdateTime + "' where authorId='" + tb_aut.intauthorId + "'";
        oledcmd = new OleDbCommand(strAdd, oledCon);
        intResult = oledcmd.ExecuteNonQuery();
        return intResult;
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.Message.ToString());
        return intResult;
    }
}
```

(2) 添加删除明星信息的方法 dictionaryDelete, 具体代码如下所示:

```
public void dictionaryDelete(string strFalg)
{
    try
    {
        getConnection getCon = new getConnection();
        oledCon = getCon.OledCon();
        oledCon.Open();
        string strAdd = "delete * from tb_authorinfo where authorId='" + strFalg + "'";
        oledcmd = new OleDbCommand(strAdd, oledCon);
        oleRed = oledcmd.ExecuteReader();
    }
}
```

```
catch (Exception ee)
{
    MessageBox.Show(ee.Message.ToString());
}
}
```

到此为止，整个项目的“功能升级”工作全部完成。此时执行项目，即可发现后台数据维护模块中已经新增了修改信息功能和删除信息功能。执行效果如图 6-25 所示。



图 6-25 升级后的执行效果

#### 6.9.4 以“人性化”为目标的功能升级

作为一个完善、科学、合理的项目，最基本的要求是做到操作“人性化”。到目前为止，本项目已经能够很好地满足用户的基本需求和升级需求。但是如果能满足“人性化”这一要求，我们还可以做得更好。

前面的升级操作中，已经为后台维护模块增加了“修改”和“删除”操作的升级，在界面中增加了“删除”和“修改”按钮。升级后整个项目的功能增强了，但是，也随之带来了新的问题：按钮过多，容易导致操作失误。例如，管理员想修改某一个管理员用户的信息，在输入修改信息后，却不小心将“删除”按钮当成了“修改”按钮，错误地将这名管理员用户的信息删除了。这样的误操作是实际项目维护过程中会遇到的常见的问题，为了避免上述误操作的发生，我们可以为本项目添加“人性化”的功能，比如，可以在管理员信息维护模块中设置按钮的可用性，具体要求如下所示：

- 当添加新用户信息时，“修改”按钮和“删除”按钮不可用。
- 当修改某个用户的信息时，“添加”按钮和“删除”按钮不可用。

为了满足上述“人性化”需求，需要进行如下所示的编码工作。

(1) 打开后台维护模块的主文件 Main.cs，修改单击 btnSave 按钮的处理事件代码，添加如下所示的加粗代码：

```
...
if (computer.tb_computerAdd(computer) == 1)
{
    MessageBox.Show("添加成功!", "提示");
    computer.tbMusicinfoFill(LvUser);
}
```

```

txtUser.Enabled = true;
txtPassWord.Text = "";
txtUser.Text = "";
btnUserAdd.Enabled = true;
btnUserDelete.Enabled = true;
btnUserUpdate.Enabled = true;
}
else
{
    MessageBox.Show("失败失败!", "提示");
    txtPassWord.Text = "";
    txtUser.Text = "";
    btnUserAdd.Enabled = true;
    btnUserDelete.Enabled = true;
    btnUserUpdate.Enabled = true;
}
...
if (intFalg == 2)
{
    if (strUser == null)
    {
        MessageBox.Show("选择要修改的用户");
        return;
    }
    else
    {
        computer.strcmp_ID = strUser;
    }
    computer.strcmp_name = txtUser.Text;
    computer.strcmp_Paww = txtPassWord.Text;
    computer.strcmp_DataTime = DateTime.Now.Date.ToString();
    computer.strcmp_Falg = "0";
    if (computer.tb_computerUpdate(computer) == 1)
    {
        MessageBox.Show("修改成功!", "提示");
        computer.tbMusicinfoFill(LvUser);
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
    else
    {
        MessageBox.Show("修改失败!", "提示");
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
}
if (intFalg == 3)
{
    if (strUser == null)
    {
        MessageBox.Show("选择要删除的用户");
        return;
    }
    else
    {

```



```

        computer.strcmp_ID = strUser;
    }
    computer.strcmp_Falg = "1";
    if (computer.tb_computerDelete(computer) == 1)
    {
        MessageBox.Show("删除成功!", "提示");
        computer.tbMusicinfoFill(LvUser);
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
    else
    {
        MessageBox.Show("删除失败!", "提示");
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
}

```

(2) 打开后台维护模块主文件 Main.cs, 重新升级修改按钮 btnUserUpdate、删除按钮 btnUserDelete、添加按钮 btnUserAdd 的事件处理代码, 添加如下所示的加粗代码:

```

private void btnUserAdd_Click(object sender, EventArgs e) //添加用户
{
    intFalg = 1;
    txtPassWord.Text = "";
    txtUser.Text = "";
    txtUser.Enabled = true;
    btnUserAdd.Enabled = true;
    btnUserDelete.Enabled = false;
    btnUserUpdate.Enabled = false;
}
private void btnUserUpdate_Click(object sender, EventArgs e) //修改用户
{
    intFalg = 2;
    btnUserAdd.Enabled = false;
    btnUserDelete.Enabled = false;
    btnUserUpdate.Enabled = true;
}
//删除用户
private void btnUserDelete_Click(object sender, EventArgs e)
{
    intFalg = 3;
    btnUserAdd.Enabled = false;
    btnUserDelete.Enabled = true;
    btnUserUpdate.Enabled = false;
}
public int intFalg = 0;

//保存用户
private void btnSave_Click(object sender, EventArgs e)
{
    if (txtUser.Text == "")
    {
        MessageBox.Show("用户名不能为空!");
    }
}

```

```

txtUser.Focus();
return;
}
if (intFalg != 3)
{
    if (txtPassWord.Text == "")
    {
        MessageBox.Show("用户密码不能为空!");
        txtPassWord.Focus();
        return;
    }
}
if (intFalg == 1)
{
    computer.strcmp_ID = computer.getSellID();
    computer.strcmp_name = txtUser.Text;
    computer.strcmp_Paww = txtPassWord.Text;
    computer.strcmp_DataTime = DateTime.Now.Date.ToString();
    computer.strcmp_Falg = "0";
    if (computer.tb_computerLogin(computer, 1) == 1)
    {
        MessageBox.Show("此用户名已被占用");
        txtUser.Text = "";
        txtUser.Focus();
        txtPassWord.Text = "";
        return;
    }
    if (computer.tb_computerAdd(computer) == 1)
    {
        MessageBox.Show("添加成功!", "提示");
        computer.tbMusicinfoFill(LvUser);
        txtUser.Enabled = true;
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
    else
    {
        MessageBox.Show("失败失败!", "提示");
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
}
if (intFalg == 2)
{
    if (strUser == null)
    {
        MessageBox.Show("选择要修改的用户");
        return;
    }
    else
    {
        computer.strcmp_ID = strUser;
    }
    computer.strcmp_name = txtUser.Text;
    computer.strcmp_Paww = txtPassWord.Text;
}

```

```

computer.strptime_DataTime = DateTime.Now.Date.ToString();
computer.strptime_Falg = "0";
if (computer.tb_computerUpdate(computer) == 1)
{
    MessageBox.Show("修改成功!", "提示");
    computer.tbMusicinfoFill(LvUser);
    txtPassWord.Text = "";
    txtUser.Text = "";
    btnUserAdd.Enabled = true;
    btnUserDelete.Enabled = true;
    btnUserUpdate.Enabled = true;
}
else
{
    MessageBox.Show("修改失败!", "提示");
    txtPassWord.Text = "";
    txtUser.Text = "";
    btnUserAdd.Enabled = true;
    btnUserDelete.Enabled = true;
    btnUserUpdate.Enabled = true;
}
}
if (intFalg == 3)
{
    if (strUser == null)
    {
        MessageBox.Show("选择要删除的用户");
        return;
    }
    else
    {
        computer.strptime_ID = strUser;
    }
    computer.strptime_Falg = "1";
    if (computer.tb_computerDelete(computer) == 1)
    {
        MessageBox.Show("删除成功!", "提示");
        computer.tbMusicinfoFill(LvUser);
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
    else
    {
        MessageBox.Show("删除失败!", "提示");
        txtPassWord.Text = "";
        txtUser.Text = "";
        btnUserAdd.Enabled = true;
        btnUserDelete.Enabled = true;
        btnUserUpdate.Enabled = true;
    }
}
}
}

```

到此为止，整个项目“人性化”操作升级工作全部结束，此时，执行后，会发现管理员用户维护模块的“修改”按钮和“删除”按钮不能同时使用，这样就不会发生误操作的情形了。执行效果如图 6-26 所示。



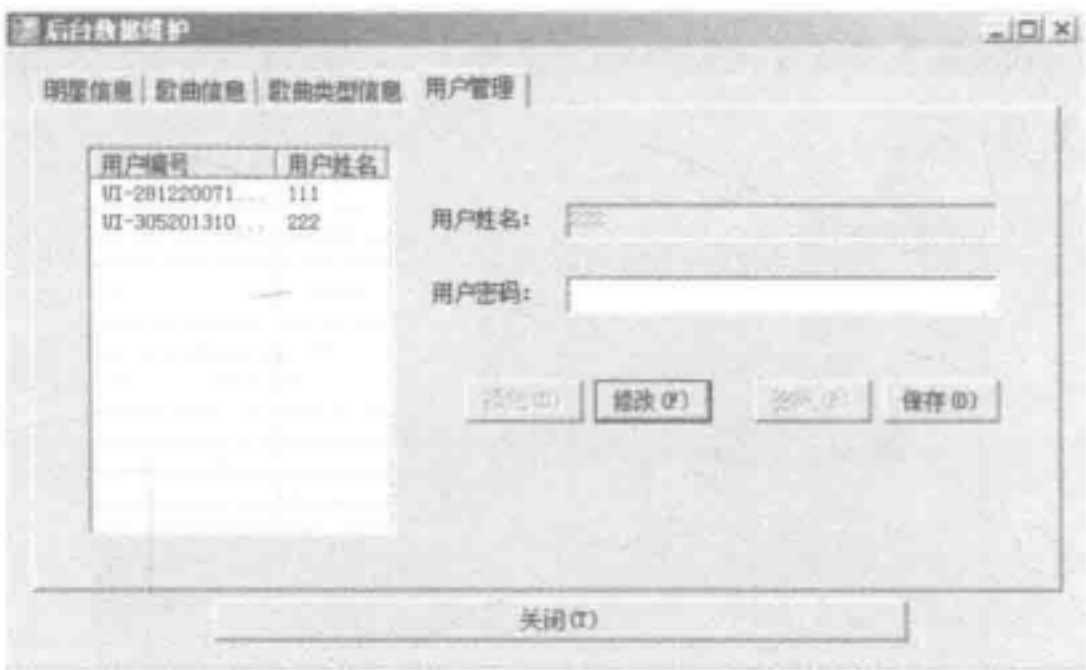


图 6-26 “修改”按钮和“删除”按钮不能同时使用

## 6.10 数据库工具升级

视频讲解 光盘：视频\第 6 章\C#项目开发中的数据库技术

数据库是 C#项目开发的核心内容之一，是 C#语言实现动态项目的根本。在本节的内容中，将详细讲解将本项目从 Access 数据库升级为 SQL Server 的方法。

### 6.10.1 导入数据

随着系统的使用和升级，数据库的容量会越来越大。如果整个系统的容量扩大到一定程度，Access 数据库就不能满足对系统的海量信息进行存储了，此时，需要对数据库进行升级。最常用的数据库是 SQL Server 2008。

要想将本项目的 Access 数据库升级为 SQL Server，需要做如下所示的工作。

- (1) 打开 SQL Server 数据库，新建一个名为 db\_KTV 的数据库，如图 6-27 所示。

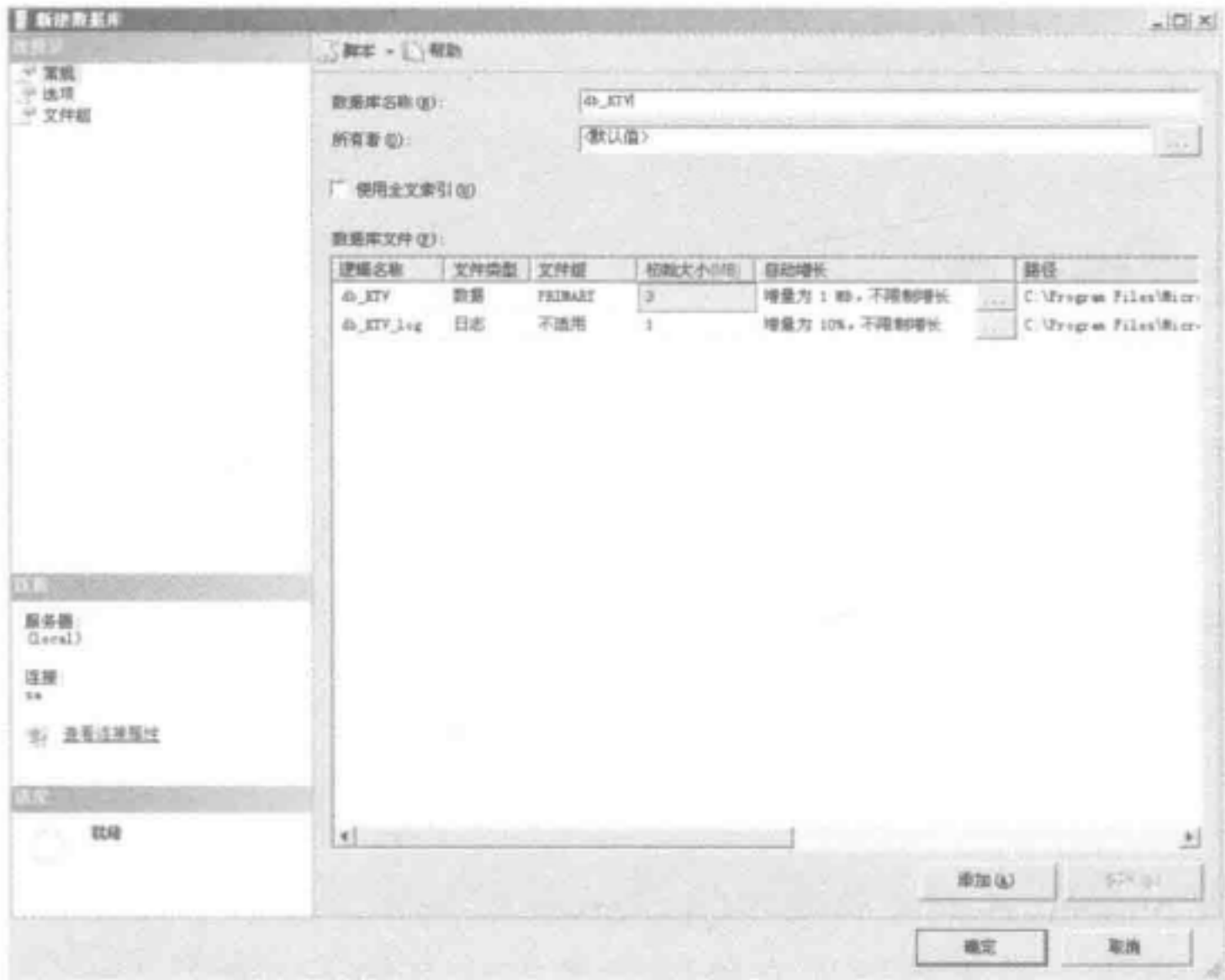
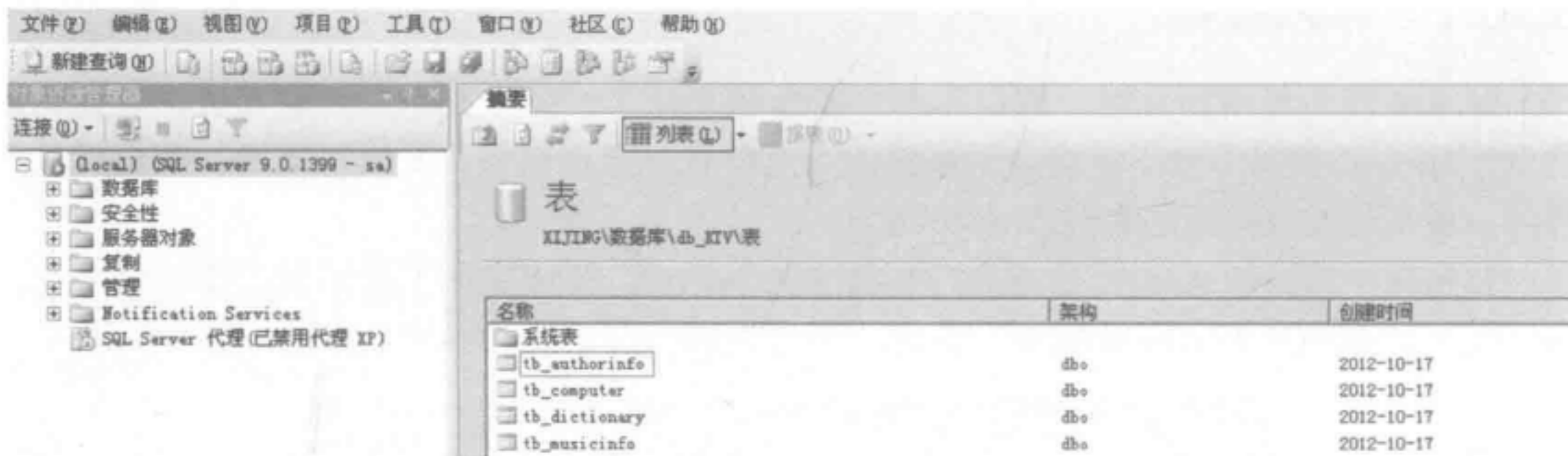


图 6-27 新建一个名为 db\_KTV 的 SQL Server 数据库

|                   |                    |
|-------------------|--------------------|
| 新建数据库 (N)...      | 分离 (D)...          |
| 新建查询 (Q)          | 脱机 (I)             |
| 编写数据库脚本为 (S) ▶    | 联机 (L)             |
| <b>任务 (T) ▶</b>   | 收缩 (S) ▶           |
| 重命名 (M)           | 备份 (B)...          |
| 删除 (D)            | 还原 (R) ▶           |
| 刷新 (F)            | 镜像 (M)...          |
| 属性 (P)            | 传送事务日志 (L)...      |
| EnterpriseManager | 生成脚本 (S)...        |
| EntLib_BBS        | <b>导入数据 (I)...</b> |
| Exam              | 导出数据 (E)...        |
| HMSys             | 复制数据库 (C)...       |
| Hospital_MIS      |                    |
| kehu              |                    |

(3) 根据默认提示,将原来 Access 数据库中的数据导入新建的 SQL Server 数据库,导入成功后的效果如图 6-29 所示。



### 6.10.2 修改连接程序

(1) 使用 OleDbConnection 对象连接 OLE DB 数据源, 格式如下:

```
provider=SQLOLEDB;  
Data Source=服务器名;  
Initial Catalog=数据库名;  
uid=用户;  
pwd=密码
```

使用绝对路径，例如：

```
"AttachDbFilename=D:\\Solution1\\Web\\App_Data\\data.mdf"
```

使用服务器相对路径, 例如:

```
"AttachDbFilename="+Server.MapPath("~/App_Data/data.mdf")
```

使用最简单的相对路径, 例如:

```
"AttachDbFilename=|DataDirectory|\\data.mdf"
```

在此推荐使用最后一种方式, 即|DataDirectory|, 代表 ASP.NET 项目里自动创建的 App\_Data 文件夹。

接下来, 开始介绍使用 SqlConnection 对象连接 SQL Server 数据库的具体代码。

#### ① 以 SQL Server 验证模式连接 SQLServer

以数据库名连接方式:

```
Server=服务器名;  
Database=数据库名称;  
User ID=用户名;  
Password=密码
```

或者(使用缩写与别名):


```
Server=服务器名;  
Initial Catalog=数据库名称;  
Uid=用户;  
Pwd=密码
```

以数据库文件完整路径连接方式, 格式是:

```
"Serve=服务器名;  
AttachDbFilename=数据库文件路径;  
User ID=用户名;  
Password=密码"
```

下面是具体的演示代码:

```
Server=.\SQLEXPRESS; Database=DatabaseName; User ID =sa; Password=abc123"  
Server=.\SQLEXPRESS; Initial Catalog =DatabaseName; Uid =sa; Pwd=abc123"  
Server=(local)\SQLEXPRESS; AttachDbFilename=D:\\Solution1\\Web\\App_Data\\data.mdf;  
User ID =sa; Password=abc123"
```

 **注意:** 此处的密码可以为空。

#### ② 以 Windows 验证模式连接

以数据库名连接方式, 格式是:

```
Server=服务器名;  
Database=数据库名称;  
Integrated Security=SSPI
```

以数据库文件完整路径连接方式, 格式是:

```
"Serve=服务器名;AttachDbFilename=数据库文件路径; Integrated Security=true"
```

下面是具体的演示代码:

```
Server=服务器名;  
Database=数据库名称;
```



```
Integrated Security=SSPI
Server=(local)\SQLEXPRESS;
AttachDbFilename=D:\Solution1\Web\App_Data\data.mdf;
Integrated Security=true"
```

 注意：SSPI 即为 true。

### 6.10.3 用 OleDbConnection 对象连接 OLE DB 数据源

看到标题后，读者肯定会有疑问：为什么使用 OleDbConnection 对象的方式进行升级？在前面内容中了解到，C#有众多连接 SQL Server 数据库的方式，但是作者在此决定采用 OleDbConnection 对象的方式。这是因为在 Access 版本中，采用的是 OleDbConnection 对象连接的 Access 数据库。在原来的程序中，随处可见如下代码片段：

```
using System.Data.OleDb;
```

并且原来版本中的数据库连接文件 getConnection.cs 中，使用了 OleDbConnection 对象，代码如下所示：

```
class getConnection
{
    public OleDbConnection OledCon()
    {
        //创建连接数据库的字符串
        string reportPath = Application.StartupPath.Substring(0,
            Application.StartupPath.Substring(0,
                Application.StartupPath.LastIndexOf(@"\")).LastIndexOf(@"\"));
        reportPath += @"\DataBase\db_KTV.mdb";
        string ConStr = "Provider=Microsoft.Jet.OLEDB.6.0;Data source=" + reportPath;
        //创建 OleDbConnection 对象
        OleDbConnection con = new OleDbConnection(ConStr);
        //con.Open();
        return con;
    }
}
```

由上述代码可知，整个项目充斥着大量的 OleDbConnection 对象和类。所以，如果使用 OleDbConnection 对象的方式进行升级，整个工作将更加简单，所付出的修改工作量也更少。

针对本项目，使用 OleDbConnection 对象连接的方式来升级数据库，本地测试的具体连接代码如下所示：

```
provider=SQLOLEDB;
Data Source=(local);
Initial Catalog=db_KTV;
uid=sa;
pwd=sa
```

## 第7章 在线商城系统

赢凡项目开发

在线商城，是指在网上建立一个在线销售平台，用户可以通过这个平台实现在线购买和提交订单，达到购买商品的目的。随着电子商务的蓬勃发展，在线商城系统在现实中得到了迅猛发展。对于售方来说，可以节省店铺的经营成本。对买方来说，可以实现即时购买，满足自己多方位的需求。

在本章的内容中，将通过一个在线商城系统实例，初步了解 C#技术在 Web 商城系统中的应用，使读者真正了解 C#开发技术的精髓；并详细介绍如何创建一个功能齐全的在线商城系统，本系统将实现用户浏览商品及实现对商品的订购，以及对订单实现管理等电子商务功能。



### 赠送的超值电子书

- 061 C#面向对象编程
- 062 统一建模语言
- 063 序列图和状态图
- 064 定义类
- 065 最简单的数据成员
- 066 最重要的函数成员
- 067 类成员访问修饰符
- 068 类的访问修饰符
- 069 创建、使用对象
- 070 类成员

## 7.1 模块化编程

视频讲解 光盘：视频\第7章\模块化编程.avi

在当今的软件开发领域中，闭门造车的软件开发时代早已过去。在 C、Java、C# 等程序开发过程中，几乎每一位开发者都需要依赖别人写的类库或框架。这种借助并复用他人提供的基础设施、框架及类库的做法，好处在于能使自己能够专注于应用本身的逻辑中，这样就能缩短软件开发所需要的时间。利用别人的现成代码和框架的过程，其实就是遵循了模块化编程的原则。开发者要想使自己的开发效率更高，模块化编程就必不可少。在本节的内容中，将引领读者一起探讨模块化编程的奥秘。

### 7.1.1 谈模块化设计思想

模块化编程是指将一个庞大的程序划分为若干个功能独立的模块，对各个模块进行独立开发，然后再将这些模块统一合并为一个完整的程序。这是 C 语言面向过程的编程方法，可以缩短开发周期，提高程序的可读性和可维护性。

#### 1. 从一个例子说起

例如，打开一个典型的 Web 站点，会发现整个站点是由不同的功能模块构成的。一个典型项目程序的基本结构如图 7-1 所示。

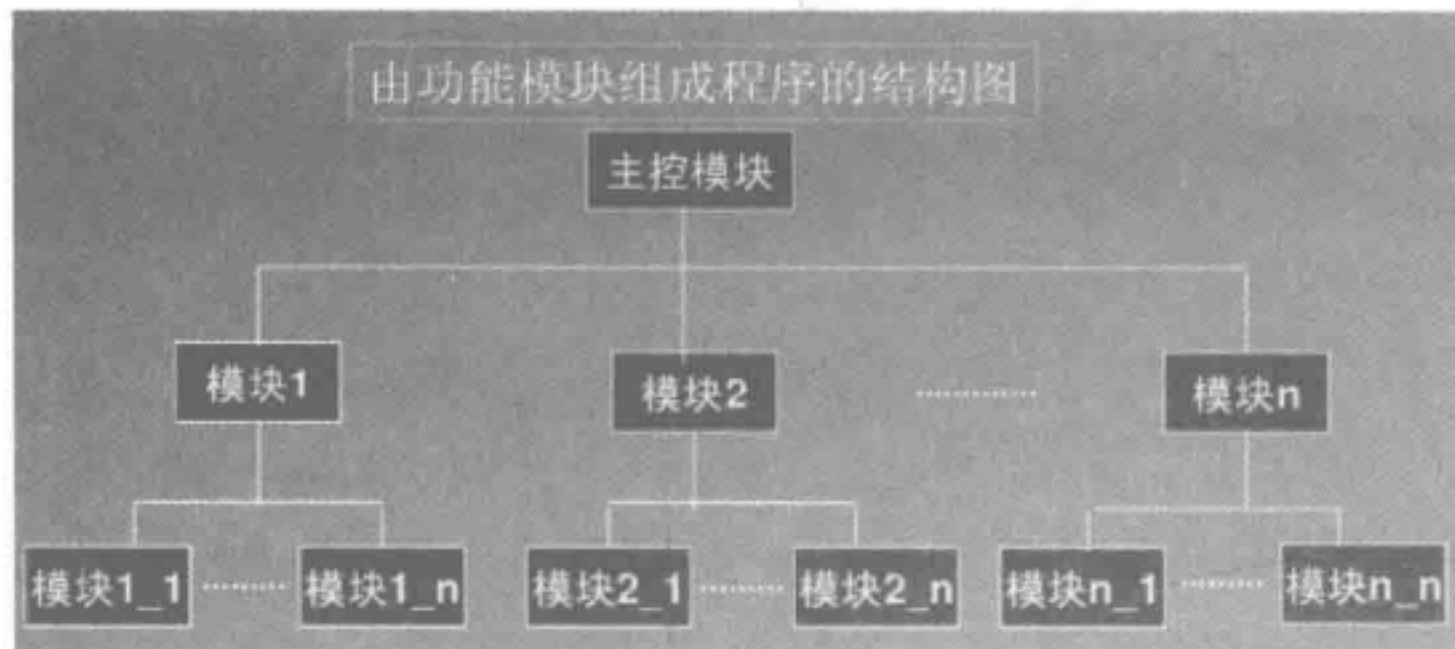


图 7-1 Web 站点的结构

再打个比方，有一个最基本的会员登录系统，可以由以下模块构成。

- (1) 表单模块：显示用户的登录表单。
- (2) 登录验证模块：验证用户输入的登录信息是否合法。

当以后系统升级时，例如验证选项有变化时，就只须对某一处进行修改即可。这样，便能够减轻后期维护的负担，提高工作效率。而在编程语言中，函数就是一个能够完成某个功能的模块。

#### 2. 再看实例代码中的函数

假如有一个题目：计算两个数的和。根据面向对象的思想，需要将功能和实现分离，即先编写一个函数实现两个数的计算，然后将计算的结果单独输出。在具体实现上，我们



可以编写一个 `sum()` 函数来实现，这个函数是完全独立的，输出结果是通过在主函数中调用 `sum()` 实现的。在此，`sum()` 函数能够实现求和的功能，我们可以将其称为一个模块。这样，当我们以后遇到求和问题时，就可以直接调用此函数了。这样将大大提高编程的效率。

在日常学习和工作过程中，建议读者收集一些有用的、能够完成某些功能的代码模块和函数，特别是常用的用户登录验证、留言板、新闻日志、信息管理等模块。这样，在日后的项目开发工作中，可以拿来这些模块直接使用，也可以稍做修改后使用，从而可以提高开发的效率。具体怎样使用这些收集的模块呢？我们来看下面的例子。

### 3. 应用

如果你接了一个企业站点项目，客户肯定会要求实现如下功能：

- 企业信息显示。
- 会员登录管理。
- 留言系统。
- 后台信息管理。

对于上述 4 个功能，企业信息显示与普通的新闻显示模块实现的原理是同样的，而会员登录管理与一般的用户登录验证管理模块原理是同样的，留言系统可以在网络中随处找到，后台管理无非是对信息的添加、删除和编辑等处理。如果我们已经收集了普通的和常用的模块的代码，就可以直接借鉴，甚至直接拿来套用，唯一需要做的工作是对这些模块稍微进行修改和修饰。

当然，我们没有必要将一个程序的所有代码都模块化，否则会适得其反。那么，什么样的代码段应该模块化呢，我们认为，应该具有以下条件。

- (1) 大量地在多个页面或程序中重复使用的代码段。
- (2) 有待进一步研究的代码段。
- (3) 程序中的关键功能、核心内容。
- (4) 能扩展第三方插件的代码段。

在 C# 语言开发领域中，为了实现某一个具体的功能，通过编写函数的方式来实现，这种函数实现方式遵循了模块化编程思想。由此可见，模块化编程就是将开发领域中常见的功能独立编码。当以后在不同的项目中用到这个功能时，就可以直接拿来使用了。

## 7.1.2 赢在模块化思想——实现高内聚和低耦合的代码

模块化编程思想的核心是高内聚和低耦合。内聚是从功能角度来衡量模块内的联系，一个好的内聚模块应当恰好做一件事，它描述的是模块内的功能联系；而耦合是软件结构中各模块之间相互连接的一种量度，耦合强弱取决于模块间接口的复杂程度、进入或访问一个模块的点以及通过接口的数据。高内聚低耦合是判断设计好坏的标准，对于面向对象的设计而言，主要是看类的内聚性是否高，耦合度是否低。

软件架构设计的目的，是在保持软件内在联系的前提下分解软件系统，降低软件系统开发的复杂性，而分解软件系统的基本方法无外乎分层和分割。但是在保持软件内在联系的前提下，如何分层分割系统，分层分割到什么样的力度，并不是一件容易的事，这方面有各种各样的分解方法，比如关注点分离、面向方面、面向对象、面向接口、面向服务、

依赖注入，以及各种各样的设计原则等。

(1) 在软件架构领域，耦合可以分为以下几种，它们之间的耦合度由高到低排列。

① 内容耦合：有下列情形之一时，两个模块就发生了内容耦合。

- 一个模块访问另一个模块的内部数据。
- 一个模块不通过正常入口而转到另一个模块的内部。
- 一个模块有多个入口。

② 公共耦合：当两个或多个模块通过公共数据环境相互作用时，它们之间的耦合称为公共环境耦合。

③ 控制耦合：如果两个模块通过参数交换信息，交换的信息有控制信息，那么这种耦合就是控制耦合。

④ 特征耦合：如果被调用的模块需要使用作为参数传递进来的数据结构中的所有数据时，那么把这个数据结构作为参数整体传送是完全正确的。但是，当把整个数据结构作为参数传递而使用其中一部分数据元素时，就出现了特征耦合。在这种情况下，被调用的模块可以使用的数据多于它确实需要的数据，这将导致对数据的访问失去控制，从而给计算机犯错误提供机会。

⑤ 数据耦合：如果两个模块通过参数交换信息，而且交换的信息仅仅是数据，那么这种耦合就是数据耦合。

(2) 在软件架构领域，内聚有如下所示的种类，它们之间的内聚度由弱到强排列。

① 偶然内聚：模块中的代码无法定义其不同功能的调用。但它使该模块能执行不同的功能，这种模块又称为巧合强度模块。

② 逻辑内聚。这种模块把几种相关的功能组合在一起，每次被调用时，通过传送给模块参数来确定该模块应该完成哪一种功能。

③ 时间内聚：把需要同时执行的动作组合在一起，形成的模块为时间内聚模块。

④ 过程内聚：构件或者操作的组合方式，是允许在调用前面的构件或操作之后，马上调用后面的构件或操作，即使两者之间没有数据进行传递。

⑤ 通信内聚：指模块内所有处理元素都在同一个数据结构上操作(有时称为信息内聚)，或者指各处理使用相同的输入数据或者产生相同的输出数据。

⑥ 顺序内聚：指一个模块中各个处理元素都密切相关于同一功能且必须顺序执行，前一功能元素的输出就是下一功能元素的输入。

⑦ 功能内聚：共同完成同一功能，缺一不可，模块不可再分割。

高内聚、低耦合的系统有什么好处呢？事实上，从短期来看，并没有很明显的好处，甚至短期内不会影响系统的开发进度，因为高内聚、低耦合的系统对开发设计人员提出了更高的要求。高内聚、低耦合的好处体现在系统持续发展的过程中。高内聚、低耦合的系统具有更好的重用性、维护性、扩展性，可以更高效地完成系统的维护开发，持续地支持业务的发展，而不会成为业务发展的障碍。

综上所述，模块化编程就是模块合并的过程，就是建立每个模块的头文件和源文件并将其加入到主体程序的过程。主体程序调用模块的函数是通过包含模块的头文件来实现的，模块的头文件和源文件是模块密不可分的两个部分，缺一不可。所以，模块化编程必须提供每个模块的头文件和源文件。



## 7.2 新的项目

视频讲解 光盘：视频\第7章\新的项目.avi

本项目的客户是一家鲜花配送公司，客户提出了如下3点要求。

- (1) 每个商品都可以留言。
- (2) 实现在线购物车处理和订单处理。
- (3) 实现对产品、购物车和订单的管理功能。

本项目的开发团队组成如下所示。

- 项目经理 DP：负责前期功能分析，策划和构建系统模块，检查项目的进度和质量。
- 软件工程师 A：负责配置系统文件、搭建数据库、实现数据访问层。
- 软件工程师 B：负责购物车处理模块、订单处理模块、商品评论模块、商品搜索模块的编码工作。
- 软件工程师 C：负责样式设计、系统测试、后期调试，并负责商品显示模块、商品分类模块、商品管理模块的编码工作。

本项目的具体职责流程如图 7-2 所示。

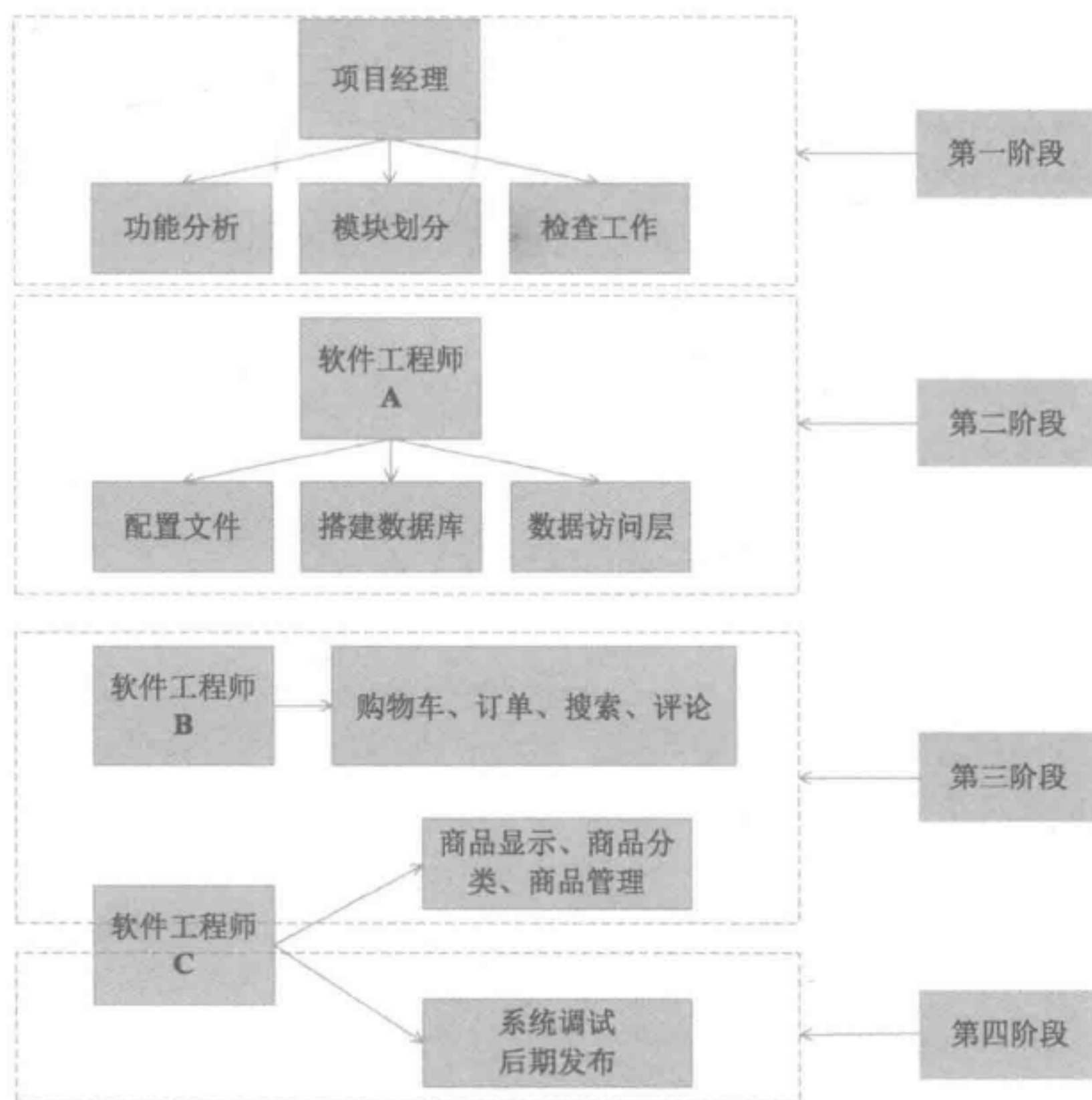


图 7-2 职责流程图



## 7.3 项目规划分析

视频讲解 光盘：视频\第7章\项目规划分析.avi

一个典型的在线商城系统的构成模块如下。

### (1) 会员处理模块

为了方便用户购买商品，提高人气，设立会员功能。成为系统会员后，可以对自己的资料进行管理，并且可以集中管理自己的订单。会员系统必须具备以下功能。

- 会员注册：能够通过注册表单成为系统内的新会员。
- 会员管理：不但能管理个人的基本信息，而且能够管理自己的订单信息。

### (2) 购物车处理模块

作为网上商城系统必不可少的环节，为满足用户的购物需求，设立购物车功能。用户可以把需要的商品放到购物车中保存，提交在线订单后即可完成在线商品的购买。

### (3) 商品查询模块

为了方便用户购买，系统设立商品快速查询模块，用户可以根据商品的信息快速找到自己需要的商品。

### (4) 订单处理模块

为方便商家处理用户的购买信息，系统设立订单处理功能。通过该功能，可以实现对用户购物车信息的及时处理，使用户能够尽快地拿到自己的商品。

### (5) 商品分类模块

为了便于用户对系统商品的浏览，将系统的商品划分为不同的类别，以便用户迅速找到自己需要的商品类别。

### (6) 商品管理模块

为方便对系统的升级和维护，建立专用的商品管理模块，实现商品的添加、删除和修改功能，以满足系统更新的需求。

## 7.4 规划项目文件

视频讲解 光盘：视频\第7章\规划项目文件.avi

(1) 分别创建两个文件夹“shop”和“data”来保存项目文件，具体说明如下。

- shop：保存系统的项目文件。
- data：保存系统的数据库文件。

(2) 为整个项目规划具体的实现文件，各构成模块文件的具体说明如下所示。

- 系统配置文件：功能是对项目程序进行总体配置。
- 样式设置模块：功能是设置系统文件的显示样式。
- 数据库文件：功能是搭建系统数据库平台，保存系统的登录数据。
- 商品显示模块：功能是将系统内的商品逐一显示出来。

- 购物车处理模块：功能是将满意的系统商品放在购物车内。
- 订单处理模块：功能是实现系统内购物订单的处理。
- 商品评论模块：功能是供用户对系统内的某商品发布评论。
- 商品搜索模块：功能是使用户迅速地搜索出自己需要的商品。
- 商品分类模块：功能是将系统内的商品类别以指定样式显示出来。
- 系统管理模块：功能是对系统内的数据进行管理和维护。

在此再次声明规划阶段的重要性。作为一个全新的项目，商城系统我们没有做过。但是仔细分析了网络上的一些在线购物系统后，基本功能就了解得差不多了。

任何购物系统都需要几个核心功能：商品展示、购物车处理、订单处理。只要设计好上述必需的核心功能，然后在此基础上进行扩充即可。总结完毕之后，决定早点休息，为接下来的配置工作做准备。

## 7.5 系统配置文件

视频讲解 光盘：视频\第7章\系统配置文件.avi

本项目要求使用 SQL Server 2008 数据库，并用 Ajax 技术实现无刷新处理。

Ajax 是 Asynchronous JavaScript and XML(异步 JavaScript 和 XML)的缩写，随着当前网络技术的发展，Ajax 迅速成为当前最为火爆的技术之一。Ajax 是一种创建交互式网页的开发技术。其中，XMLHttpRequest 是最为核心的内容，它能够为页面中的 JavaScript 脚本提供特定的通信方式，从而使页面通过 JavaScript 脚本与服务器之间实现动态交互。另外，XMLHttpRequest 的最大优点是页面内的 JavaScript 脚本可以不用刷新页面而直接与服务器完成数据交互。Ajax 技术的最大好处是实现无刷新处理，在浏览者眼中形成一个模式：页面没有刷新却实现了数据处理和交互。

在传统的 Web 应用模型中，浏览器负责向服务器提出访问请求，并显示服务器返回的处理结果。而在 Ajax 处理模型中，使用了 Ajax 中间引擎来处理上述通信。Ajax 中间引擎实质上是一个 JavaScript 对象或函数，只有当信息必须从服务器上获得的时候才调用它。与传统的处理模型不同，Ajax 不再需要为其他资源提供链接，而只是当需要调度和执行时才执行这些请求。而这些请求都是通过异步传输完成的，不必等到收到响应之后才执行。

当 Ajax 引擎收到服务器的响应时，将会触发一些操作，通常是完成数据解析，以及基于所提供的数据对用户界面做一些修改。图 7-3 和图 7-4 分别列出了传统模型和 Ajax 模型的处理方式。



图 7-3 传统模型的处理方式

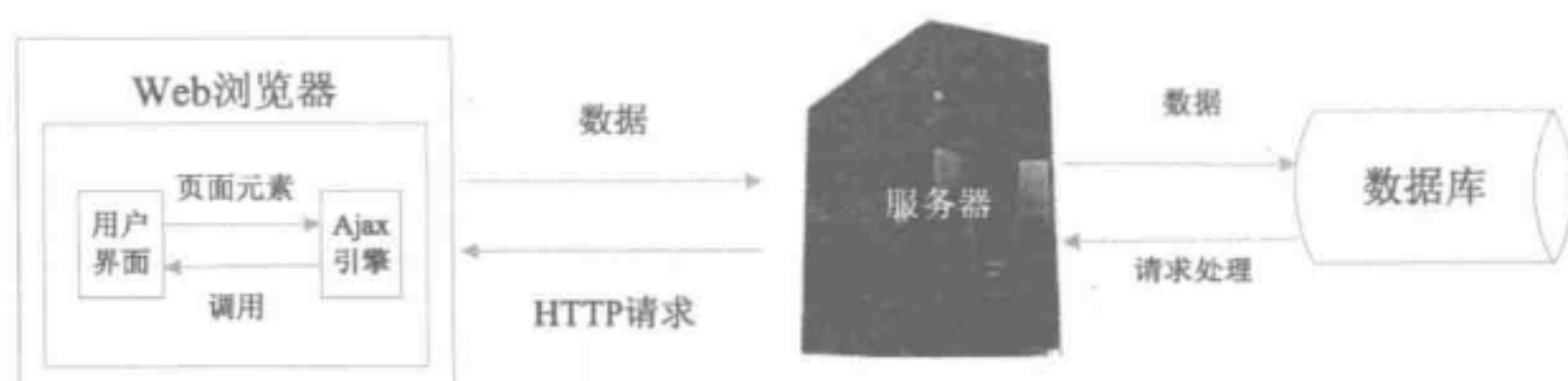


图 7-4 Ajax 模型的处理方式

根据用户的无刷新效果要求，首先根据用户的需求编写了配置文件 Web.config，其主要功能是设置数据库的连接参数，并配置了系统与 Ajax 服务器的相关内容。

### 1. 配置连接字符串参数

配置连接字符串参数即设置系统程序连接数据库的参数，其对应的实现代码如下：

```

<connectionStrings>
  <add name="SQLCONNECTIONSTRING"
    connectionString="data source=(local);user id=sa;pwd=888888;database=bookshop"
    providerName="System.Data.SqlClient"/>
</connectionStrings>
    
```

其中，source 设置连接的数据库服务器；user id 和 pwd 分别指定数据库的登录名和密码；database 设置连接数据库的名称。

### 2. 配置 Ajax 服务器参数

配置 Ajax 服务器参数即配置 Ajax Control Toolkit 程序集参数，为 AjaxControlToolkit.dll 程序集提供了一个前缀字符串“AjaxControlToolkit”。

这样，系统页面在引用 AjaxControlToolkit.dll 中的控件时，不需要额外添加<Register>代码了。

上述功能在<controls>元素内的对应实现代码如下所示：

```

<pages>
  <controls>
    <add namespace="AjaxControlToolkit" assembly="AjaxControlToolkit"
      tagPrefix="ajaxToolkit"/>
    <add tagPrefix="asp" namespace="System.Web.UI"
      assembly="System.Web.Extensions, Version=1.0.61027.0, Culture=neutral,
      PublicKeyToken=31bf3856ad364e35"/>
  </controls>
</pages>
    
```

## 7.6 搭建数据库

视频讲解 光盘：视频\第 7 章\搭建数据库.avi

数据库是动态网站的根本、核心和桥梁，本项目采用 SQL Server 2008 数据库，数据库名为“shop”。



7.6.1 数据库的设计

数据库 shop 内共有 8 个表。

(1) Attribute 表的具体设计结构如表 7-1 所示。

表 7-1 商品属性信息表(Attribute)

| 字段名称       | 数据类型          | 是否主键 | 默 认 值 | 功能描述   |
|------------|---------------|------|-------|--------|
| ID         | int           | 是    | 递增 1  | 编号     |
| CategoryID | int           | 否    | Null  | 类别编号   |
| Name       | varchar(50)   | 否    | Null  | 属性编号   |
| Text       | varchar(50)   | 否    | Null  | 属性名称   |
| DataType   | varchar(10)   | 否    | Null  | 属性数据格式 |
| Unit       | varchar(10)   | 否    | Null  | 单位     |
| Remark     | varchar(1000) | 否    | Null  | 备注     |

(2) Category 表的具体设计结构如表 7-2 所示。

表 7-2 系统商品类别信息表(Category)

| 字段名称      | 数据类型        | 是否主键 | 默 认 值 | 功能描述   |
|-----------|-------------|------|-------|--------|
| ID        | int         | 是    | 递增 1  | 编号     |
| Name      | varchar(50) | 否    | Null  | 名称     |
| ParentID  | int         | 否    | Null  | 所属父类编号 |
| ShowOrder | int         | 否    | Null  | 显示顺序   |
| Remark    | text        | 否    | Null  | 备注     |

(3) Order 表的具体设计结构如表 7-3 所示。

表 7-3 系统订单信息表(Order)

| 字段名称        | 数据类型        | 是否主键 | 默 认 值 | 功能描述 |
|-------------|-------------|------|-------|------|
| ID          | int         | 是    | 递增 1  | 编号   |
| OrderNo     | varchar(50) | 否    | Null  | 订单编号 |
| UserID      | int         | 否    | Null  | 用户编号 |
| CreateDate  | datetime    | 否    | Null  | 时间   |
| TotalNumber | int         | 否    | Null  | 商品数  |
| TotalMoney  | money       | 否    | Null  | 金额   |
| Status      | tinyint     | 否    | Null  | 状态   |

(4) OrderItem 表的具体设计结构如表 7-4 所示。

表 7-4 订单详情信息表(OrderItem)

| 字段名称      | 数据类型 | 是否主键 | 默 认 值 | 功能描述    |
|-----------|------|------|-------|---------|
| ID        | int  | 是    | 递增 1  | 编号      |
| OrderID   | int  | 否    | Null  | 订单编号    |
| ProductID | int  | 否    | Null  | 商品编号    |
| Number    | int  | 否    | Null  | 一种商品的数量 |

(5) Product 表的具体设计结构如表 7-5 所示。

表 7-5 系统商品信息表(Product)

| 字段名称       | 数据类型         | 是否主键 | 默 认 值 | 功能描述   |
|------------|--------------|------|-------|--------|
| ID         | int          | 是    | 递增 1  | 编号     |
| Name       | int          | 否    | Null  | 名称     |
| Remark     | text         | 否    | Null  | 说明     |
| Price      | money        | 否    | Null  | 价格     |
| Stock      | int          | 否    | Null  | 库存数    |
| SaleNumber | int          | 否    | Null  | 销售数    |
| PictureUrl | varchar(255) | 否    | Null  | 图片地址   |
| CategoryID | int          | 否    | Null  | 所属类别编号 |
| UserID     | int          | 否    | Null  | 所属用户   |
| CreateDate | datetime     | 否    | Null  | 上架时间   |
| LasterDate | datetime     | 否    | Null  | 最后浏览时间 |
| ViewCount  | int          | 否    | Null  | 浏览次数   |
| Status     | tinyint      | 否    | Null  | 状态     |

(6) ProductAttribute 表的具体设计结构如表 7-6 所示。

表 7-6 商品属性信息表(ProductAttribute)

| 字段名称        | 数据类型 | 是否主键 | 默 认 值 | 功能描述 |
|-------------|------|------|-------|------|
| ID          | int  | 是    | 递增 1  | 编号   |
| ProductID   | int  | 否    | Null  | 商品编号 |
| AttributeID | int  | 否    | Null  | 属性编号 |
| Value       | text | 否    | Null  | 属性值  |

(7) ProductComment 表的具体设计结构如表 7-7 所示。

表 7-7 商品评论信息表(ProductComment)

| 字段名称       | 数据类型          | 是否主键 | 默 认 值 | 功能描述 |
|------------|---------------|------|-------|------|
| ID         | int           | 是    | 递增 1  | 编号   |
| Title      | varchar(50)   | 否    | Null  | 标题   |
| Body       | varchar(1000) | 否    | Null  | 内容   |
| IP         | varchar(50)   | 否    | Null  | IP   |
| Email      | varchar(255)  | 否    | Null  | 邮箱   |
| CreateDate | datetime      | 否    | Null  | 时间   |
| ProductID  | int           | 否    | Null  | 商品编号 |

(8) User 表的具体设计结构如表 7-8 所示。

表 7-8 系统用户信息表(User)

| 字段名称       | 数据类型         | 是否主键 | 默 认 值 | 功能描述 |
|------------|--------------|------|-------|------|
| ID         | int          | 是    | 递增 1  | 编号   |
| Username   | varchar(50)  | 否    | Null  | 用户名  |
| Password   | varchar(255) | 否    | Null  | 密码   |
| Email      | varchar(255) | 否    | Null  | 邮箱   |
| TelePhone  | varchar(50)  | 否    | Null  | 电话   |
| Address    | varchar(200) | 否    | Null  | 地址   |
| Postcode   | varchar(50)  | 否    | Null  | 邮编   |
| CreateDate | datetime     | 否    | Null  | 时间   |
| State      | tinyint      | 否    | Null  | 状态   |
| Remark     | varchar(100) | 否    | Null  | 备注   |

在上述表设计的过程中， Order 表和 OrderItem 表是不同的，前者的功能是保存订单的整体信息，而后者的功能是保存订单的详细信息。例如，在 Order 表中，将保存某订单的商品总数、金额总数和时间等信息。而在 OrderItem 表中将保存一系列商品信息，包括商品的编号和单价等，然后使用订单编号对每一商品信息进行标识。这样，当查看某一订单详情时，将调用 Order 表来显示订单的整体信息，调用 OrderItem 表来显示订单内的各商品信息。

7.6.2 设置系统参数

Global.asax 文件的功能是实现页面载入、结束和错误初始化，并保存系统的登录数据。具体实现代码如下所示：

```
<%@ Application Language="C#" %>
<script runat="server">
    void Application_Start(object sender, EventArgs e)
    {
        void Application_End(object sender, EventArgs e)
```



```
{  
void Application_Error(object sender, EventArgs e)  
{  
void Session_Start(object sender, EventArgs e)  
{  
void Session_End(object sender, EventArgs e)  
{  
</script>
```

只有在 Web.config 文件中的 sessionstate 模式设置为 InProc 时，才会引发 Session\_End 事件。如果会话模式设置为 StateServer 或 SQLServer，则不会引发该事件。

## 7.7 数据访问层

视频讲解 光盘：视频\第7章\数据访问层.avi

数据访问层是整个项目的核心和难点，整个数据访问层分为如下所示的 5 个部分：商品显示、订单处理、商品评论、商品分类、商品管理。A 深知数据访问层的重要性，为了便于后期维护，专门编写了独立文件来实现。

### 7.7.1 商品显示

在数据访问文件 Product.cs 中，与商品显示相关的方法如下：

- GetProducts()
- GetProductByFenlei(int categoryID)
- GetSingleProduct(int productID)
- UpdateProductViewCount(int productID)

上述方法的运行流程如图 7-5 所示。

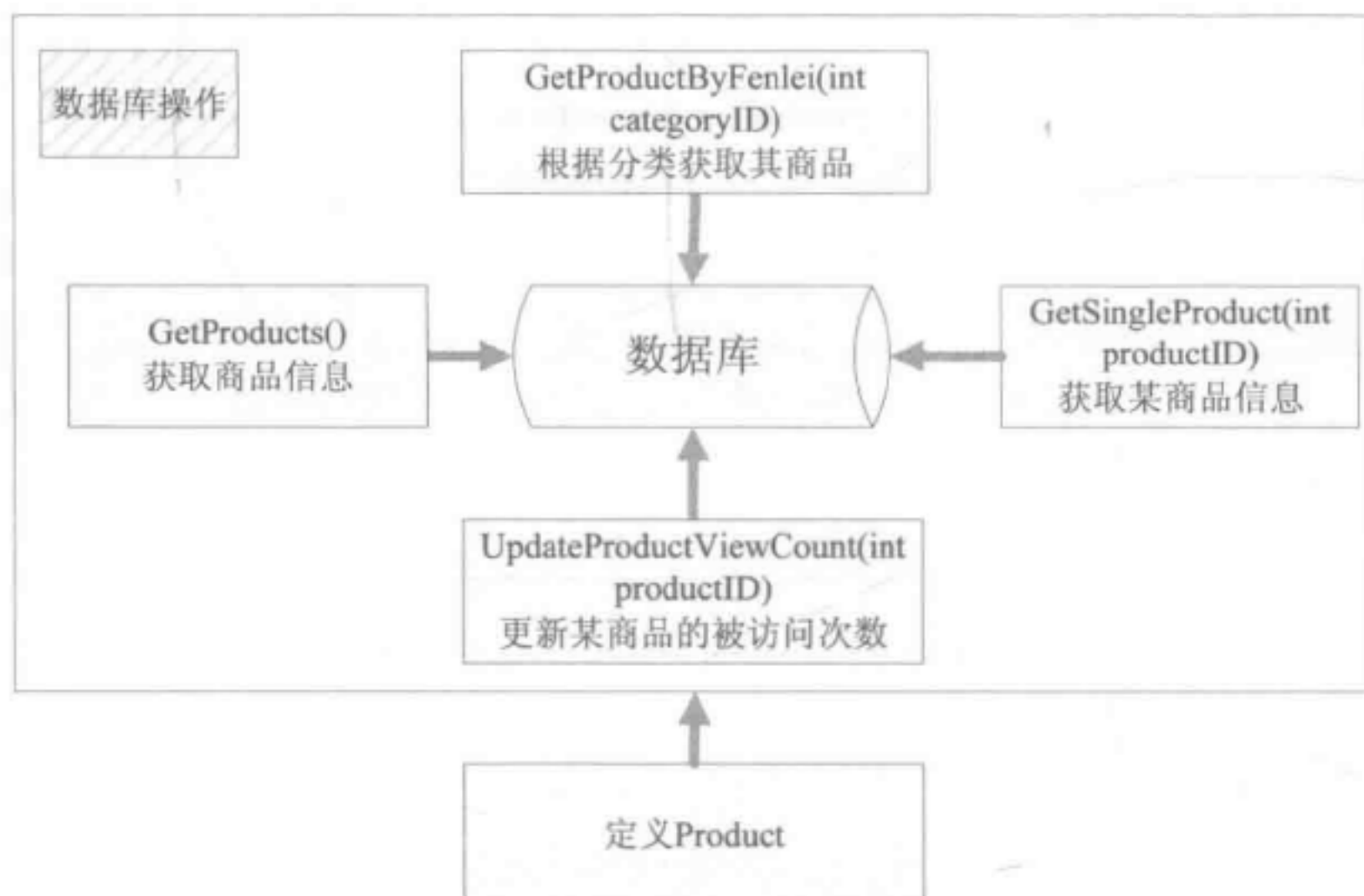


图 7-5 商品显示模块数据访问层的运行流程

## 1. 定义 Product 类

定义 Product 类的实现代码如下所示:

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;
using System.Web.UI.WebControls;
namespace ASPNETAJAXWeb.AjaxEBusiness
{
    public class Product
    {
        public Product()
        {
            ...
        }
    }
}
```

## 2. 获取商品信息

获取商品信息即获取系统库内存在的所有商品信息,该功能是由方法 GetProducts()实现的,具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数,将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象,实现数据库连接。
- (3) 调用获取所有商品信息的存储过程 Pr\_GetProducts,获取系统商品的基本信息。
- (4) 创建获取数据的对象 da。
- (5) 把对象 da 的执行方式设置为存储过程。
- (6) 打开数据库连接,获取数据,将获取的数据保存在 ds 中。
- (7) 操作成功,返回 ds。

上述功能的对应实现代码如下所示:

```
public DataSet GetProducts()
{
    ///获取连接字符串
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "Pr_GetProducts";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///设置执行方式为存储过程
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        ///打开连接
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
}
```

```

        catch(Exception ex)
        {
            ///抛出异常
            throw new Exception(ex.Message, ex);
        }
        finally
        {
            ///关闭连接
            con.Close();
        }
        return ds;
    }
}

```

### 3. 获取分类商品信息

所谓获取分类商品信息，即根据分类参数获取其对应的商品信息，这个功能是由方法 `GetProductByFenlei(int categoryID)` 实现的，具体实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。
- (2) 使用连接字符串创建 `con` 对象，实现数据库连接。
- (3) 调用存储过程 `Pr_GetProductByFenlei`，获取系统中所有商品的基本信息。
- (4) 创建获取数据的对象 `da`。
- (5) 把对象 `da` 的执行方式设置为存储过程。
- (6) 打开数据库连接，获取数据，将获取的数据保存在 `ds` 中。
- (7) 操作成功，返回 `ds`。

上述功能的对应实现代码如下所示：

```

public DataSet GetProductByFenlei(int categoryID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "Pr_GetProductByFenlei";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///设置执行方式为存储过程
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@CategoryID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = categoryID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch(Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {

```



```

        con.Close();
    }
    return ds;
}

```

#### 4. 获取指定商品的信息

获取指定商品的信息即获取系统数据库内指定编号的商品信息，这个功能是由方法 `GetSingleProduct(int productID)` 实现的，具体实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。
- (2) 使用连接字符串创建 `con` 对象，实现数据库连接。
- (3) 新建 SQL 查询语句，获取指定 ID 商品的信息。
- (4) 创建获取数据的对象 `cmd`。
- (5) 打开数据库连接，获取数据，将获取的数据保存在 `dr` 中。
- (6) 操作成功，返回 `dr`。

上述功能的对应实现代码如下所示：

```

public SqlDataReader GetSingleProduct(int productID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT [Product].*, [User].Username FROM Product INNER JOIN [User]
        ON [Product].UserID = [User].ID WHERE [Product].ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = productID;
    ///定义 SqlDataReader
    SqlDataReader dr;
    try
    {
        con.Open();
        ///读取数据
        dr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    return dr;
}

```

#### 5. 更新浏览信息

更新浏览信息即对系统数据库内某商品的被浏览次数进行更新处理，该功能是由方法 `UpdateProductViewCount(int productID)` 实现的，具体实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数，将其保存在 `connectionString` 连接字符串中。

- (2) 使用该连接字符串创建 con 对象，实现数据库连接。
- (3) 新建 SQL 更新语句，修改指定 ID 商品的被浏览次数。
- (4) 创建获取数据的对象 cmd。
- (5) 打开数据库链接获取数据，将获取的数据保存在 result 中。
- (6) 操作成功，返回 result。

上述功能的对应实现代码如下所示：

```
public int UpdateProductViewCount(int productID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;

    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行的 SQL 语句
    string cmdText = "UPDATE [Product] SET ViewCount=ViewCount+1 WHERE ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = productID;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}
```

在此总结一下数据处理效率的问题。因为数据库技术是动态项目的基础，所以在 Web 程序内会有大量的查询语句。同时随着站点访问量的增加，一个站点可能同时需要查询大量的数据，所以数据库查询的效率问题便提上了议程。

在此向读者提出如下两条建议。

#### (1) 合理使用索引

并不是所有索引对查询都有效，SQL 是根据表中的数据来进行查询优化的，当索引列有大量数据重复时，SQL 查询可能不会去利用索引，如一表中有字段 sex，其 male、female 几乎各一半，那么即使在 sex 上建了索引，也对查询效率起不了作用。读者可以在百度中通过检索“索引效率优化”关键字来获取相关的知识。

#### (2) 使用存储过程

存储过程是一个很好的工具，不但提高了程序的安全性，而且也提高了数据处理效率。编写合理的语句可以决定存储过程和触发器的效率。

## 7.7.2 订单处理

订单处理模块的数据访问层是由文件 ShoppingCart.cs 实现的，其主要功能是在 ASPNETAJAXWeb.AjaxEBusiness 空间内建立 Order 类，并定义多个方法来实现数据库数据的处理。在文件 ShoppingCart.cs 中，与订单处理模块相关的方法如下：

- GetOrderLastOrderNo()
- GetOrderByUser(int userID)
- GetSingleOrder(int orderID)
- GetOrderItemByOrder(int orderID)
- AddOrder(string orderNo, int userID, int totalNumber, decimal totalMoney)
- AddOrderItem(int orderID, int productID, int number)
- UpdateOrderStatus(int orderID, byte status)

上述方法的运行流程如图 7-6 所示。

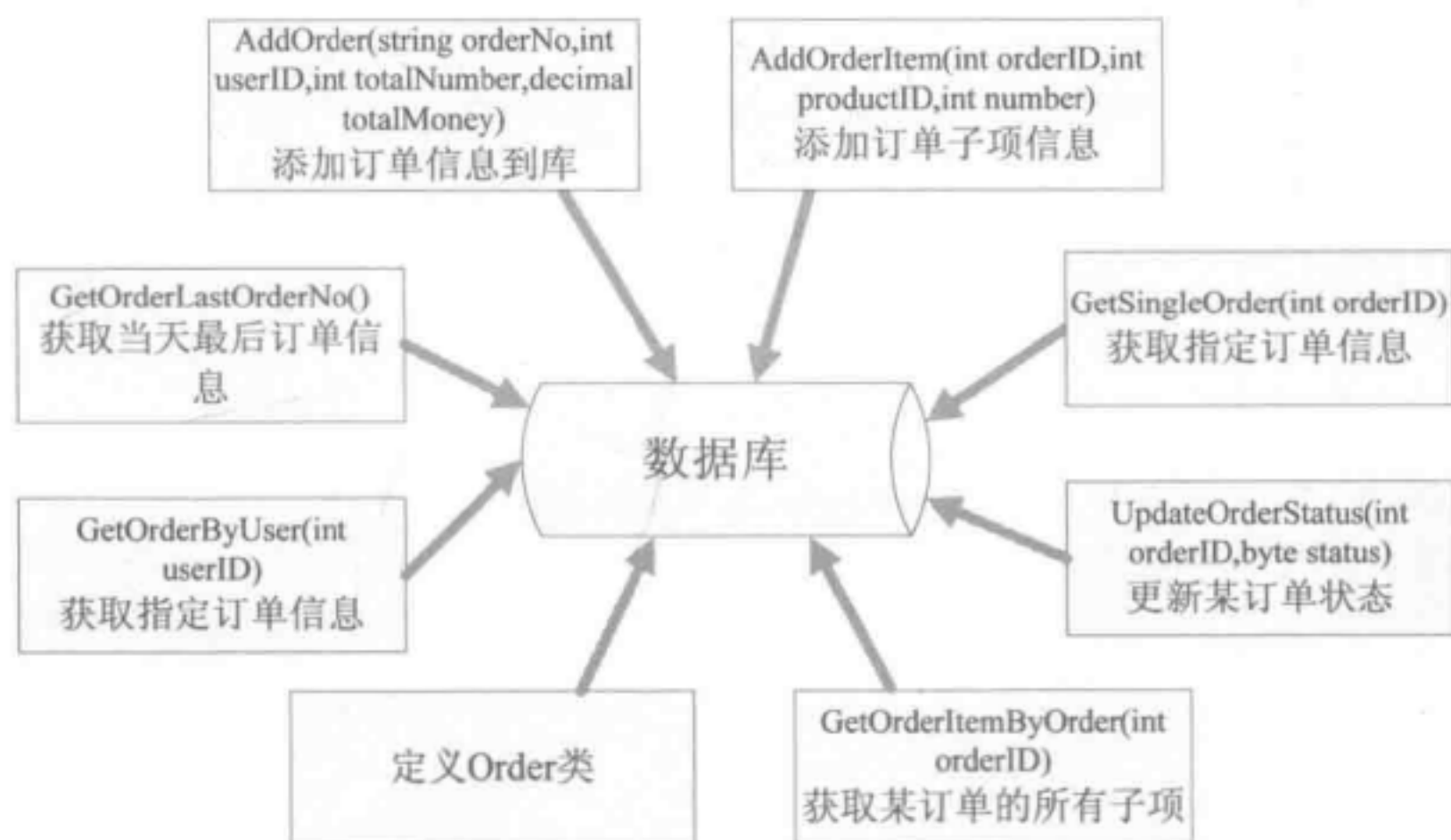


图 7-6 订单处理模块数据访问层的运行流程

### 1. 定义 Order 类

定义 Order 类的实现代码，如下所示：

```

using System;
using System.Data;
using System.Collections;
using System.Configuration;
using System.Data.SqlClient;
using System.Web.SessionState;
namespace ASPNETAJAXWeb.AjaxEBusiness
...
public class Order
{
    public Order()
    {
    }
}
...
  
```



## 2. 获取最后订单信息

获取最后订单信息即获取当天内最后一个订单的信息。上述功能是由方法 `GetOrderLastOrderNo()` 实现的，其具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。

(2) 使用该连接字符串创建 `con` 对象，实现数据库连接。

(3) 新建 SQL 查询语句，获取当天最后一个订单编号的信息。

(4) 创建获取数据的对象 `cmd`。

(5) 打开数据库连接，获取查询数据。

(6) 将获取的查询结果保存在 `orderNo` 中，并返回 `orderNo`。

上述功能的对应实现代码如下所示：

```
public string GetOrderLastOrderNo()
{
    ///获取连接字符串
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行 SQL 语句
    string cmdText = "SELECT TOP 1 [Order].OrderNo FROM [Order] WHERE
DATEDIFF(year,CreateDate,GETDATE()) = 0 AND DATEDIFF(month,CreateDate,GETDATE()) = 0 AND
DATEDIFF(day,CreateDate,GETDATE()) = 0 ORDER BY CreateDate DESC";
    ///创建 SqlDataAdapter
    SqlCommand cmd = new SqlCommand(cmdText, con);
    object orderNo;
    try
    {
        ///打开连接
        con.Open();
        ///填充数据
        orderNo = cmd.ExecuteScalar();
    }
    catch (Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
        con.Close();
    }
    return orderNo==null? string.Empty : orderNo.ToString();
}
```

在数据库操作中，与时间段相关的操作处理比比皆是。在上面获取当天最后订单信息的处理过程中，在 SQL 语句中使用了时间段的相关操作。

具体语句的说明如下。

(1) `DATEDIFF(year, CreateDate, GETDATE()) = 0`：功能是比较当前记录的年份和当前的实际年份是否相等。

(2) DATEDIFF(month, CreateDate, GETDATE()) = 0: 功能是比较当前记录的月份和当前的实际月份是否相等。

(3) DATEDIFF(day, CreateDate, GETDATE()) = 0: 功能是比较当前记录的日期和当前的实际日期是否相等。

如果上述 3 个条件都成立, 则说明该记录为当天的记录数据。

### 3. 获取某用户的订单信息

获取某用户订单信息即获取系统内某指定用户的所有订单信息。上述功能是由方法 GetOrderByUser(int userID)实现的, 其具体实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。

(2) 使用该连接字符串创建 con 对象, 实现数据库连接。

(3) 新建 SQL 查询语句, 获取指定编号用户的所有订单信息。

(4) 创建获取数据的对象 da。

(5) 打开数据库连接, 获取查询数据。

(6) 将获取的查询结果保存在 ds 中, 并返回 ds。

上述功能的对应实现代码如下所示:

```
public DataSet GetOrderByUser(int userID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;

    SqlConnection con = new SqlConnection(connectionString);

    ///设置被执行 SQL 语句
    string cmdText = "SELECT [Order].* FROM [Order] WHERE [Order].UserID=@UserID ORDER
BY CreateDate DESC";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@UserID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = userID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return ds;
}
```

#### 4. 获取某订单信息

获取某订单信息即获取系统内某编号订单的详细信息。

该功能是由方法 `GetSingleOrder(int orderID)` 实现的，其具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。

(2) 使用连接字符串创建 `con` 对象，实现数据库连接。

(3) 新建 SQL 查询语句，获取指定编号订单的详细信息。

(4) 创建获取数据的对象 `cmd`。

(5) 打开数据库连接，获取查询数据。

(6) 将获取的查询结果保存在 `dr` 中，并返回 `dr`。

上述功能的对应实现代码如下所示：

```
public SqlDataReader GetSingleOrder(int orderID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT [Order].*, OrderItem.ProductID, OrderItem.Number FROM [Order]
INNER JOIN OrderItem ON [Order].ID = OrderItem.OrderID WHERE [Order].ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = orderID;
    ///定义 SqlDataReader
    SqlDataReader dr;
    try
    {
        con.Open();
        ///读取数据
        dr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    return dr;
}
```

#### 5. 获取某订单子项信息

获取某订单子项信息即获取系统内某编号订单的所有子项信息。

该功能是由方法 `GetOrderItemByOrder(int orderID)` 实现的，具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。

(2) 使用该连接字符串创建 `con` 对象，实现数据库连接。

(3) 新建 SQL 查询语句，获取指定编号订单子项的详细信息。

(4) 创建获取数据的对象 `da`。



- (5) 打开数据库连接, 获取查询数据。
- (6) 将获取的查询结果保存在 ds 中, 并返回 ds。

上述功能的对应实现代码如下所示:

```
public DataSet GetOrderItemByOrder(int orderID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行 SQL 语句
    string cmdText = "SELECT OrderItem.*, [Product].Name, [Product].Price FROM OrderItem
INNER JOIN [Product] ON [Product].ID = OrderItem.ProductID WHERE
OrderItem.OrderID=@OrderID";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@OrderID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = orderID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return ds;
}
```

## 6. 添加订单信息

添加订单信息即向系统库内添加新的订单信息。这一功能是由方法 AddOrder(string orderNo, int userID, int totalNumber, decimal totalMoney)实现的, 具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象, 实现数据库连接。
- (3) 调用存储过程 Pr\_AddOrder, 将此订单信息添加到系统库中。
- (4) 创建获取数据的对象 da。
- (5) 把 da 对象的执行方式设置为存储过程。
- (6) 打开数据库连接, 执行插入操作。

上述功能的对应实现代码如下所示:

```
public int AddOrder(string orderNo, int userID, int totalNumber, decimal totalMoney)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
```

```

SqlConnection con = new SqlConnection(connectionString); ///创建连接
string cmdText = "Pr_AddOrder"; ///设置被执行存储过程的名称
///创建 SqlCommand
SqlCommand cmd = new SqlCommand(cmdText, con);
///设置执行方式为存储过程
cmd.CommandType = CommandType.StoredProcedure;
///创建参数并赋值
cmd.Parameters.Add("@orderNo", SqlDbType.VarChar, 50);
cmd.Parameters.Add("@UserID", SqlDbType.Int, 4);
cmd.Parameters.Add("@totalNumber", SqlDbType.Int, 4);
cmd.Parameters.Add("@totalMoney", SqlDbType.Money);
cmd.Parameters[0].Value = orderNo;
cmd.Parameters[1].Value = userID;
cmd.Parameters[2].Value = totalNumber;
cmd.Parameters[3].Value = totalMoney;
cmd.Parameters.Add("@RETURN", SqlDbType.Int, 4);
cmd.Parameters[4].Direction = ParameterDirection.ReturnValue;
int result = -1;
try
{
    con.Open();
    ///操作数据
    result = cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    throw new Exception(ex.Message, ex);
}
finally
{
    con.Close();
}
return (int)cmd.Parameters[4].Value;
}

```

⚠ **注意：** 在上面插入操作过程中，通过存储过程 Pr\_AddOrder 定义了对应的 SQL 插入语句。存储过程 Pr\_AddOrder 的具体代码如下：

```

USE [shop]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Pr_AddOrder]
(
    @OrderNo varchar(50),
    @UserID int,
    @TotalNumber int,
    @TotalMoney money
)
AS
INSERT [Order]
    (OrderNo, UserID, TotalNumber, TotalMoney, CreateDate, Status)
VALUES
    (@OrderNo, @UserID, @TotalNumber, @TotalMoney, GETDATE(), 0)

RETURN @@Identity

```

## 7. 添加订单子项信息

添加订单子项信息即向系统库内添加某编号订单的子项信息。该功能是由方法 `AddOrderItem(int orderID, int productID, int number)` 实现的, 其具体实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数, 并将其保存在 `connectionString` 连接字符串中。
- (2) 使用连接字符串创建 `con` 对象, 实现数据库连接。
- (3) 新建 SQL 插入语句, 添加某编号订单的子项信息。
- (4) 创建获取数据的对象 `cmd`。
- (5) 打开数据库连接, 执行插入操作。
- (6) 将获取的查询结果保存在 `result` 中, 并返回 `result`。

上述功能的对应实现代码如下所示:

```
public int AddOrderItem(int orderID, int productID, int number)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "INSERT INTO OrderItem(OrderID, ProductID, Number)
        VALUES (@OrderID, @ProductID, @Number)";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@OrderID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@ProductID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@Number", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = orderID;
    cmd.Parameters[1].Value = productID;
    cmd.Parameters[2].Value = number;
    int result = -1;
    try
    {
        ///打开连接
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}
```

## 8. 更新订单状态

更新订单状态即更新系统库内某订单的状态。这一功能是由方法 `UpdateOrderStatus(int orderID, byte status)` 实现的, 具体实现流程如下。



- (1) 从系统配置文件 Web.config 内获取数据库连接参数,并将其保存在 connectionString 连接字符串中。
  - (2) 使用该连接字符串创建 con 对象,实现数据库连接。
  - (3) 新建 SQL 更新语句,更新系统库内某订单的状态。
  - (4) 创建获取数据的对象 cmd。
  - (5) 打开数据库连接,执行插入操作。
  - (6) 将获取的查询结果保存在 result 中,并返回 result。
- 上述功能的对应实现代码如下所示:

```
public int UpdateOrderStatus(int orderID, byte status)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行的 SQL 语句
    string cmdText = "UPDATE [Order] SET Status=@Status WHERE ID=@ID";
    SqlCommand cmd = new SqlCommand(cmdText, con); ///创建 SqlCommand
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@Status", SqlDbType.TinyInt, 1);
    cmd.Parameters[0].Value = orderID;
    cmd.Parameters[1].Value = status;
    int result = -1;
    try
    {
        con.Open(); ///打开连接
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}
```

### 7.7.3 商品评论

在 Product.c 文件中,与商品评论模块相关的方法如下:

- AddProductComment(string title, string body, string ip, string email, int productID)
- DeleteProductComment(int commentID)
- GetCommentByProdcut(int productID)

#### 1. 定义 Product 类

定义 Product 类的实现代码如下所示:

```

...
namespace ASPNETAJAXWeb.AjaxEBusiness
{
    public class Product
    {
        public Product()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }
    }
}
...

```

## 2. 获取评论信息

获取评论信息即获取系统内某商品评论的信息。

该功能是由方法 `GetCommentByProduct(int productID)` 实现的，具体实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。
- (2) 使用该连接字符串创建 `con` 对象，实现数据库连接。
- (3) 新建 SQL 查询语句，获取某编号商品的评论信息。
- (4) 创建获取数据的对象 `da`。
- (5) 打开数据库连接，获取查询数据。
- (6) 将获取的查询结果保存在 `ds` 中，并返回 `ds`。

上述功能的对应实现代码如下所示：

```

public DataSet GetCommentByProdcut(int productID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行 SQL 语句
    string cmdText = "SELECT * FROM ProductComment WHERE ProductID=@ProductID ORDER BY
CreateDate DESC";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@ProductID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = productID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close(); ///关闭连接
    }
}

```

```

    }
    return ds;
}

```

### 3. 添加评论信息

添加评论信息即向系统库添加新的评论信息。

该功能是由方法 `AddProductComment(string title, string body, string ip, string email, int productID)` 实现的，具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。

(2) 使用该连接字符串创建 `con` 对象，实现数据库连接。

(3) 新建 SQL 添加语句，向系统库内添加新的评论信息。

(4) 创建获取数据的对象 `cmd`。

(5) 打开数据库连接，进行添加操作。

(6) 将处理后的结果保存在 `result` 中，并返回 `result`。

上述功能的对应实现代码如下所示：

```

public int AddProductComment(string title, string body, string ip,
    string email, int productID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行的 SQL 语句
    string cmdText = "INSERT INTO ProductComment (Title, Body, IP, Email,
        ProductID, CreateDate) VALUES (@Title, @Body, @IP, @Email, @ProductID, GETDATE())";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@Title", SqlDbType.VarChar, 200);
    cmd.Parameters.Add("@Body", SqlDbType.VarChar, 1000);
    cmd.Parameters.Add("@IP", SqlDbType.VarChar, 50);
    cmd.Parameters.Add("@Email", SqlDbType.VarChar, 255);
    cmd.Parameters.Add("@ProductID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = title;
    cmd.Parameters[1].Value = body;
    cmd.Parameters[2].Value = ip;
    cmd.Parameters[3].Value = email;
    cmd.Parameters[4].Value = productID;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
}

```



```

    return result;
}

```

#### 4. 删除评论信息

删除评论信息即删除系统数据库内指定的评论信息。

该功能是由方法 DeleteProductComment(int commentID)实现的, 具体实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。

(2) 使用该连接字符串创建 con 对象, 实现数据库连接。

(3) 新建 SQL 删除语句, 删除系统库内指定编号的评论信息。

(4) 创建获取数据的对象 cmd。

(5) 打开数据库连接, 进行删除操作。

(6) 将处理后的结果保存在 result 中, 并返回 result。

上述功能的对应实现代码如下所示:

```

public int DeleteProductComment(int commentID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行的 SQL 语句
    string cmdText = "DELETE ProductComment WHERE ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = commentID;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}

```

### 7.7.4 商品分类

商品分类模块的数据库访问层功能是由 Category.cs 文件实现的, 其主要功能是在 ASPNETAJAXWeb.AjaxEBusiness 空间内建立 Category 类, 并定义多个方法实现对数据库中商品数据的处理。在 Category.cs 文件中, 与分类处理模块相关的方法如下:

- GetFenleis()

- GetSubFenlei(int categoryID)
- GetSingleFenlei(int categoryID)
- AddFenlei(string name, int parentID, string remark)
- UpdateFenlei(int categoryID, string name, string remark)
- UpdateFenleiOrder(int categoryID, string moveFlag)
- DeleteFenlei(int categoryID)

## 1. 定义 Category 类

在空间 ASPNETAJAXWeb.AjaxEBusiness 中定义 Category 类的实现代码，如下所示：

```
namespace ASPNETAJAXWeb.AjaxEBusiness
{
    public class Category
    {
        public Category()
        {
            ///
        }
    }
}
```

## 2. 获取分类信息

获取分类信息即获取系统库内存在的所有分类信息，上述功能是由方法 GetFenleis()实现的，其具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数，并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象，实现数据库连接。
- (3) 调用获取所有分类信息的存储过程 Pr\_GetFenleis，获取系统内的商品分类信息。
- (4) 创建获取数据的对象 da。
- (5) 把对象 da 的执行方式设置为存储过程。
- (6) 打开数据库连接，获取数据，将获取的数据保存在 ds 中。
- (7) 操作成功，返回 ds。

上述功能的对应实现代码如下所示：

```
public DataSet GetFenleis()
{
    ///获取连接字符串
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "Pr_GetFenleis";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///设置执行方式为存储过程
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open(); ///打开连接
    }
}
```

```

        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch(Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
        con.Close();
    }
    return ds;
}

```

### 3. 获取子类信息

获取子类信息即获取系统数据库指定编号分类的子类信息。

该功能是由方法 GetSubFenlei(int categoryID)实现的, 具体的实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。

(2) 使用该连接字符串创建 con 对象, 实现数据库连接。

(3) 调用获取某分类下子类信息的存储过程 Pr\_GetSubFenlei, 获取子类信息。

(4) 创建获取数据的对象 da。

(5) 把对象 da 的执行方式设置为存储过程。

(6) 打开数据库连接, 获取数据, 将获取的数据保存在 ds 中。

(7) 操作成功, 返回 ds。

上述功能的对应实现代码如下所示:

```

public DataSet GetSubFenlei(int categoryID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "Pr_GetSubFenlei";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///设置执行方式为存储过程
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@ParentID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = categoryID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch(Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
}

```



```
    }  
    finally  
    {  
        con.Close();  
    }  
    return ds;  
}
```

#### 4. 获取分类的详细信息

获取分类信息即获取系统内指定编号分类的详细信息。

该功能是由方法 `GetSingleFenlei(int categoryID)` 实现的，具体的实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。

(2) 使用该连接字符串创建 `con` 对象，实现数据库连接。

(3) 新建 SQL 查询语句，获取某 ID 分类的数据。

(4) 创建获取数据的对象 `cmd`。

(5) 打开数据库连接，获取查询数据。

(6) 将获取的查询结果保存在 `dr` 中，并返回 `dr`。

上述功能的对应实现代码如下所示：

```
public SqlDataReader GetSingleFenlei(int categoryID)  
{  
    SqlConnection con = new SqlConnection(connectionString);  
    ///创建 SQL 语句  
    string cmdText = "SELECT * FROM Category WHERE ID=@ID";  
    ///创建 SqlCommand  
    SqlCommand cmd = new SqlCommand(cmdText, con);  
    ///创建参数并赋值  
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);  
    cmd.Parameters[0].Value = categoryID;  
    ///定义 SqlDataReader  
    SqlDataReader dr;  
    try  
    {  
        con.Open();  
        ///读取数据  
        dr = cmd.ExecuteReader(CommandBehavior.CloseConnection);  
    }  
    catch (Exception ex)  
    {  
        throw new Exception(ex.Message, ex);  
    }  
    return dr;  
}
```

#### 5. 添加分类信息

添加分类信息即向系统库内添加新的分类信息，上述功能是由方法 `AddFenlei(string name, int parentID, string remark)` 实现的，其具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。

- (2) 使用该连接字符串创建 con 对象, 实现数据库连接。
- (3) 调用添加分类信息的存储过程 Pr\_AddFenlei, 进行添加操作。
- (4) 创建添加数据的对象 da。
- (5) 把对象 da 的执行方式设置为存储过程。
- (6) 打开数据库连接, 执行插入操作, 将处理结果保存在 result 中。
- (7) 操作成功返回 result。

上述功能的对应实现代码如下所示:

```
public int AddFenlei(string name, int parentID, string remark)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "Pr_AddFenlei";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///设置执行方式为存储过程
    cmd.CommandType = CommandType.StoredProcedure;
    ///创建参数并赋值
    cmd.Parameters.Add("@Name", SqlDbType.VarChar, 50);
    cmd.Parameters.Add("@ParentID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@Remark", SqlDbType.Text);
    cmd.Parameters[0].Value = name;
    cmd.Parameters[1].Value = parentID;
    cmd.Parameters[2].Value = remark;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}
```

## 6. 修改分类信息

修改分类信息即修改系统库内某编号的分类信息。

该功能是由方法 UpdateFenlei(int categoryID, string name, string remark)实现的, 具体的实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象, 实现数据库连接。
- (3) 调用修改类信息的存储过程 Pr\_UpdateFenlei, 进行修改操作。

- (4) 创建修改数据的对象 da。
- (5) 把对象 da 的执行方式设置为存储过程。
- (6) 打开数据库连接, 执行修改操作, 将处理结果保存在 result 中。
- (7) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```
public int UpdateFenlei(int categoryID, string name, string remark)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "Pr_UpdateFenlei";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///设置执行方式为存储过程
    cmd.CommandType = CommandType.StoredProcedure;
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@Name", SqlDbType.VarChar, 50);
    cmd.Parameters.Add("@Remark", SqlDbType.Text);
    cmd.Parameters[0].Value = categoryID;
    cmd.Parameters[1].Value = name;
    cmd.Parameters[2].Value = remark;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}
```

## 7. 修改分类次序

修改分类次序即修改系统数据库内某分类的排列顺序。

该功能是由方法 UpdateFenleiOrder(int categoryID, string moveFlag)实现的, 具体的实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象, 实现数据库连接。
- (3) 调用修改类顺序的存储过程 Pr\_UpdateFenleiOrder, 进行修改操作。
- (4) 创建修改数据的对象 da。
- (5) 把对象 da 的执行方式设置为存储过程。



(6) 打开数据库连接, 执行修改操作, 将处理结果保存在 result 中。

(7) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```
public int UpdateFenleiOrder(int categoryID, string moveFlag)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///设置被执行存储过程的名称
    string cmdText = "Pr_UpdateFenleiOrder";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///设置执行方式为存储过程
    cmd.CommandType = CommandType.StoredProcedure;
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@MoveFlag", SqlDbType.VarChar, 20);
    cmd.Parameters[0].Value = categoryID;
    cmd.Parameters[1].Value = moveFlag;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch(Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}
```

## 8. 删除分类信息

删除分类信息即删除系统数据库内某编号分类的信息。

该功能是由方法 DeleteFenlei(int categoryID)实现的, 具体的实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数, 将其保存在 connectionString 连接字符串中。

(2) 使用该连接字符串创建 con 对象, 实现数据库连接。

(3) 调用类信息的存储过程 Pr\_DeleteCategory, 进行删除操作。

(4) 创建删除数据的对象 da。

(5) 把对象 da 的执行方式设置为存储过程。

(6) 打开数据库连接, 执行删除操作, 将处理结果保存在 result 中。

(7) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```
public int DeleteFenlei(int categoryID)
{

```

```

string connectionString =
    ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
///设置被执行存储过程的名称
string cmdText = "Pr_DeleteCategory";
///创建 SqlCommand
SqlCommand cmd = new SqlCommand(cmdText, con);
///设置执行方式为存储过程
cmd.CommandType = CommandType.StoredProcedure;
///创建参数并赋值
cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
cmd.Parameters[0].Value = categoryID;
int result = -1;
try
{
    con.Open();
    ///操作数据
    result = cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    throw new Exception(ex.Message, ex);
}
finally
{
    con.Close();
}
return result;
}

```

## 9. 存储过程说明

在上述数据库访问层的操作过程中，涉及 6 个数据库存储过程。各存储过程的具体说明如下。

### (1) Pr\_GetFenleis

存储过程 Pr\_GetFenleis 的功能是获取系统库内的所有商品分类信息，主要代码如下：

```

SELECT
    A.ID,
    A.Name,
    A.ParentID,
    A.ShowOrder,
    A.Remark,
    ISNULL((SELECT Name FROM Category AS B WHERE A.ParentID = B.ID),null) AS ParentName,
    ISNULL((SELECT COUNT(*) FROM Category AS C WHERE A.ID = c.ParentID),0) AS SubCount,
    ISNULL((SELECT COUNT(*) FROM Category AS D WHERE A.ParentID = D.ParentID),0) AS
    SiblingCount
FROM
    Category AS A
ORDER BY
    ParentID, ShowOrder

```

### (2) Pr\_GetSubFenlei

存储过程 Pr\_GetSubFenlei 的功能是某分类下子分类的信息，主要代码如下：

```

SELECT
    A.ID,
    A.Name,

```

```

A.ParentID,
A.ShowOrder,
A.Remark,
ISNULL((SELECT Name FROM Category AS B WHERE A.ParentID = B.ID),null) AS ParentName,
ISNULL((SELECT COUNT(*) FROM Category AS C WHERE A.ID = c.ParentID),0) AS SubCount,
ISNULL((SELECT COUNT(*) FROM Category AS D WHERE A.ParentID = D.ParentID),0) AS
SiblingCount
FROM
    Category AS A
WHERE
    ParentID = @ParentID
ORDER BY
    ParentID,ShowOrder

```

### (3) Pr\_AddFenlei

存储过程 Pr\_AddFenlei 的功能是向系统库内添加某分类的信息，主要代码如下：

```

AS
DECLARE @Count int
SET
    @Count = ISNULL((SELECT COUNT(*) FROM Category WHERE ParentID = @ParentID),0)
INSERT INTO
    Category
    (Name,ParentID,ShowOrder,Remark)
VALUES
    (@Name,@ParentID,@Count + 1,@Remark)
RETURN @@Identity

```

### (4) Pr\_UpdateFenlei

存储过程 Pr\_UpdateFenlei 的功能是修改系统库内某分类的信息，主要代码如下：

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Pr_UpdateFenlei]
(
    @ID int,
    @Name varchar(200),
    @Remark text
)
AS
DECLARE @ParentID int
SET
    @ParentID = ISNULL((SELECT ParentID FROM Category WHERE ID = @ID),-2)
DECLARE @Count int
SET
    @Count = ISNULL((SELECT COUNT(*) FROM Category WHERE ParentID = @ParentID AND Name
= @Name AND ID <> @ID),-1)
IF @Count <= 0
BEGIN
    UPDATE
        Category
    SET
        Name = @Name,
        Remark = @Remark
    WHERE
        ID = @ID
    RETURN @Count
END

```



```
ELSE
    RETURN @Count
```

### (5) Pr\_UpdateFenleiOrder

存储过程 Pr\_UpdateFenleiOrder 的功能是修改系统库内某分类的顺序，主要代码如下：

```
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Pr_UpdateFenleiOrder]
(
    @ID int,
    @MoveFlag varchar(20)
)
AS
DECLARE @UpdateOrder int
DECLARE @ParentID int
SET @UpdateOrder =(SELECT ShowOrder FROM Category WHERE ID = @ID)
SET @ParentID =(SELECT ParentID FROM Category WHERE ID = @ID)
BEGIN TRAN
IF @MoveFlag = 'up'
BEGIN
    DECLARE @LitterID int
    SET @LitterID =(
        SELECT ID FROM Category
        WHERE ParentID = @ParentID AND ShowOrder = @UpdateOrder -1
    )
    UPDATE
        Category
    SET
        ShowOrder = @UpdateOrder -1
    WHERE
        ID = @ID
    UPDATE
        Category
    SET
        ShowOrder = @UpdateOrder
    WHERE
        ID = @LitterID
END
ELSE
    IF @MoveFlag = 'down'
    BEGIN
        DECLARE @GreaterID int
        SET @GreaterID =(
            SELECT ID FROM Category
            WHERE ParentID = @ParentID AND ShowOrder = @UpdateOrder + 1
        )
        UPDATE
            Category
        SET
            ShowOrder = @UpdateOrder + 1
        WHERE
            ID = @ID
        UPDATE
            Category
        SET
            ShowOrder = @UpdateOrder
        WHERE
            ID = @GreaterID
    END
COMMIT TRAN
```

## (6) Pr\_DeleteCategory

存储过程 Pr\_DeleteCategory 的功能是删除系统库内某编号分类的信息, 主要代码如下:

```

DECLARE @ShowOrder int
SET @ShowOrder = ISNULL((SELECT ShowOrder FROM Category WHERE ID = @ID), 0)
DECLARE @ParentID int
SET @ParentID = (ISNULL((SELECT @ParentID FROM Category WHERE ID = @ID), -1))
DECLARE @Count int
BEGIN TRAN
    UPDATE
        Category
    SET
        ShowOrder = ShowOrder - 1
    WHERE
        Showorder > @ShowOrder AND ParentID = @ParentID
    DELETE
        Category
    WHERE
        ID = @ID
    SET
        @Count = @@ROWCOUNT
COMMIT TRAN
RETURN @Count

```

### 7.7.5 商品管理

在数据访问层文件 Product.cs 中, 与商品管理模块相关的各方法的具体说明如下。

- AddProduct(string name, int categoryID, int userID, decimal price, int stock, string remark): 功能是添加商品到库。
- UpdateProduct(int productID, string name, string remark): 功能是更新某编号的商品信息。
- UpdateProductPicture(int productID, string pictureUrl): 功能是更新某编号商品的图片信息。
- DeleteProduct(int productID): 功能是删除某编号商品的信息。
- GetAttributeByFenlei(int categoryID): 功能是根据分类获取商品的属性。
- GetAttributeByProduct(int productID): 功能是根据商品 ID 获取其属性。
- AddAttributeValue(int productID, int attributeID, string value): 功能是添加商品的属性值。

#### 1. 定义 Product 类

在空间 ASPNETAJAXWeb.AjaxEBusiness 内定义 Product 类的实现代码如下所示:

```

using System.Web.UI.WebControls;
namespace ASPNETAJAXWeb.AjaxEBusiness
{
    public class Product
    {
        public Product()
        {
        }
    }
}

```

## 2. 添加商品信息

添加商品信息即向系统库内添加新的商品信息，上述功能是由方法 `AddProduct(string name, int categoryID, int userID, decimal price, int stock, string remark)` 实现的，具体的实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。
- (2) 使用该连接字符串创建 `con` 对象，实现数据库的连接。
- (3) 调用添加商品信息的存储过程 `Pr_AddProduct`，向系统库内添加新的商品信息。
- (4) 创建添加数据的对象 `cmd`。
- (5) 把对象 `cmd` 的执行方式设置为存储过程。
- (6) 打开数据库，执行添加操作，将处理结果保存在 `result` 中。
- (7) 操作成功，返回 `result`。

上述功能的对应实现代码如下所示：

```
public int AddProduct(string name, int categoryID, int userID, decimal price,
    int stock, string remark)
{
    ///获取连接字符串
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString); ///创建连接
    ///设置被执行存储过程的名称
    string cmdText = "Pr_AddProduct";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///设置执行方式为存储过程
    cmd.CommandType = CommandType.StoredProcedure;
    ///创建参数并赋值
    cmd.Parameters.Add("@Name", SqlDbType.VarChar, 50);
    cmd.Parameters.Add("@CategoryID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@UserID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@Price", SqlDbType.Money, 8);
    cmd.Parameters.Add("@Stock", SqlDbType.Int, 4);
    cmd.Parameters.Add("@Remark", SqlDbType.Text);
    cmd.Parameters[0].Value = name;
    cmd.Parameters[1].Value = categoryID;
    cmd.Parameters[2].Value = userID;
    cmd.Parameters[3].Value = price;
    cmd.Parameters[4].Value = stock;
    cmd.Parameters[5].Value = remark;
    cmd.Parameters.Add("@RETURN", SqlDbType.Int, 4);
    cmd.Parameters[6].Direction = ParameterDirection.ReturnValue;
    int result = -1;
    try
    {
        con.Open(); ///打开连接
        result = cmd.ExecuteNonQuery(); ///操作数据库
    }
    catch (Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
}
```



```

finally
{
    ///关闭连接
    con.Close();
}
return (int)cmd.Parameters[6].Value;
}

```

在上述操作处理的执行过程中,通过调用存储过程 Pr\_AddProduct,实现了 SQL 语句的数据添加操作。存储过程 Pr\_AddProduct 的主要实现代码如下:

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Pr_AddProduct]
(
    @Name varchar(200),
    @CategoryID int,
    @UserID int,
    @Price money,
    @Stock int,
    @Remark text
)
AS
INSERT INTO
    [Product]
    (Name,CategoryID,UserID,Price,Stock,SaleNumber,CreateDate,LasterDate,
    Status,Remark)
VALUES
    (@Name,@CategoryID,@UserID,@Price,@Stock,0,GETDATE(),GETDATE(),0,@Remark)
RETURN @@Identity

```

由于本书篇幅有限,在此只对商品添加的过程进行了代码讲解,至于其他的方法和存储过程的具体实现,读者参阅本书所附光盘中的对应文件即可。

在上述编码过程中,不但实现了基本的数据处理功能,而且通过存储过程提高了整个系统的存储效率。存储过程(Stored Procedure)是一组为了完成特定功能的 SQL 语句集,经编译后存储在数据库中。用户通过指定存储过程的名字并给出参数(如果该存储过程带有参数)来执行它。存储过程是数据库中的一个重要对象,任何一个设计良好的数据库应用程序都应该用到存储过程。

存储过程是利用 SQL Server 所提供的 Transact-SQL 语言所编写的程序。Transact-SQL 语言是 SQL Server 提供的专为设计数据库应用程序的语言,它是应用程序与 SQL Server 数据库间的主要程序设计接口。它好比 Oracle 数据库系统中的 PL-SQL 和 Informix 的数据库系统结构中的 Informix-4GL 语言。

## 7.8 显示商品

视频讲解  光盘: 视频\第7章\显示商品.avi

在本节的内容中,开始讲解显示商品模块的具体实现过程。为了便于用户浏览商品,将整个首页通过框架分隔,划分为 4 个部分:主框架页、顶部导航页面、左侧导航页、右侧

信息详情页。

## 7.8.1 主框架页

本系统实例的系统主页是一个框架页面，其功能是调用各框架子页以显示指定的信息。该功能的实现文件如下：

- Default.aspx.cs。
- Default.aspx。

主页处理页面文件 Default.aspx.cs 的功能是，引入命名空间并声明 Default 类，实现主框架页面的初始化处理。具体的实现代码如下所示：

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}
```

## 7.8.2 顶部导航页面

本系统实例的顶部导航页面是一个动态页面，其功能是根据用户的状态来显示对应的导航信息。该功能的实现文件如下：

- Daohang.aspx。
- Daohang.aspx.cs。

导航处理页面文件 Daohang.aspx.cs 的功能，是引入命名空间并声明 Toolbar 类，实现登录表单的判断处理。具体的实现代码如下所示：

```
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class Toolbar : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        ///动态载入用户登录控件或者显示用户信息的控件
        pLogin.Controls.Clear();
        if(Session["UserID"] == null)
        {
            pLogin.Controls.Add(Page.LoadControl("~/Login/LoginUC.ascx"));
        }
        else
        {
            pLogin.Controls.Add(Page.LoadControl("~/Login/LogoffUC.ascx"));
        }
    }
}
```

```

    }
}

```

### 7.8.3 左侧导航——分类列表页面

本系统实例的左侧类别列表页面是一个动态页面，其功能是将系统内所有商品分类的信息显示出来。该功能的实现文件如下：

- Classes.aspx。
- Classes.aspx.cs。

类别列表处理页面文件 Classes.aspx.cs 的功能，是引入命名空间并声明 CategoryPage 类，将商品类别数据调用并显示出来。具体的实现代码如下所示：

```

using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class CategoryPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    private void BindPageData()
    {
        Category category = new Category();
        category.InitCatalogTreeView(tvCategory);
    }
}

```

### 7.8.4 右侧导航——商品列表页面

本系统实例的右侧商品列表页面是一个动态页面，其功能是将系统内的商品信息以列表样式显示出来。该功能的实现文件如下：

- Product.aspx。
- Product.aspx.cs。

商品列表处理页面文件 Product.aspx.cs 的功能，是引入命名空间并声明 ProductPage 类，从地址栏中获取某商品类别的值。具体的实现代码如下所示：

```

using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class ProductPage : System.Web.UI.Page
{
    protected int categoryID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取商品种类的 ID
        if (Request.Params["CategoryID"] != null)

```



```

    {
        categoryID = Int32.Parse(Request.Params["CategoryID"].ToString());
    }
}

```

## 7.8.5 按被点击次数显示

按被点击次数显示就是把系统内的商品信息按照被点击次数的高低来排序显示。实现文件如下：

- Click.aspx。
- Click.aspx.cs。

点击次数处理页面文件 Click.aspx.cs 的功能，是在初始化处理中按点击次数显示商品页面，获取并显示对应的商品信息。具体的实现流程如下。

- (1) 引入命名空间，并声明 ViewProductByCategoryCount 类。
- (2) 定义 Page\_Load 初始化处理，并从地址栏中获取 CategoryID 变量值。
- (3) 获取并按点击次数排序，显示商品数据。
- (4) 定义购物车处理事件。

Click.aspx.cs 文件的具体实现代码如下所示：

```

using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class ViewProductByCategoryCount : System.Web.UI.Page
{
    private int categoryID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取商品种类的 ID
        if (Request.Params["CategoryID"] != null)
        {
            categoryID = Int32.Parse(Request.Params["CategoryID"].ToString());
        }
        ///显示商品数据
        if (!Page.IsPostBack && categoryID > 0)
        {
            ///绑定数据初始化
            gvProduct.DataSource = null;
            gvProduct.DataBind();
            BindPageData(categoryID);
        }
    }
    private void BindPageData(int categoryID)
    {
        Product product = new Product();
        DataSet ds = product.GetProductByFenlei(categoryID);
        if (ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        ///设置按访问次数排序
        DataView dv = ds.Tables[0].DefaultView;
        dv.Sort = "ViewCount DESC";
        gvProduct.DataSource = dv;
        gvProduct.DataBind();
    }
}

```

```

}
protected void gvProduct_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if(e.CommandName == "buy")
    {
        ShoppingCartItem item = new ShoppingCartItem();
        int rowIndex = Int32.Parse(e.CommandArgument.ToString());
        if(rowIndex <= -1 || rowIndex >= gvProduct.Rows.Count) return;
        ///获取商品 ID 和数量
        item.ProductID =
            Int32.Parse(gvProduct.DataKeys[rowIndex]["ID"].ToString());
        item.Number = 1;
        ///获取商品名称
        Label lbName = (Label)gvProduct.Rows[rowIndex].FindControl("lbName");
        if(lbName != null)
        {
            item.Name = lbName.Text;
        }
        ///获取商品价格
        Label lbPrice = (Label)gvProduct.Rows[rowIndex].FindControl("lbPrice");
        if(lbPrice != null)
        {
            item.Price = decimal.Parse(lbPrice.Text);
        }
        ShoppingCart shoppingCart = new ShoppingCart(Session);
        if(shoppingCart.AddProductToShoppingCart(item) > -1)
        {
            AjaxEBusinessSystem.ShowAjaxDialog(
                (Button)e.CommandSource, "恭喜您, 添加商品到购物车成功。");
        }
    }
}

protected void gvProduct_RowDataBound(object sender, GridViewRowEventArgs e)
{
    Button btnBuy = (Button)e.Row.FindControl("btnBuy");
    if(btnBuy != null)
    {
        ///设置 CommandArgument 属性的值为当前行的索引
        btnBuy.CommandArgument = e.Row.RowIndex.ToString();
    }
}
}

```

在 ASP.NET 项目中, **DataView** 是一个十分重要的数据源。在 **DataView** 中, 实现了对 **DataTable** 中数据的过滤和排序操作。当从数据库中选择数据时, 用户可以单击列标题, 并对数据进行排序。并且可以只过滤要显示在某些行中的数据, 例如用户修改过的所有数据。**DataView** 允许限制要显示给用户的数据行, 但不允许限制 **DataTable** 中的数据列。


例如, 根据现有的 **DataTable** 创建 **DataView** 的代码如下:

```
DataView dv = new DataView(dataTable);
```

创建好后, 就可以改变 **DataView** 上的设置, 当该视图显示在 **DataGrid** 中时, 这些设置会影响要显示的数据, 以及允许对这些数据进行的操作。具体说明如下:

- **AllowEdit = false**, 表示在数据行上禁用所有列的编辑功能。
- **AllowNew = false**, 表示禁用新行功能。
- **AllowDelete = false**, 表示禁用删除行的功能。

- 设置 RowStateFilter 只显示指定状态的行。
- 设置 RowFilter 可过滤数据行。

 **注意：** DataView 不允许修改要显示的数据列，只允许修改要显示的数据行。

## 7.8.6 按商品名称显示模块

按商品名称显示模块的功能，是将系统内的商品信息按商品名称的排序方式来显示。该功能的实现文件如下：

- Name.aspx。
- Name.aspx.cs。

按名称显示处理页面文件 Name.aspx.cs 的功能，是在初始化处理中按名称显示商品页面，获取并显示对应的商品信息。具体的实现流程如下。

- (1) 引入命名空间，并声明 ViewProductByCategoryName 类。
- (2) 定义 Page\_Load 初始化处理，并从地址栏中获取 CategoryID 变量值。
- (3) 获取并按点击次数排序，显示商品数据。
- (4) 定义购物车处理事件。

Name.aspx.cs 文件的具体实现代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class ViewProductByCategoryName : System.Web.UI.Page
{
    private int categoryID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取商品种类的 ID
        if (Request.Params["CategoryID"] != null)
        {
            categoryID = Int32.Parse(Request.Params["CategoryID"].ToString());
        }
        ///显示商品数据
        if (!Page.IsPostBack && categoryID > 0)
        {
            ///绑定数据初始化
            gvProduct.DataSource = null;
            gvProduct.DataBind();
            BindPageData(categoryID);
        }
    }
    private void BindPageData(int categoryID)
    {
        Product product = new Product();
        DataSet ds = product.GetProductByFenlei(categoryID);
        if (ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        ///设置按名称排序
        DataView dv = ds.Tables[0].DefaultView;
        dv.Sort = "Name";
        gvProduct.DataSource = dv;
        gvProduct.DataBind();
    }
    protected void gvProduct_RowCommand(object sender, GridViewCommandEventArgs e)
    {

```



```

if(e.CommandName == "buy")
{
    ShoppingCartItem item = new ShoppingCartItem();
    int rowIndex = Int32.Parse(e.CommandArgument.ToString());
    if(rowIndex <= -1 || rowIndex >= gvProduct.Rows.Count) return;
    ///获取商品 ID 和数量
    item.ProductID =
        Int32.Parse(gvProduct.DataKeys[rowIndex]["ID"].ToString());
    item.Number = 1;
    ///获取商品名称
    Label lbName = (Label)gvProduct.Rows[rowIndex].FindControl("lbName");
    if(lbName != null)
    {
        item.Name = lbName.Text;
    }
    ///获取商品价格
    Label lbPrice = (Label)gvProduct.Rows[rowIndex].FindControl("lbPrice");
    if(lbPrice != null)
    {
        item.Price = decimal.Parse(lbPrice.Text);
    }
    ShoppingCart shoppingCart = new ShoppingCart(Session);
    if(shoppingCart.AddProductToShoppingCart(item) > -1)
    {
        AjaxEBusinessSystem.ShowAjaxDialog(
            (Button)e.CommandSource, "恭喜您, 添加商品到购物车成功。");
    }
}
}

protected void gvProduct_RowDataBound(object sender, GridViewRowEventArgs e)
{
    Button btnBuy = (Button)e.Row.FindControl("btnBuy");
    if(btnBuy != null)
    {
        ///设置 CommandArgument 属性的值为当前行的索引
        btnBuy.CommandArgument = e.Row.RowIndex.ToString();
    }
}
}

```

### 7.8.7 商品详情页面

商品详情显示模块的功能, 是将系统内某编号的商品信息详细地显示出来。对应的实现文件如下:

- ShowProduct.aspx。
- ShowProduct.aspx.cs。

商品详情处理页面文件 ShowProduct.aspx.cs 的功能, 是初始化处理商品详情显示页面, 获取并显示此编号商品的详细信息。具体的实现流程如下。

- (1) 引入命名空间, 并声明 ShowProduct 类。
- (2) 定义 Page\_Load 初始化处理。
- (3) 获取商品编号和 IP 地址。
- (4) 通过 BindPageData(int productID) 获取并显示此编号的商品信息。
- (5) 显示商品的评论。

- (6) 设置 btnBuy\_Click(object sender, EventArgs e)事件, 实现购物车处理。
- (7) 设置 Commit\_Click(object sender, EventArgs e)事件, 及时载入评论发布数据。
- 文件 ShowProduct.aspx.cs 的具体实现代码如下所示:

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
using System.Data.SqlClient;
public partial class ShowProduct : System.Web.UI.Page
{
    protected string ProductName = string.Empty;
    protected string ViewCount = string.Empty;
    protected string CreateDate = string.Empty;
    protected string Username = string.Empty;
    protected string Remark = string.Empty;
    private int productID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取客户端的 IP 地址
        tbIP.Text = Request.UserHostAddress;
        if (Request.Params["ProductID"] != null)
        {
            productID = Int32.Parse(Request.Params["ProductID"].ToString());
        }
        if (productID > 0)
        {
            BindPageData(productID);
            if (!Page.IsPostBack)
            {
                ///更新商品被访问的次数
                Product product = new Product();
                product.UpdateProductViewCount(productID);
            }
        }
    }
    private void BindPageData(int productID)
    {
        ///获取商品评论
        Product product = new Product();
        SqlDataReader dr = product.GetSingleProduct(productID);
        if (dr == null) return;
        if (dr.Read())
        {
            ///读取商品信息
            ProductName = dr["Name"].ToString();
            ViewCount = dr["ViewCount"].ToString();
            CreateDate = dr["CreateDate"].ToString();
            Username = dr["Username"].ToString();
            Remark = dr["Remark"].ToString();
            lbPrice.Text = string.Format("{0:f2}", dr["Price"]);
            imgProduct.ImageUrl = "~/ " + dr["PictureUrl"].ToString();
            ///获取属性的数据
            DataSet ds = product.GetAttributeByProduct(productID);
            ///绑定并显示属性的数据
            gvAttribute.DataSource = ds;
            gvAttribute.DataBind();
        }
        dr.Close();
        ///显示商品的评论
        gvComment.DataSource = product.GetCommentByProduct(productID);
    }
}

```

```

        gvComment.DataBind();
    }
    protected void btnBuy_Click(object sender, EventArgs e)
    {
        ///设置商品的属性
        ShoppingCartItem item = new ShoppingCartItem();
        item.ProductID = productID;
        item.Name = ProductName;
        item.Price = Decimal.Parse(lbPrice.Text);
        item.Number = 1;
        ///添加到购物车
        ShoppingCart shoppingCart = new ShoppingCart(Session);
        if(shoppingCart.AddProductToShoppingCart(item) > -1)
        {
            AjaxEBusinessSystem.ShowAjaxDialog(
                (Button)sender, "恭喜您, 添加商品到购物车成功。");
        }
    }
    protected void btnCommit_Click(object sender, EventArgs e)
    {
        Product product = new Product();
        ///发表评论
        if(product.AddProductComment(
            tbTitle.Text, tbBody.Text, Request.UserHostAddress, tbEmail.Text, productID) > 0)
        {
            ///重新显示数据
            BindPageData(productID);
        }
    }
}

```

通过上述编码可知, 所谓的动态站点, 就是依靠数据库这个中间存储媒介实现的。在这个管理模块中, 整个实现过程都与数据库息息相关: ①查询系统数据库, 将库内的商品源信息显示在页面上; ②在后面分别添加、删除和修改链接; ③单击删除链接后, 会将数据库内对应的商品删除, 对应的页面中再也不能显示此商品的信息了。

## 7.9 商品分类处理

视频讲解  光盘: 视频\第7章\商品分类处理.avi

商品分类即商品的种类, 很多读者可能会对此功能提出疑问: 本例中, 客户方只卖花这一种商品, 所以分类功能好像没有什么意义啊。

其实, 这里使用分类功能有如下所示的两个好处。

- (1) 用户可以对花进行细分, 例如从种类细分和从用途细分。
- (2) 可以兼卖其他的商品, 这样就能构建一个综合型商城。

商品分类处理模块的功能是, 对系统库内的商品类别进行处理操作。本模块功能的实现文件如下:

- AddClass.aspx。
- AddClass.aspx.cs。
- Classes.aspx。



- Classes.aspx.cs。
- UpdateClasses.aspx。
- UpdateClasses.aspx.cs。

## 7.9.1 设置分类的层次结构

通常，在 ASP.NET + Ajax 开发中，使用两种分类层次结构，具体说明如下：

- 使用 TreeView 控件来显示分类层次结构。
- 使用 DropDownList 控件来显示分类层次结构。

### 1. TreeView 控件的实现

在实例中，将分别通过方法 InitCatalogTreeView 和 CreateChildNode 实现分类层次结构。其中，InitCatalogTreeView 方法的功能，是使用 TreeView 控件来显示分类层次结构。具体的实现流程如下。

- (1) 调用 GetFenleis()方法获取所有商品类别信息，使用 ds 对象保存结果。
- (2) 清空控件 dv 内的节点，获取根节点的数据，并且创建根节点 root，以及设置其对应的属性。
- (3) 将根节点添加到控件 dv 内。
- (4) 调用 CreateChildNode 方法，使用递归方法创建根节点对应下的子节点。

Category.cs 文件内，InitCatalogTreeView 方法的具体实现代码如下所示：

```
public void InitCatalogTreeView(TreeView tv)
{
    DataSet ds = GetFenleis();
    if(ds == null) return;
    if(ds.Tables.Count <= 0) return;
    DataTable dt = ds.Tables[0];
    tv.Nodes.Clear();    ///清空树的所有节点
    DataRow[] rowList = dt.Select("ParentID='0'");
    if(rowList.Length < 1) return;
    TreeNode root = new TreeNode();
    root.Text = rowList[0]["Name"].ToString();
    ///设置根节点的 value 值
    root.Value = rowList[0]["ID"].ToString();
    root.Target = "Product";
    root.NavigateUrl = "~/Product.aspx?CategoryID=" + root.Value;
    root.Expanded = true;
    ///添加根节点
    tv.Nodes.Add(root);
    ///创建其他节点
    CreateChildNode(root, dt, "Product", "~/Product.aspx?CategoryID=");
}
```

方法 CreateChildNode 的功能，是创建父节点 parentNode 内的所有子节点，参数的具体说明如下。

- parentNode: 指定父节点。
- dt: 指定数据源。
- target: 指定 Target 的属性值。
- url: 指定节点 NavigateUrl 的值。

方法 CreateChildNode 的具体实现流程如下。

- (1) 从数据源 dt 获取父节点内的子节点数据, 并按照 ShowOrder 字段排序。
- (2) 将获取的字段数据保存在变量 rowList 中, 并使用 foreach 逐一读取里面的数据。
- (3) 为每个 rowList 数据创建一个节点, 并设置各节点的属性。
- (4) 调用 CreateChildNode 方法, 使用递归方法创建当前节点对应下的子节点。

Category.cs 文件内, CreateChildNode 方法的具体实现代码如下所示:

```
private void CreateChildNode(TreeNode parentNode, DataTable dt, string target, string url)
{
    ///选择数据时, 添加了排序表达式 OrderBy
    DataRow[] rowList = dt.Select("ParentID='" + parentNode.Value + "'", "ShowOrder");
    foreach(DataRow row in rowList)
    {
        ///创建新节点
        TreeNode node = new TreeNode();
        ///设置节点的属性
        node.Text = row["Name"].ToString();
        node.Value = row["ID"].ToString();
        node.Target = target;
        node.NavigateUrl = url + node.Value;
        node.Expanded = true;
        parentNode.ChildNodes.Add(node);
        ///递归调用, 创建其他节点
        CreateChildNode(node, dt, target, url);
        if(node.ChildNodes.Count > 0)
        {
            node.SelectAction = TreeNodeSelectAction.None;
        }
    }
}
```

## 2. DropDownList 控件的实现

在实例中, 将分别通过方法 InitFenleiList 和 CreateSubNode 来实现分类层次结果。其中, InitFenleiList 方法的功能, 是使用 DropDownList 控件来显示分类层次结构。

Category.cs 文件内 InitFenleiList 方法的具体实现代码如下:

```
public void InitFenleiList(ListControl list)
{
    DataSet ds = GetFenleis();
    if(ds == null) return;
    if(ds.Tables.Count <= 0) return;
    DataTable dt = ds.Tables[0];
    list.Items.Clear();    ///清空树的所有节点
    DataRow[] rowList = dt.Select("ParentID='0'", "ShowOrder");
    if(rowList.Length < 1) return;
    string name = string.Empty;
    string value = string.Empty;
    foreach(DataRow row in rowList)
    {
        name = "|--" + row["Name"].ToString();
        value = row["ID"].ToString();
        list.Items.Add(new ListItem(name, value));
        CreateSubNode(list, dt, row["ID"].ToString(), name);
    }
}
```

CreateSubNode 方法的功能，是创建 list 节点的所有数据项，从而实现子项的显示。Category.cs 文件内 CreateSubNode 方法的具体实现代码如下：

```
private void CreateSubNode(ListControl list, DataTable dt, string parentValue,
    string parentName)
{
    ///选择数据时，添加了排序表达式 OrderBy
    DataRow[] rowList = dt.Select("ParentID='" + parentValue + "'", "ShowOrder");
    string name = string.Empty;
    string value = string.Empty;
    foreach(DataRow row in rowList)
    {
        name = parentName + " |--" + row["Name"].ToString();
        value = row["ID"].ToString();
        list.Items.Add(new ListItem(name, value));
        CreateSubNode(list, dt, row["ID"].ToString(), name);
    }
}
```

## 7.9.2 添加分类模块

添加分类模块的功能，是向系统库内添加新的商品类别。上述功能的实现文件如下。

- 文件 AddClass.aspx：分类添加表单页面。
- 文件 AddClass.aspx.cs：分类添加处理文件。

分类添加处理文件 AddClass.aspx.cs 的功能，是初始化添加表单界面，并将表单内的数据添加到系统库中。其具体实现流程如下。

- (1) 引入命名空间，定义 Category\_AddCategory 类。
- (2) 声明 Page\_Load 方法，进行页面初始化处理。
- (3) 调用 BindPageData()方法，载入显示分类树数据。
- (4) 调用 Commit\_Click(object sender, EventArgs e)方法，向库中添加新的分类数据。

文件 AddClass.aspx.cs 的具体实现代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class Category_AddCategory : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if(!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    private void BindPageData()
    {
        ///获取数据
        Category category = new Category();
        ///显示分类的层次结构
        category.InitFenleiList(ddlCategory);
        ///设置第一项为选择项，如果存在
        if(ddlCategory.Items.Count > 0)
        {
            ddlCategory.SelectedIndex = 0;
        }
    }
}
```



```

}
protected void btnCommit_Click(object sender, EventArgs e)
{
    ///添加新的分类
    Category category = new Category();
    if(category.AddFenlei(tbName.Text, Int32.Parse(ddlCategory.SelectedValue),
        tbRemark.Text) > 0)
    {
        ///重定向到管理页面
        Response.Redirect("~/Category/Classes.aspx");
    }
}
}

```

### 7.9.3 分类修改模块

分类修改模块的功能，是对系统库内的某商品分类信息进行修改。上述功能的实现文件如下。

- UpdateClasses.aspx: 分类修改表单页面。
- UpdateClasses.aspx.cs: 分类修改处理文件。

分类修改处理文件 UpdateClasses.aspx.cs 的功能，是初始化修改表单界面，并将表单内的数据更新到系统库中。具体的实现流程如下。

- (1) 引入命名空间，定义 Category\_UpdateCategory 类。
- (2) 声明 Page\_Load 方法，进行页面初始化处理。
- (3) 获取修改类别的 ID。
- (4) 获取显示数据。
- (5) 调用 BindPageData(int categoryID)方法，显示分类树的数据。
- (6) 调用 btnCommit\_Click(object sender, EventArgs e)方法，对数据库中此编号分类的数据进行更新。

文件 UpdateClasses.aspx.cs 的具体实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
using System.Data.SqlClient;
public partial class Category_UpdateCategory : System.Web.UI.Page
{
    int categoryID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取被修改数据的 ID
        if(Request.Params["CategoryID"] != null)
        {
            categoryID = Int32.Parse(Request.Params["CategoryID"].ToString());
        }
        ///显示被修改的数据
        if(!Page.IsPostBack && categoryID > 0)
        {
            BindPageData(categoryID);
        }
        ///设置按钮是否可用
        btnCommit.Enabled = categoryID>0? true : false;
    }
}

```

```

private void BindPageData(int categoryID)
{
    ///读取数据
    Category category = new Category();
    ///显示分类的层次结构
    category.InitFenleiList(ddlCategory);
    SqlDataReader dr = category.GetSingleFenlei(categoryID);
    if(dr == null) return;
    if(dr.Read())
    {
        ///显示数据
        tbName.Text = dr["Name"].ToString();
        tbRemark.Text = dr["Remark"].ToString();
        AjaxEBusinessSystem.ListSelectedItemByValue(
            ddlCategory, dr["ParentID"].ToString());
    }
    dr.Close();
}
protected void btnCommit_Click(object sender, EventArgs e)
{
    Category category = new Category();
    ///修改分类的属性
    if (category.UpdateFenlei(categoryID, tbName.Text, tbRemark.Text) > 0)
    {
        ///重定向到管理页面
        Response.Redirect("~/Category/Classes.aspx");
    }
}
}

```

## 7.9.4 分类管理模块

分类管理模块的功能，是将系统库内的商品分类信息以列表样式显示出来。并提供对每种分类的操作链接。上述功能的实现文件如下。

- Classes.aspx: 分类管理列表显示页面。
- Classes.aspx.cs: 分类管理列表处理文件。

分类管理列表处理文件 Classes.aspx.cs 的功能，是初始化修改表单界面，并将表单内的数据更新到系统库中。其具体实现流程如下。

- (1) 引入命名空间，定义 Category\_Category 类。
- (2) 声明 Page\_Load 方法，进行页面初始化处理。
- (3) 调用 BindPageData(int categoryID)方法，获取并显示分类树的数据。
- (4) 定义 gvCategory\_RowCommand 方法，根据用户的操作处理重定向到对应的页面。
- (5) 弹出“删除确认”对话框，将选定信息删除。

文件 Classes.aspx.cs 的具体实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class Category_Category : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if(!Page.IsPostBack)
        {

```

```

        BindPageData();
    }
}

private void BindPageData()
{
    ///获取数据
    Category category = new Category();
    DataSet ds = category.GetFenleis();
    ///显示数据
    gvCategory.DataSource = ds;
    gvCategory.DataBind();
}

protected void gvCategory_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if(e.CommandName.ToLower() == "update")
    {
        ///重定向到修改分类页面
        Response.Redirect("~/Category/UpdateClasses.aspx?CategoryID="
            + e.CommandArgument.ToString());
        return;
    }
    Category category = new Category();
    if(e.CommandName.ToLower() == "up" || e.CommandName.ToLower() == "down")
    {
        category.UpdateFenleiOrder(
            Int32.Parse(e.CommandArgument.ToString()), e.CommandName);
        BindPageData();
        return;
    }
    if(e.CommandName.ToLower() == "del")
    {
        ///删除选择的商品分类
        if (category.DeleteFenlei(Int32.Parse(e.CommandArgument.ToString())) > 0)
        {
            BindPageData();
        }
        return;
    }
}

protected void gvCategory_RowDataBound(object sender, GridViewRowEventArgs e)
{
    ///添加删除确认的对话框
    ImageButton imgDelete = (ImageButton)e.Row.FindControl("imgDelete");
    if(imgDelete != null)
    {
        imgDelete.Attributes.Add(
            "onclick", "return confirm(\"您确认要删除当前行的商品分类吗? \");");
    }
}

protected void btnAdd_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Category/AddClass.aspx");
}
}

```



## 7.10 商品管理

视频讲解 光盘：视频\第7章\商品管理.avi

完成分类处理后，接下来，是实现对具体某件商品的管理。商品管理其实很简单，只需分别实现对数据库中数据的添加、删除和修改等操作即可。只要记住一条：所有的数据变化都是基于对数据库的操作来实现的。

本项目商品管理模块功能的实现文件如下：

- AddProduct.aspx。
- AddProduct.aspx.cs。
- UpdateProduct.aspx。
- UpdateProduct.aspx.cs。
- Product.aspx。
- Product.aspx.cs。
- ProductPicture.aspx。
- ProductPicture.aspx.cs。

### 7.10.1 商品添加模块

商品添加的功能，是向系统数据库内添加新的商品数据。该功能的实现文件如下。

- AddProduct.aspx：商品添加表单页面。
- AddProduct.aspx.cs：商品添加处理文件。

商品添加处理文件 AddProduct.aspx.cs 的功能是，初始化添加表单界面，并将表单内的数据添加到系统数据库中。具体实现流程如下。

- (1) 引入命名空间，定义 Product\_AddProduct 类。
  - (2) 声明 Page\_Load 方法，进行页面初始化处理。
  - (3) 用户登录判断处理。
  - (4) 调用方法 btnCommit\_Click(object sender, EventArgs e)，执行商品添加处理。
- 文件 AddProduct.aspx.cs 的具体实现代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class Product_AddProduct : System.Web.UI.Page
{
    int categoryID = -1;
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录，并获取用户信息
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Default.aspx");
        }
        userID = Int32.Parse(Session["UserID"].ToString());
    }
}
```

```

protected void btnCommit_Click(object sender, EventArgs e)
{
    if (AddProduct() > 0)
    {
        Response.Redirect("~/Product/Product.aspx");
    }
}
private int AddProduct()
{
    Product product = new Product();
    int productID = product.AddProduct(
        tbName.Text,
        ucProductAttribute.CategoryID,
        userID,
        decimal.Parse(tbPrice.Text.Trim()),
        Int32.Parse(tbStock.Text.Trim()),
        tbRemark.Text);
    if (productID <= 0) return productID;
    ///添加商品特性的值
    foreach (Control c in ucProductAttribute.AttributePanel.Controls)
    {
        if (c is TextBox)
        {
            ///读取属性及其值
            TextBox tbox = (TextBox)c;
            string s = tbox.Text;
            product.AddAttributeValue(
                productID,
                Int32.Parse(tbox.Attributes["IDValue"]),
                tbox.Text);
        }
    }
    return productID;
}
}

```

### 7.10.2 商品修改模块

商品修改模块的功能是，对系统库内的某商品分类信息进行修改。上述功能的实现文件如下。

- UpdateClasses.aspx: 商品修改表单页面。
- UpdateProduct.aspx.cs: 商品修改处理文件。

商品修改处理文件 UpdateProduct.aspx.cs 的功能，是初始化修改表单界面，并将表单内的商品数据更新到系统数据库中。具体的实现流程如下。

- (1) 引入命名空间，定义 Product\_UpdateProduct 类。
- (2) 声明 Page\_Load 方法，进行页面初始化处理。
- (3) 获取修改商品的 ID。
- (4) 调用 BindPageData(int categoryID)方法，获取并显示原数据。
- (5) 按钮可用性判断处理。
- (6) 调用 UpdateProduct 方法，实现对库中此编号商品数据的更新。

文件 UpdateProduct.aspx.cs 的具体实现代码如下所示：

```
///引入新的命名空间
```

```
using ASPNETAJAXWeb.AjaxEBusiness;
using System.Data.SqlClient;
public partial class Product_UpdateProduct : System.Web.UI.Page
{
    private int productID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取商品的ID值
        if(Request.Params["ProductID"] != null)
        {
            productID = Int32.Parse(Request.Params["ProductID"].ToString());
        }
        if(!Page.IsPostBack && productID>0)
        {
            BindPageData(productID);
        }
    }

    private void BindPageData(int productID)
    {
        ///获取商品信息
        Product product = new Product();
        SqlDataReader dr = product.GetSingleProduct(productID);
        if(dr == null) return;
        ///读取商品信息
        if(dr.Read())
        {
            tbName.Text = dr["Name"].ToString();
            tbRemark.Text = dr["Remark"].ToString();
        }
        dr.Close();
    }

    private int UpdateProduct()
    {
        ///修改商品的属性值
        Product product = new Product();
        int success = product.UpdateProduct(productID, tbName.Text, tbRemark.Text);
        if(success <= 0) return success - 1;
        ///修改商品特性的值
        foreach(Control c in ucProductAttribute.AttributePanel.Controls)
        {
            if(c is TextBox)
            {
                TextBox tbx = (TextBox)c;
                product.AddAttributeValue(productID,
                    Int32.Parse(tbx.Attributes["IDValue"]),
                    tbx.Text);
            }
        }
        return success;
    }

    protected void btnCommit_Click(object sender, EventArgs e)
    {
        if(UpdateProduct() > 0)
        {
            Response.Redirect("~/Product/Product.aspx");
        }
    }
}
```



### 7.10.3 商品管理列表模块

商品管理列表模块的功能是，将系统库内的商品信息以列表样式显示出来。并提供对每种分类的操作链接。上述功能的实现文件如下。

- Product.aspx: 商品管理列表显示页面。
- Product.aspx.cs: 商品管理列表处理文件。

商品管理列表处理文件 Product.aspx.cs 的功能，是初始化商品管理列表显示界面。具体的实现流程如下。

- (1) 引入命名空间，定义 Product\_Product 类。
- (2) 声明 Page\_Load 方法，进行页面初始化处理。
- (3) 调用 BindPageData()方法，获取并显示分类树的数据。
- (4) 调用 BindProductData(int categoryID)方法，获取并显示商品数据。
- (5) 定义 gvProduct\_RowCommand 方法，根据用户的操作处理重定向到对应的页面。
- (6) 弹出“删除确认”对话框，将选定信息删除。

文件 Product.aspx.cs 的具体实现代码如下所示：

```
using System.Web.UI.HtmlControls;
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class Product_Product : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    private void BindPageData()
    {
        ///获取数据
        Category category = new Category();
        ///显示分类的层次结构
        category.InitFenleiList(ddlCategory);
        if (ddlCategory.Items.Count > 0) ///如果存在，设置第一项为选择项
        {
            ddlCategory.SelectedIndex = 0;
            BindProductData(Int32.Parse(ddlCategory.SelectedValue));
        }
    }
    private void BindProductData(int categoryID)
    {
        ///获取商品数据
        Product product = new Product();
        DataSet ds = product.GetProductByFenlei(categoryID);
        if (ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        ///显示商品信息
        gvProduct.DataSource = ds;
        gvProduct.DataBind();
    }
    protected void gvProduct_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        if (e.CommandName.ToLower() == "update")
```

```
{
    ///重定向到修改商品页面
    Response.Redirect("~/Product/UpdateProduct.aspx?ProductID="
        + e.CommandArgument.ToString());
    return;
}
if(e.CommandName.ToLower() == "picture")
{
    ///重定向到修改商品图片页面
    Response.Redirect("~/Product/ProductPicture.aspx?ProductID="
        + e.CommandArgument.ToString());
    return;
}
if(e.CommandName.ToLower() == "comment")
{
    ///重定向到管理商品评论页面
    Response.Redirect("~/Product/ProductComment.aspx?ProductID="
        + e.CommandArgument.ToString());
    return;
}
if(e.CommandName.ToLower() == "del")
{
    ///删除选择的商品
    Product product = new Product();
    if(product.DeleteProduct(Int32.Parse(e.CommandArgument.ToString())) > 0)
    {
        BindPageData();
    }
    return;
}
}
protected void gvProduct_RowDataBound(object sender, GridViewRowEventArgs e)
{
    ///添加删除确认的对话框
    ImageButton imgDelete = (ImageButton)e.Row.FindControl("imgDelete");
    if(imgDelete != null)
    {
        imgDelete.Attributes.Add("onclick",
            "return confirm(\"您确认要删除当前行的商品吗? \");");
    }
}
protected void btnAdd_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Product/AddProduct.aspx");
}
protected void ddlCategory_SelectedIndexChanged(object sender, EventArgs e)
{
    BindProductData(Int32.Parse(ddlCategory.SelectedValue));
}
}
```

## 7.10.4 商品图片修改模块

商品图片修改模块的功能，是对系统库内的某商品图片信息进行修改。对应的实现文件如下。

- ProductPicture.aspx: 图片修改表单页面。
- ProductPicture.aspx.cs: 图片修改处理文件。

商品图片修改处理文件 ProductPicture.aspx.cs 的功能,是初始化商品图片界面。具体的实现流程如下所示。

- (1) 引入命名空间,定义 Product\_ProductPicture 类。
- (2) 声明 Page\_Load 方法,进行页面初始化处理。
- (3) 获取修改商品的 ID。
- (4) 调用 BindPageData(int categoryID)方法,获取并显示原数据。
- (5) 调用 btnCommit\_Click(object sender, EventArgs e),实现此编号商品图片的更新。
- (6) 输出对应提示。

文件 ProductPicture.aspx.cs 的具体实现代码如下所示:

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
using System.Data.SqlClient;
using System.IO;
public partial class Product_ProductPicture : System.Web.UI.Page
{
    private int productID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取商品的 ID 值
        if(Request.Params["ProductID"] != null)
        {
            productID = Int32.Parse(Request.Params["ProductID"].ToString());
        }
        ///显示商品信息
        if(!Page.IsPostBack && productID>0)
        {
            BindPageData(productID);
        }
    }
    private void BindPageData(int productID)
    {
        ///获取商品信息
        Product product = new Product();
        SqlDataReader dr = product.GetSingleProduct(productID);
        if(dr == null) return;
        ///显示商品信息
        if(dr.Read())
        {
            lbProductName.Text = dr["Name"].ToString();
            imgProduct.ImageUrl = "~/ " + dr["ImageUrl"].ToString();
        }
        dr.Close();
    }
    protected void btnCommit_Click(object sender, EventArgs e)
    {
        if(productID <= -1) return;
        ///判断上载文件的内容是否为空
        if(fuPicture.HasFile == false || fuPicture.PostedFile.ContentLength <= 0)
        {
            lbMessage.Text = "上载文件的内容为空,请重新选择文件!";
            return;
        }
        ///获取上载文件的属性,如类型、大小、名称等
        string type = fuPicture.PostedFile.ContentType;
        int size = fuPicture.PostedFile.ContentLength;
    }
}

```



```

    ///创建基于时间的文件名称
    string fileName = AjaxEBusinessSystem.CreateDateTimeString();
    string extension = Path.GetExtension(fuPicture.PostedFile.FileName);
    ///判断文件是否合法
    bool isAllow = false;
    foreach(string ext in AjaxEBusinessSystem.ALLOWPICTUREFILELIST)
    {
        if(ext == extension.ToLower())
        {
            isAllow = true;
            break;
        }
    }
    if(isAllow == false) return;
    ///构建保存文件位置的路径
    string url = "img/" + fileName + extension;
    ///映射为物理路径
    string fullPath = Server.MapPath("~/ " + url);
    ///判断文件是否存在
    if(File.Exists(fullPath) == true)
    {
        lbMessage.Text = "上载文件的已经存在, 请重新选择文件! ";
        return;
    }
    try
    {
        ///上载文件
        fuPicture.SaveAs(fullPath);
        Product product = new Product();
        ///添加到数据库中
        if(product.UpdateProductPicture(productID,url) > 0)
        {
            ///显示操作结果
            lbMessage.Text = "恭喜您, 上载商品图片成功.";
            imgProduct.ImageUrl = "~/ " + url;
        }
    }
    catch(Exception ex)
    {
        ///显示错误信息
        lbMessage.Text = "上载文件错误, 错误原因为: " + ex.Message;
        return;
    }
}

```

在很多项目中, 需要动态地管理图片文件。这就不可避免地用到图片上传处理, 但是, 怎样才能确保上传的图片不被侵权使用呢? 解决方法是为图片加水印。下面是一个给图片加水印的通用代码:

```

1. using System;
2. using System.IO;
3. using System.Drawing;
4. using System.Drawing.Imaging;
7. using System.Data;
6. using System.Configuration;
7. using System.Linq;
8. using System.Web;
9. using System.Web.Security;
10. using System.Web.UI;

```

```

11. using System.Web.UI.HtmlControls;
12.
13. using System.Web.UI.WebControls.WebParts;
14. using System.Xml.Linq;
17.
16. /// <summary>
17. /// ImageHandler 的摘要说明
18. /// </summary>
19. public class ImageHandler:IHttpHandler
20. {
21.     private const string waterMark_URL = "~/Images/waterMark.jpg";
22.     private const string defaultImage_URL = "~/Images/Default.jpg";
23.
24.     public ImageHandler()
27.     {
26.         //
27.         //TODO: 在此处添加构造函数逻辑
28.         //
29.     }
30.
31.     public void ProcessRequest(HttpContext context)
32.     {
33.         System.Drawing.Image ImageConver;
34.         if (File.Exists(context.Request.PhysicalPath))
37.         {
36.             //加载文件
37.             ImageConver = Image.FromFile(context.Request.PhysicalPath);
38.             //加载水印图片
39.             Image waterMark = Image.FromFile(
                                     context.Request.MapPath(waterMark_URL));
40.             //重新画布
41.             Graphics g = Graphics.FromImage(ImageConver);
42.             g.DrawImage(waterMark, new Rectangle(ImageConver.Width - waterMark.Width,
ImageConver.Height - waterMark.Height, waterMark.Width, waterMark.Height), 0, 0,
waterMark.Width, waterMark.Height, GraphicsUnit.Pixel);
43.             g.Dispose();
44.             waterMark.Dispose();
47.         }
46.         else
47.         {
48.             ImageConver=Image.FromFile(context.Request.MapPath(defaultImage_URL));
49.         }
50.
51.         //设置输出格式
52.         context.Response.ContentType="image/jpeg";
53.         ImageConver.Save(context.Response.OutputStream,
                             System.Drawing.Imaging.ImageFormat.Jpeg);
54.         ImageConver.Dispose();
57.         context.Response.End();
56.     }
57.
58.     public bool IsReusable
59.     {
60.         get
61.         {
62.             return false;
63.         }
64.     }
67. }

```

## 7.11 购物车

视频讲解 光盘：视频\第7章\购物车.avi

购物车处理的功能，是让用户把预购商品放入购物车，从而完成系统内的购物处理。本模块功能的实现文件是 ViewShoppingCart.aspx 和 ViewShoppingCart.aspx.cs。

在基于 C# 语言的 ASP.NET 项目中，购物车通常使用 Session 或 Cookie 实现，本实例是采用 Cookie 实现的。有关 Session 对象和 Cookie 对象的比较，应注意如下 4 点。

### (1) 应用场景

Cookie 的典型应用场景是 Remember Me 服务，即用户的账户信息通过 Cookie 的形式保存在客户端，当用户再次请求匹配的 URL 的时候，账户信息会被传送到服务端，交由相应的程序完成自动登录等功能。当然，也可以保存一些客户端信息，比如页面布局以及搜索历史等。

Session 的典型应用场景是用户登录某网站之后，将其登录信息放入 Session，在以后的每次请求中，查询相应的登录信息以确保该用户合法。

### (2) 安全性

Cookie 将信息保存在客户端，如果不进行加密的话，无疑会暴露一些隐私信息，安全性很差，一般情况下，敏感信息是经过加密后存储在 Cookie 中的，但很容易就会被窃取。而 Session 只会将信息存储在服务器端，如果存储在文件或数据库中，也有被窃取的可能，只是可能性比 Cookie 小了很多。

Session 在安全性方面，比较突出的是存在会话劫持的问题，这是一种安全威胁。但总体来讲，Session 的安全性要高于 Cookie。

### (3) 性能

Cookie 存储在客户端，消耗的是客户端的 I/O 和内存，而 Session 存储在服务端，消耗的是服务端的资源。但是 Session 对服务器造成的压力比较集中，而 Cookie 很好地分散了资源消耗，就这点来说，Cookie 是优于 Session 的。

### (4) 时效性

Cookie 可以通过设置有效期，使其在较长时间内存在于客户端，而 Session 一般只有比较短的有效期(用户主动销毁 Session 或关闭浏览器后将引发超时)。

### 7.11.1 购物车组件的设计

本功能模块所使用的处理函数是由文件 ShoppingCart.cs 来实现的，其主要的功能，是在 ASPNETAJAXWeb.AjaxEBusiness 空间内建立需要的类，并定义多个函数方法，以实现购物车数据的处理。

在文件 Product.cs 中，分别定义了两个新类，实现对购物车的处理，具体说明如下：

- ShoppingCartItem 类。
- ShoppingCart 类。



### 1. 定义 ShoppingCartItem 类

在 ShoppingCartItem 类中,封装了购物车内商品的基本信息,例如商品编号、名称、数量和价格等。为此, ShoppingCartItem 类定义了 4 个属性,分别传递上述 4 个元素信息。

定义 ShoppingCartItem 类的实现代码如下所示:

```
namespace ASPNETAJAXWeb.AjaxEBusiness
{
    public class ShoppingCartItem
    {
        private int productID = -1;
        private string name = string.Empty;
        private int number = 0;
        private decimal price = 0.0m;
    }
}
```

### 2. 定义 ShoppingCart 类

在 ShoppingCart 类中,首先定义了两个私有变量 session 和 shoppingCartList; 然后定义了一个公开变量 SHOPPINTCARTKEY; 最后定义了一个公开属性 ShoppingCartList。对应的实现代码如下所示:

```
public class ShoppingCart
{
    public const string SHOPPINTCARTKEY = "SHOPPINTCARTKEY";
    private ArrayList shoppingCartList;
    private HttpSessionState session = null;
    public ArrayList ShoppingCartList
    {
        get
        {
            return shoppingCartList;
        }
    }
    private ShoppingCart()
    {
    }
}
```

### 3. 定义处理方法

在 ShoppingCart 类中,定义 5 个方法,实现对购物车内的数据的处理,具体如下:

- ShoppingCart(HttpSessionState session)
- AddProductToShoppingCart(ShoppingCartItem product)
- DeleteProductFromShoppingCart(ShoppingCartItem product)
- UpdateShoppingCart(ArrayList products)
- ClearShoppingCart()

上述方法的具体运行流程如图 7-7 所示。

#### (1) 购物车初始化

购物车初始化就是载入页面时对购物车数据进行初始化处理。这一功能是由方法 ShoppingCart(HttpSessionState session)实现的,具体的实现流程如下。

- ① 初始化保存变量 shoppingCartList。
- ② 将值保存在 session 对象中。



图 7-7 购物车处理模块的运行流程

上述功能的对应实现代码如下所示：

```

public ShoppingCart(HttpSessionState session)
{
    this.session = session;
    if(session != null)
    {
        if(session[SHOPPINGCARTKEY] != null)
        {
            shoppingCartList = (ArrayList)session[SHOPPINGCARTKEY];
        }
        else
        {
            shoppingCartList = new ArrayList();
            session[SHOPPINGCARTKEY] = shoppingCartList;
        }
    }
}
    
```

## (2) 添加购物车商品

购物车商品添加就是把用户选取的商品添加到购物车内。

该功能是由方法 `AddProductToShoppingCart(ShoppingCartItem product)` 实现的，具体的实现流程如下。

- ① 获取要添加的商品。
- ② 如果购物车内没有此商品，则将商品添加到购物车内。
- ③ 如果购物车内有此商品，则修改此商品的数量。
- ④ 把更新后的购物车数据重新保存到 session 对象中。

上述功能的对应实现代码如下所示：

```

public int AddProductToShoppingCart(ShoppingCartItem product)
{
    if(product == null) return -1;
    ///获取购物车中的商品
    shoppingCartList = (ArrayList)session[SHOPPINGCARTKEY];
    if(shoppingCartList == null) return -1;
    ///比较购物车中是否已经添加了该商品
    int index = 0;
    for(index=0; index<shoppingCartList.Count; index++)
    
```

```

{
    ///如果已经添加了, 则修改购物车中商品的数量
    if(((ShoppingCartItem) shoppingCartList[index]).ProductID == product.ProductID)
    {
        ((ShoppingCartItem) shoppingCartList[index]).Number++;
        break;
    }
}
///如果没有添加, 则把该商品添加到购物车中
if(index == shoppingCartList.Count)
{
    shoppingCartList.Add(product);
}
///重新保存购物车中的数据
session[SHOPPINTCARTKEY] = shoppingCartList;
return 1;
}

```

### (3) 修改购物车商品

购物车商品修改即把购物车内的某商品数量修改为 products 参数内的商品数量, 该功能是由方法 UpdateShoppingCart(ArrayList products)实现的, 具体的实现流程如下。


- ① 获取购物车内的商品。
- ② 使用 foreach 一词处理购物车内的每一个商品。
- ③ 如果当前商品在 products 参数内出现, 则把当前处理的商品数量修改为 products 参数内的该商品数量。
- ④ 把更新后的购物车数据重新保存到 session 对象中。

上述功能的对应实现代码如下所示:

```

public int UpdateShoppingCart(ArrayList products)
{
    if(products == null || products.Count <= 0) return -1;
    ///获取购物车中的商品
    shoppingCartList = (ArrayList)session[SHOPPINTCARTKEY];
    if(shoppingCartList == null) return -1;
    ///更新购物车中的商品
    for(int index=0; index<shoppingCartList.Count; index++)
    {
        foreach(ShoppingCartItem product in products)
        {
            if(((ShoppingCartItem) shoppingCartList[index]).ProductID
                == product.ProductID)
            {
                ((ShoppingCartItem) shoppingCartList[index]).Number = product.Number;
                break;
            }
        }
    }
    ///重新保存购物车中的数据
    session[SHOPPINTCARTKEY] = shoppingCartList;
    return 1;
}

```

 **注意:** 上面的 products 参数, 即购物车表单内获取商品数量的参数。用户可以在 products 参数表单内输入预购商品的数量。



#### (4) 删除购物车商品

购物车商品删除即把购物车内的某商品删除。

该功能是由方法 `DeleteProductFromShoppingCart(ShoppingCartItem product)` 实现的，其具体实现流程如下。

- ① 获取购物车内的商品。
- ② 使用 `foreach` 一词处理购物车内的每一个商品。
- ③ 获取删除商品的编号。
- ④ 把更新后的购物车数据重新保存到 `session` 对象中。

上述功能的对应实现代码如下所示：

```
public int DeleteProductFromShoppingCart(ShoppingCartItem product)
{
    if(product == null) return -1;

    ///获取购物车中的商品
    shoppingCartList = (ArrayList)session[SHOPPINTCARTKEY];
    if(shoppingCartList == null) return -1;
    ///从购物车查找被删除的商品
    foreach(ShoppingCartItem item in shoppingCartList)
    {
        if(item.ProductID == product.ProductID)
        {
            ///移除该商品
            shoppingCartList.Remove(item);
            break;
        }
    }

    ///重新保存购物车中的数据
    session[SHOPPINTCARTKEY] = shoppingCartList;
    return 1;
}
```

#### (5) 清空购物车商品

购物车商品的清空就是将购物车内的所有商品清空。

该功能是由方法 `ClearShoppingCart()` 实现的，具体的实现流程如下。

- ① 获取购物车内的商品。
- ② 清空购物车内的每一个商品。
- ③ 把更新后的购物车数据重新保存到 `session` 对象中。

上述功能的对应实现代码如下所示：

```
public int ClearShoppingCart()
{
    ///获取购物车中的商品
    shoppingCartList = (ArrayList)session[SHOPPINTCARTKEY];
    if(shoppingCartList == null) return -1;

    ///清空购物车中的商品
    shoppingCartList.Clear();
    session[SHOPPINTCARTKEY] = null;
    return 1;
}
```

## 7.11.2 购物车商品添加模块

购物车商品添加模块的功能，是当用户在系统页面内单击某商品后的“加入购物车”按钮后，将此商品添加到购物车内。

在下面的内容中，将详细介绍购物车商品添加模块的具体实现过程。

### 1. 索引设置

当用户单击“加入购物车”按钮后，将首先激活 btnBuy 购物车按钮事件，然后把它的 CommandArgument 属性设置为当前索引。上述功能的具体实现代码如下所示：

```
protected void gvProduct_RowDataBound(object sender, GridViewRowEventArgs e)
{
    Button btnBuy = (Button)e.Row.FindControl("btnBuy");
    if(btnBuy != null)
    {
        ///设置 CommandArgument 属性的值为当前行的索引
        btnBuy.CommandArgument = e.Row.RowIndex.ToString();
    }
}
```

### 2. 添加处理

在商品显示列表文件中，通过定义的 gvProduct\_RowCommand(object sender, GridViewCommandEventArgs e) 相关事件，实现购物车内的商品添加功能。

当用户单击“加入购物车”按钮后，将会激活 buy 属性，从而将当前商品添加到购物车内，具体操作流程如下。

- (1) 创建表示商品 ShoppingCartItem 类的对象 item。
- (2) 设置 item 对象的 ID、数量、名称和价格属性。
- (3) 调用 ShoppingCart 类的 AddProductToShoppingCart 方法，把 item 对象添加到购物车内。


上述功能的具体实现代码如下所示：

```
protected void gvProduct_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if(e.CommandName == "buy")
    {
        ShoppingCartItem item = new ShoppingCartItem();
        int rowIndex = Int32.Parse(e.CommandArgument.ToString());
        if(rowIndex <= -1 || rowIndex >= gvProduct.Rows.Count) return;
        ///获取商品 ID 和数量
        item.ProductID = Int32.Parse(gvProduct.DataKeys[rowIndex]["ID"].ToString());
        item.Number = 1;
        ///获取商品名称
        Label lbName = (Label)gvProduct.Rows[rowIndex].FindControl("lbName");
        if(lbName != null)
        {
            item.Name = lbName.Text;
        }
        ///获取商品价格
        Label lbPrice = (Label)gvProduct.Rows[rowIndex].FindControl("lbPrice");
        if(lbPrice != null)
```

```

    {
        item.Price = decimal.Parse(lbPrice.Text);
    }
    ShoppingCart shoppingCart = new ShoppingCart(Session);
    if (shoppingCart.AddProductToShoppingCart(item) > -1)
    {
        AjaxEBusinessSystem.ShowAjaxDialog(
            (Button)e.CommandSource, "恭喜您, 添加商品到购物车成功。");
    }
}
}

```

 **注意：** 上述购物车商品添加程序在商品列表显示页面中定义, 包括文件 Product.aspx.cs 和 5 种不同方式排序处理页面中都有定义。

### 7.11.3 购物车管理

购物车管理模块的功能, 是当用户将商品加入购物车后, 可以查看购物车内的商品信息, 并对里面的商品进行操作管理。

#### 1. 查看购物车

查看购物车即显示某购物车的详细信息, 此功能的实现文件如下:

- ViewShoppingCart.aspx。
- ViewShoppingCart.aspx.cs。

在文件 ViewShoppingCart.aspx.cs 内, 将初始化显示某购物车内的商品信息。具体的实现流程如下。

- (1) 引入命名空间, 声明 ShoppingCart\_ViewShoppingCart 类。
- (2) 定义 Page\_Load, 做初始化处理。
- (3) 定义 BindPageData(), 获取并显示购物车内的商品信息。

在文件 ViewShoppingCart.aspx.cs 内, 上述功能的对应实现代码如下所示:

```

public partial class ShoppingCart_ViewShoppingCart : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    private void BindPageData()
    {
        ///获取购物车的商品
        ShoppingCart shoppingCart = new ShoppingCart(Session);
        ///绑定数据并显示商品
        gvProduct.DataSource = shoppingCart.ShoppingCartList;
        gvProduct.DataBind();
    }
}

```

#### 2. 修改购物车数量

在修改购物车数量的界面中, 用户可以在某商品后面的数量文本框内输入合法数值,



然后单击“保存修改”按钮，实现对购物车内商品数量的修改处理。

上述功能是由文件 ViewShoppingCart.aspx.cs 内的 void btnStore\_Click(object sender, EventArgs e)事件实现的，具体的实现流程如下。

- (1) 获取购物车信息。
- (2) 检查变量 shoppingCart 内的商品数量和显示页面中显示的数量是否相等。
- (3) 如果不相等，则终止该事件的执行。
- (4) 创建保存商品的临时数组 products。
- (5) 将显示页面内的商品添加到临时数组 products 中。
- (6) 调用 shoppingCart 类的方法 UpdateShoppingCart，实现对数据的更新。

在文件 ViewShoppingCart.aspx.cs 内，上述功能的对应实现代码如下所示：

```
protected void btnStore_Click(object sender, EventArgs e)
{
    ///获取购物车的商品
    ShoppingCart shoppingCart = new ShoppingCart(Session);
    if(shoppingCart == null || shoppingCart.ShoppingCartList == null
        || shoppingCart.ShoppingCartList.Count <= 0) return;
    ///检查购物车中的商品与显示的商品是否相等，如果不相等，则数据错误
    if(shoppingCart.ShoppingCartList.Count != gvProduct.Rows.Count) return;
    ArrayList products = new ArrayList();
    foreach(GridDataRow row in gvProduct.Rows)
    {
        ///找到输入商品数量的控件
        TextBox tbNumber = (TextBox)row.FindControl("tbNumber");
        if(tbNumber == null) return;
        ///获取商品数量
        int number = -1;
        if(Int32.TryParse(tbNumber.Text.Trim(), out number) == false) return;
        ///创建一个子项，并添加到临时数组中
        ShoppingCartItem item = new ShoppingCartItem();
        ///设置子项的名称、数量、价格和商品 ID 值
        item.Name =
            ((ShoppingCartItem)shoppingCart.ShoppingCartList[row.RowIndex]).Name;
        item.Number = number;
        item.Price =
            ((ShoppingCartItem)shoppingCart.ShoppingCartList[row.RowIndex]).Price;
        item.ProductID =
            ((ShoppingCartItem)shoppingCart.ShoppingCartList[row.RowIndex]).ProductID;
        products.Add(item);
    }
    ///修改购物车中的商品数量
    shoppingCart.UpdateShoppingCart(products);
}
```

### 3. 删除购物车内的商品

在订单中，用户可以单击某商品后的图标，从购物车内删除这个商品。

文件 ViewShoppingCart.aspx.cs 内上述功能的实现事件如下：

- gvProduct\_RowDataBound(object sender, GridViewRowEventArgs e)事件。
- gvProduct\_RowCommand(object sender, GridViewCommandEventArgs e)事件。

上述功能的具体实现流程如下。

- (1) 判断单击删除的图标。

- (2) 弹出“删除确认”对话框。
- (3) 单击“确定”按钮后，开始删除此商品。
- (4) 调用 BindPageData() 重新绑定数据。

在文件 ViewShoppingCart.aspx.cs 内，上述功能的对应实现代码如下所示：

```
protected void gvProduct_RowDataBound(object sender, GridViewRowEventArgs e)
{
    ///添加删除确认的对话框
    ImageButton imgDelete = (ImageButton)e.Row.FindControl("imgDelete");
    if(imgDelete != null)
    {
        imgDelete.Attributes.Add(
            "onclick", "return confirm(\"您确认要删除当前行的商品吗?\");");
    }
}

protected void gvProduct_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if(e.CommandName.ToLower() == "del")
    {
        ///获取购物车的商品
        ShoppingCart shoppingCart = new ShoppingCart(Session);
        if(shoppingCart == null || shoppingCart.ShoppingCartList == null
            || shoppingCart.ShoppingCartList.Count <= 0) return;

        ///创建被删除的商品
        ShoppingCartItem deleteItem = new ShoppingCartItem();
        deleteItem.ProductID = Int32.Parse(e.CommandArgument.ToString());

        ///删除选中的商品
        shoppingCart.DeleteProductFromShoppingCart(deleteItem);

        ///重新绑定商品数据
        BindPageData();
    }
}
```

#### 4. 提交处理和购买处理

用户可以单击“我要创建订单”按钮，实现创建购买订单功能。也可以单击“我要购买商品”按钮，继续来到商品列表界面购买新的商品。

文件 ViewShoppingCart.aspx.cs 内的上述功能的实现事件如下。

- Commit\_Click(object sender, EventArgs e)事件：重定向到订单页面。
- btnAdd\_Click(object sender, EventArgs e)事件：重定向到商品列表页面。

文件 ViewShoppingCart.aspx.cs 内上述功能的具体实现代码如下：

```
protected void btnCommit_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Order/AddOrder.aspx");
}

protected void btnAdd_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Product.aspx?CategoryID=27");
}
```

## 7.12 订单处理

视频讲解 光盘：视频\第7章\订单处理.avi

完成购物车模块后，接下来，开始实现订单处理模块的设计工作。订单处理的功能是将购物车生成订单，实现订单提交处理，从而完成在线购物，并对提交的订单进行处理和维。本系统中，订单处理模块功能的实现文件如下：

- AddOrder.aspx。
- AddOrder.aspx.cs。
- OrderList.aspx。
- OrderList.aspx.cs。
- ViewOrder.aspx。
- ViewOrder.aspx.cs。
- OrderManage.aspx。
- OrderManage.aspx.cs。

### 7.12.1 生成订单编号

当单击“提交并创建订单”按钮后，将自动生成时间字符格式的订单名称。订单名称的格式是由文件 ASPNETAJAXWeb.cs 内的 CreatorOrderNo(string no)定义的，其具体实现代码如下所示：

```
public static string CreatorOrderNo(string no)
{
    DateTime now = DateTime.Now;
    string orderNoString = now.Year.ToString() + now.Month.ToString().PadLeft(2, '0')
        + now.Day.ToString().PadLeft(2, '0') + no.PadLeft(4, '0');
    return (orderNoString);
}
```

用户单击“提交并创建订单”按钮后，将激活 Commit\_Click(object sender, EventArgs e)事件，在创建订单之前实现订单编号的生成。具体的实现流程如下。

- (1) 调用 Order 类的方法 GetOrderLastOrderNo(), 获取当天的最后一个编号，并保存在 orderNo 变量中。
- (2) 如果 orderNo 变量值为空，则设置序列号为“0001”。
- (3) 如果 orderNo 变量值不为空，则递增生成序列号。
- (4) 调用 ASPNETAJAXWeb.cs 内的 CreatorOrderNo(string no)事件，创建新的订单编号。
- (5) 将编号保存到 orderNo 变量中。

上述功能的对应实现代码如下所示：

```
protected void btnCommit_Click(object sender, EventArgs e)
{
    Order order = new Order();
    ///获取当天最近的订单编号
    string orderNo = order.GetOrderLastOrderNo();
```



```

///创建下一个订单编号的基数
if(string.IsNullOrEmpty(orderNo) == true)
{
    ///下一个订单号的基数为1
    orderNo = "0001";
}
else
{
    ///创建下一个订单号的基数
    orderNo = (Int32.Parse(orderNo.Substring(8)) + 1).ToString();
}
///创建下一个订单编号
orderNo = AjaxEBusinessSystem.CreaterOrderNo(orderNo);
}

```

## 7.12.2 提交并创建新订单

提交、创建订单是指将订单信息提交给订单模块来处理，并生成指定编号的在线购买订单。上述功能的实现文件如下。

- AddOrder.aspx: 订单创建界面文件。
- AddOrder.aspx.cs: 订单创建处理文件。

订单创建处理文件 AddOrder.aspx.cs 的功能是，初始化订单信息，将生成的订单信息添加到系统数据库中。其具体实现流程如下。

- (1) 引入命名空间，定义 Order\_AddOrder 类。
- (2) 声明 Page\_Load，做页面初始化处理。
- (3) 判断选用户是否登录。
- (4) 调用 BindPageData()，获取并显示数据。
- (5) 激活 btnCommit\_Click(object sender, EventArgs e)事件。
- (6) 创建生成订单编号。
- (7) 获取当前购物车的商品信息。
- (8) 计算商品的总数量和总金额。
- (9) 调用方法 AddOrder，创建一个新订单。
- (10) 调用方法 AddOrderItem，将购物车信息添加到库中。
- (11) 调用方法 ClearShoppingCart，清空购物车内的商品。
- (12) 重定向返回订单详情页面。

文件 AddOrder.aspx.cs 的具体实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class Order_AddOrder : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Default.aspx");
            return;
        }
    }
}

```

```

    }
    ///获取用户信息
    userID = Int32.Parse(Session["UserID"].ToString());
    if(!Page.IsPostBack)
    {
        BindPageData();
    }
}
private void BindPageData()
{
    ///获取购物车的商品
    ShoppingCart shoppingCart = new ShoppingCart(Session);
    ///绑定数据并显示商品
    gvProduct.DataSource = shoppingCart.ShoppingCartList;
    gvProduct.DataBind();
}
protected void btnCommit_Click(object sender, EventArgs e)
{
    Order order = new Order();
    ///获取当天最近的订单编号
    string orderNo = order.GetOrderLastOrderNo();
    ///创建下一个订单编号的基数
    if(string.IsNullOrEmpty(orderNo) == true)
    {
        ///下一个订单号的基数为1
        orderNo = "0001";
    }
    else
    {
        ///创建下一个订单号的基数
        orderNo = (Int32.Parse(orderNo.Substring(8)) + 1).ToString();
    }
    ///创建下一个订单编号
    orderNo = AjaxEBusinessSystem.CreaterOrderNo(orderNo);
    ///获取购物车的商品
    ShoppingCart shoppingCart = new ShoppingCart(Session);
    ///计算购物车中的商品总数量和总金额
    int totalNumber = 0;
    decimal totalMoney = 0.0m;
    foreach(ShoppingCartItem item in shoppingCart.ShoppingCartList)
    {
        ///数量和金额累加
        totalNumber += item.Number;
        totalMoney += item.Number * item.Price;
    }
    ///创建订单
    int orderID = order.AddOrder(orderNo, userID, totalNumber, totalMoney);
    if(orderID > 0)
    {
        ///创建订单的商品项
        foreach(ShoppingCartItem item in shoppingCart.ShoppingCartList)
        {
            order.AddOrderItem(orderID, item.ProductID, item.Number);
        }
    }
    ///清空购物车中的商品
    shoppingCart.ClearShoppingCart();
    ///重定向到预览订单的页面
    Response.Redirect("~/Order/ViewOrder.aspx?OrderID=" + orderID.ToString());
}

```

```
protected void btnAdd_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Product.aspx?CategoryID=27");
}
}
```

### 7.12.3 查看订单详情

订单详情功能即把系统内指定编号的订单信息显示出来，对应的实现文件如下：

- ViewOrder.aspx: 查看订单页面文件。
- ViewOrder.aspx.cs: 查看订单处理文件。

查看订单处理文件 ViewOrder.aspx.cs 的功能，是从数据库中将相应的订单信息显示出来。具体的实现流程如下所示。

- (1) 引入命名空间，定义 Order\_ViewOrder 类。
- (2) 定义订单编号变量 orderID。
- (3) 声明 Page\_Load 方法，进行页面初始化处理。
- (4) 订单编号判断处理。
- (5) 调用 BindPageData()方法，获取订单数据。
- (6) 显示订单编号和创建时间。
- (7) 显示订单数量和商品总金额。
- (8) 调用方法 GetOrderItemByOrder，获取此编号订单内的所有商品信息。
- (9) 绑定显示数据。

ViewOrder.aspx.cs 文件的具体实现代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
using System.Data.SqlClient;
public partial class Order_ViewOrder : System.Web.UI.Page
{
    int orderID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取订单信息
        if (Request.Params["OrderID"] != null)
        {
            orderID = Int32.Parse(Request.Params["OrderID"].ToString());
        }
        if (!Page.IsPostBack && orderID > 0)
        {
            BindPageData(orderID);
        }
    }
    private void BindPageData(int orderID)
    {
        Order order = new Order();
        ///获取订单信息
        SqlDataReader dr = order.GetSingleOrder(orderID);
        if (dr == null) return;
        if (dr.Read())
        {
            ///显示订单信息
            lbOrderNo.Text = dr["OrderNo"].ToString();
        }
    }
}
```



```

        lbCreateDate.Text = dr["CreateDate"].ToString();
        ///格式化为货币格式
        lbTotalMoney.Text = string.Format("{0:C}", dr["TotalMoney"]);
        lbTotalNumber.Text = dr["TotalNumber"].ToString();
    }
    dr.Close();
    ///显示订单的详细商品信息
    gvProduct.DataSource = order.GetOrderItemByOrder(orderID);
    gvProduct.DataBind();
}
}

```

## 7.12.4 订单列表模块

订单列表模块的功能，是将系统内某编号用户的订单信息显示出来。该功能的实现文件如下：

- OrderList.aspx: 订单列表页面文件。
- OrderList.aspx.cs: 订单列表页面的后台处理文件。

订单列表处理文件 OrderList.aspx.cs 的功能，是初始化订单列表信息，将指定用户的对应订单信息以列表样式显示出来。其具体实现流程如下所示。

- (1) 引入命名空间，定义 Order\_OrderList 类。
- (2) 声明 Page\_Load 方法，进行页面初始化处理。
- (3) 用户登录判断处理。
- (4) 获取用户信息。
- (5) 调用 BindPageData() 方法，绑定显示订单数据。

OrderList.aspx.cs 文件的具体实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class Order_OrderList : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Default.aspx");
            return;
        }
        ///获取用户信息
        userID = Int32.Parse(Session["UserID"].ToString());
        if(!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    private void BindPageData()
    {
        ///获取历史订单
        Order order = new Order();
        ///绑定数据并显示订单
        gvOrder.DataSource = order.GetOrderByUser(userID);
    }
}

```

```

        gvOrder.DataBind();
    }
}

```

## 7.12.5 订单状态处理模块

订单状态处理模块的功能，是将系统内的订单信息列表显示出来，并提供对应的链接，实现对某订单的状态的处理。该功能的实现文件如下。

- OrderManage.aspx: 订单状态列表页面文件。
- OrderManage.aspx.cs: 订单状态页面的后台处理文件。

订单状态处理文件 OrderManage.aspx.cs 的功能，是初始化订单列表信息，并对某未处理的订单进行处理。具体的实现流程如下。

(1) 引入命名空间，定义 Order\_OrderManage 类。

(2) 声明 Page\_Load 方法，进行页面初始化处理。

(3) 用户登录判断处理。

(4) 获取用户 ID。

(5) 调用 BindPageData() 方法，获取并显示订单数据。

(6) 调用 CheckStockAndSale(int orderID) 方法，判断库存数量是否满足用户订单的需求数量。其具体处理流程如下。

① 调用 Order 类的方法 GetOrderItemByOrder(orderID)，获取订单信息。

② 将订单信息保存在变量 ds 中。

③ 检查变量 ds 中各商品的数量是否小于或等于该商品在系统中的库存数量，并将检测结果保存在变量 isAllowSale 中。

④ 如果检测的数量大于系统库存量，则停止事件处理。

⑤ 如果检测结果是 true，则可以对此订单进行处理，并修改系统库存中对应商品的库存数。

⑥ 操作成功，输出成功提示。

⑦ 重新载入显示订单信息。

(7) 调用 gvOrder\_RowCommand 事件，输出对应的判断处理结果。如果库存不够，则显示对应的提示，反之，则显示处理成功提示。

### 1. 初始化处理

本流程的功能，是引入命名空间和定义 Order\_OrderManage 类，并初始化载入页面程序。该功能的对应实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
using System.Data.SqlClient;
public partial class Order_OrderManage : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)

```

```

{
    Response.Redirect("~/Default.aspx");
    return;
}
///获取用户信息
userID = Int32.Parse(Session["UserID"].ToString());
if(!Page.IsPostBack)
{
    BindPageData();
}
}

```

## 2. 获取显示数据

本流程的功能，是定义 BindPageData()，获取并显示对应的订单信息。该功能的对应实现代码如下所示：

```

private void BindPageData()
{
    ///获取历史订单
    Order order = new Order();
    ///绑定数据并显示订单
    gvOrder.DataSource = order.GetOrderByUser(userID);
    gvOrder.DataBind();
}

```

## 3. 库存判断处理

本流程将会定义 gvOrder\_RowCommand(object sender, GridViewCommandEventArgs e) 和 CheckStockAndSale(int orderID)，进行库存判断处理，并将处理的结果显示出来。

该功能的对应实现代码如下所示：

```

protected void gvOrder_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if(e.CommandName.ToString() == "sale")
    {
        ///检查库存
        int orderID = Int32.Parse(e.CommandArgument.ToString());
        if(CheckStockAndSale(orderID) == false)
        {
            AjaxEBusinessSystem.ShowAjaxDialog(
                (Button)e.CommandSource, "库存不够，不能处理该订单");
            return;
        }
        AjaxEBusinessSystem.ShowAjaxDialog(
            (Button)e.CommandSource, "恭喜您，处理订单成功。");
    }
}

private bool CheckStockAndSale(int orderID)
{
    ///获取订单信息
    Order order = new Order();
    DataSet ds = order.GetOrderItemByOrder(orderID);
    if(ds==null || ds.Tables.Count<=0 || ds.Tables[0].Rows.Count<=0) return false;
    ///判断库存是否足够
    Product product = new Product();
    bool isAllowSale = true;
}

```




```

foreach(DataRow row in ds.Tables[0].Rows)
{
    ///读取商品信息
    SqlDataReader dr =
        product.GetSingleProduct(Int32.Parse(row["ProductID"].ToString()));
    if(dr == null)
    {
        isAllowSale = false;
        break;
    }
    if(dr.Read())
    {
        ///判断库存数量是否足够, 如果不够, 则不能卖该商品
        if(Int32.Parse(dr["Stock"].ToString())
            < Int32.Parse(row["Number"].ToString()))
        {
            isAllowSale = false;
            break;
        }
    }
    dr.Close();
}
if(isAllowSale == false) return false;
///修改此次交易商品的库存和销售数量
foreach(DataRow row in ds.Tables[0].Rows)
{
    ///修改库存信息和销售数量
    if(product.UpdateProductStock(
        Int32.Parse(row["ProductID"].ToString()),
        Int32.Parse(row["Number"].ToString()))<=0)
    {
        isAllowSale = false;
        break;
    }
}
if(isAllowSale == false) return false;
///提交该订单, 并重新显示数据
if(order.UpdateOrderStatus(orderID, 1) > 0)
{
    BindPageData();
}
return isAllowSale;
}

```

## 7.13 商品评论模块

视频讲解  光盘: 视频\第7章\商品评论模块.avi

商品评论与留言簿系统的原理类似, 在本项目中, 建议把商品评论模块做得简单一点, 只要能实现留言功能即可, 因为复杂了反而不好操作。对于本项目来说, 本来系统已经很复杂, 不是核心模块的评论功能应该尽量以简单为主, 这样可以避免很多不必要的麻烦。

商品评论的功能是对系统内商品的评论信息进行相关的处理操作。其对应的实现文件如下:

- ShowProduct.aspx。

- ShowProduct.aspx.cs。
- ProductComment.aspx。
- ProductComment.aspx.cs。

### 7.13.1 评论显示模块

评论显示模块的功能，是将系统内的某编号商品的评论信息显示出来。对应的实现文件如下：

- ShowProduct.aspx：列表显示评论。
- ShowProduct.aspx.cs：评论显示处理。

评论显示处理文件 ShowProduct.aspx.cs 的功能，是获取并显示指定编号商品的评论信息。文件 ShowProduct.aspx.cs 中上述功能的实现代码如下：

```
private void BindPageData(int productID)
{
    ///获取商品评论
    Product product = new Product();
    SqlDataReader dr = product.GetSingleProduct(productID);
    if(dr == null) return;
    if(dr.Read())
    {
        ///读取商品信息
        ProductName = dr["Name"].ToString();
        ViewCount = dr["ViewCount"].ToString();
        CreateDate = dr["CreateDate"].ToString();
        Username = dr["Username"].ToString();
        Remark = dr["Remark"].ToString();
        lbPrice.Text = string.Format("{0:f2}", dr["Price"]);
        imgProduct.ImageUrl = "~/ " + dr["PictureUrl"].ToString();
        ///获取属性的数据
        DataSet ds = product.GetAttributeByProduct(productID);
        ///绑定并显示属性的数据
        gvAttribute.DataSource = ds;
        gvAttribute.DataBind();
    }
    dr.Close();
    ///显示商品的评论
    gvComment.DataSource = product.GetCommentByProduct(productID);
    gvComment.DataBind();
}
```

### 7.13.2 评论管理模块

评论管理模块的功能，是将系统内某编号商品的评论信息列表显示出来，并提供对应链接，实现对评论的管理。上述功能的实现文件如下。

- ProductPinglun.aspx：评论列表显示页面。
- ProductPinglun.aspx.cs：评论列表处理文件。

评论列表处理文件 ProductPinglun.aspx.cs 的功能，是初始化评论管理列表信息，并且将指定编号的评论信息删除。具体的实现流程如下所示。

(1) 引入命名空间，定义 Product\_ProductComment 类。

- (2) 声明 Page\_Load 方法, 进行页面初始化处理。
- (3) 获取评论编号 ID, 并保存在变量 ProductID 中。
- (4) 如果 ProductID 大于 0, 则调用函数 BindPageData(int productID)。
- (5) 调用 BindPageData(int productID)方法, 获取并显示评论信息。
- (6) 定义 RowDataBound 方法, 弹出“删除确认”对话框。
- (7) 定义 gvComment\_RowCommand 方法, 删除指定编号的评论。

文件 ProductPinglun.aspx.cs 的具体实现代码如下所示:

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
public partial class Product_ProductComment : System.Web.UI.Page
{
    private int productID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取商品的 ID
        if (Request.Params["ProductID"] != null)
        {
            productID = Int32.Parse(Request.Params["ProductID"].ToString());
        }
        if (productID > 0)
        {
            BindPageData(productID);
        }
    }
    private void BindPageData(int productID)
    {
        ///显示商品的评论
        Product product = new Product();
        gvComment.DataSource = product.GetCommentByProdcut(productID);
        gvComment.DataBind();
    }
    protected void gvComment_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        Product product = new Product();
        if (e.CommandName.ToLower() == "del")
        {
            ///删除选择的商品评论
            if (product.DeleteProductComment(
                Int32.Parse(e.CommandArgument.ToString())) > 0)
            {
                BindPageData(productID);
            }
            return;
        }
    }
    protected void gvComment_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        ///添加删除确认的对话框
        ImageButton imgDelete = (ImageButton)e.Row.FindControl("imgDelete");
        if (imgDelete != null)
        {
            imgDelete.Attributes.Add(
                "onclick", "return confirm(\"您确认要删除当前行的商品评论吗? \");");
        }
    }
}

```



## 7.14 商品搜索模块

视频讲解 光盘：视频\第7章\商品搜索模块.avi

检索功能也是基于数据库来实现的，即从数据库中检索出指定的商品。此模块的最大好处，是它能够帮助用户迅速检索到自己需要的商品。本系统中，商品搜索模块对应的实现文件如下：

- SearchByKey.aspx。
- SearchByKey.aspx.cs。
- SearchByPrice.aspx。
- SearchByPrice.aspx.cs。
- SearchByTime.aspx。
- SearchByTime.aspx.cs。

下面将介绍按关键字搜索模块的功能和实现。

按关键字搜索模块的功能，是按照系统商品名称的关键字进行搜索处理。对应的实现文件如下。

- SearchByKey.aspx：搜索表单页面文件。
- SearchByKey.aspx.cs：搜索表单页面的处理文件。

搜索表单页面的处理文件 SearchByKey.aspx.cs 的功能，是按照搜索表单的关键字进行检索处理。

具体的实现流程如图 7-8 所示。

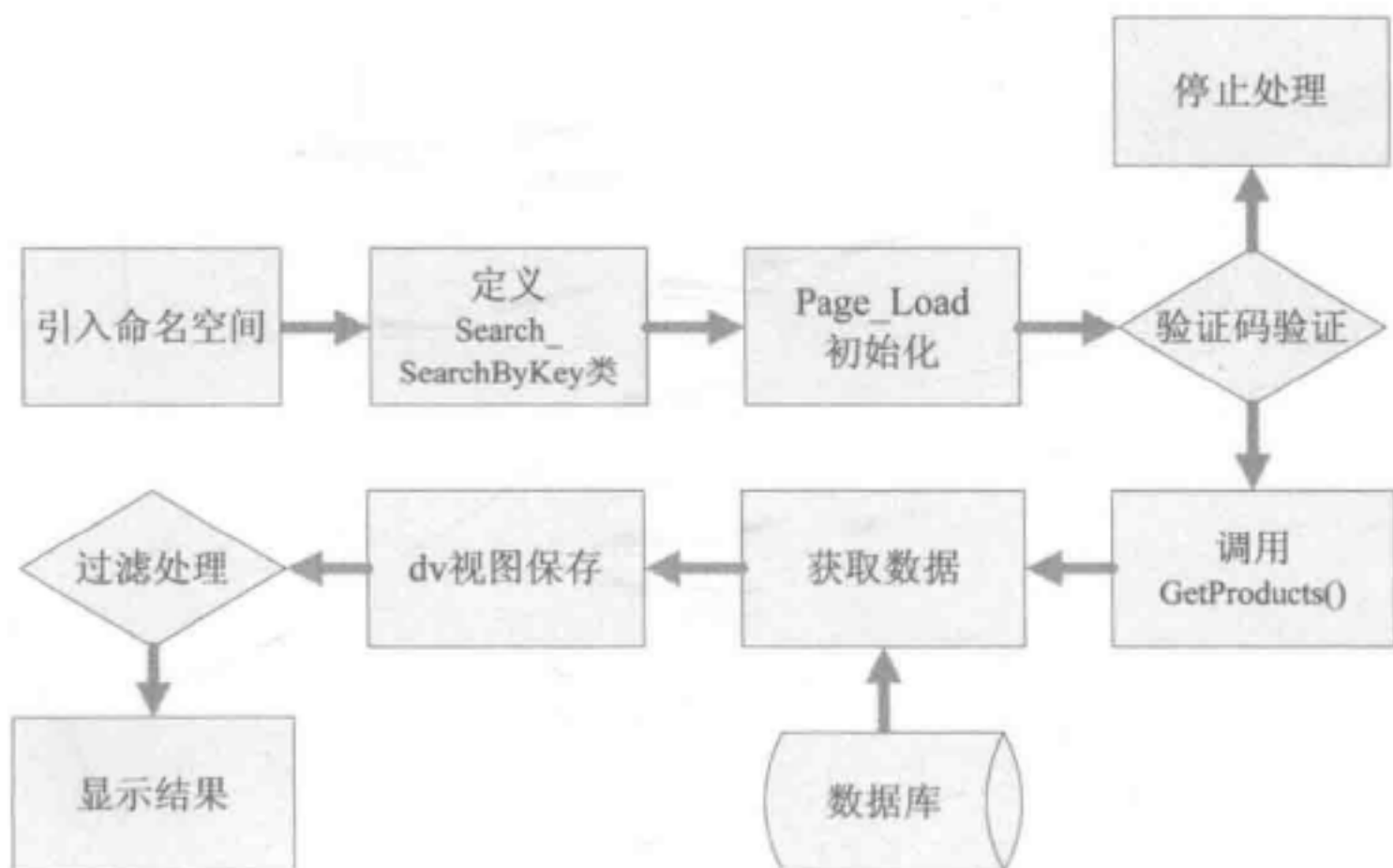


图 7-8 搜索表单页面的处理流程

上述过程的具体实现流程如下。

- (1) 引入命名空间，定义 Search\_SearchByKey 类。
- (2) 声明 Page\_Load 方法，进行页面初始化处理。
- (3) 验证码验证。

(4) 调用 BindPageData(int productID)方法, 进行搜索处理。BindPageData(int productID)函数的具体运行流程如下。

- ① 调用 GetProducts()方法, 获取所有商品的信息。将结果保存在对象 ds 中。
- ② 获取对象 ds 中 Tables[0]的视图, 并保存在视图 dv 中。
- ③ 设置 dv 视图过滤处理表达式。
- ④ 按照过滤语句获取对应的商品数据。

SearchByKey.aspx.cs 文件的具体实现代码如下所示:

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxEBusiness;
using ASPNETAJAXWeb.ValidateCode.Page;

public partial class Search_SearchByKey : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    private void BindPageData(string key)
    {
        ///获取所有商品的数据
        Product product = new Product();
        DataSet ds = product.GetProducts();
        if(ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        ///设置过滤表达式
        DataView dv = ds.Tables[0].DefaultView;
        dv.RowFilter = "Name LIKE '%" + key + "%'";
        gvProduct.DataSource = dv;
        gvProduct.DataBind();
    }

    protected void btnCommit_Click(object sender, EventArgs e)
    {
        ///判断是否创建了验证码
        if(Session[ValidateCode.VALIDATECODEKEY] == null) return;

        ///验证验证码是否相等
        if(tbCode.Text != Session[ValidateCode.VALIDATECODEKEY].ToString())
        {
            lbMessage.Text = "验证码输入错误, 请重新输入。";
            return;
        }

        BindPageData(tbName.Text);
    }
}
```

## 7.15 项目调试

视频讲解  光盘: 视频\第7章\项目调试.avi

将本项目命名为“shop”, 系统主页的显示效果如图 7-9 所示。





商品搜索界面的效果如图 7-12 所示。

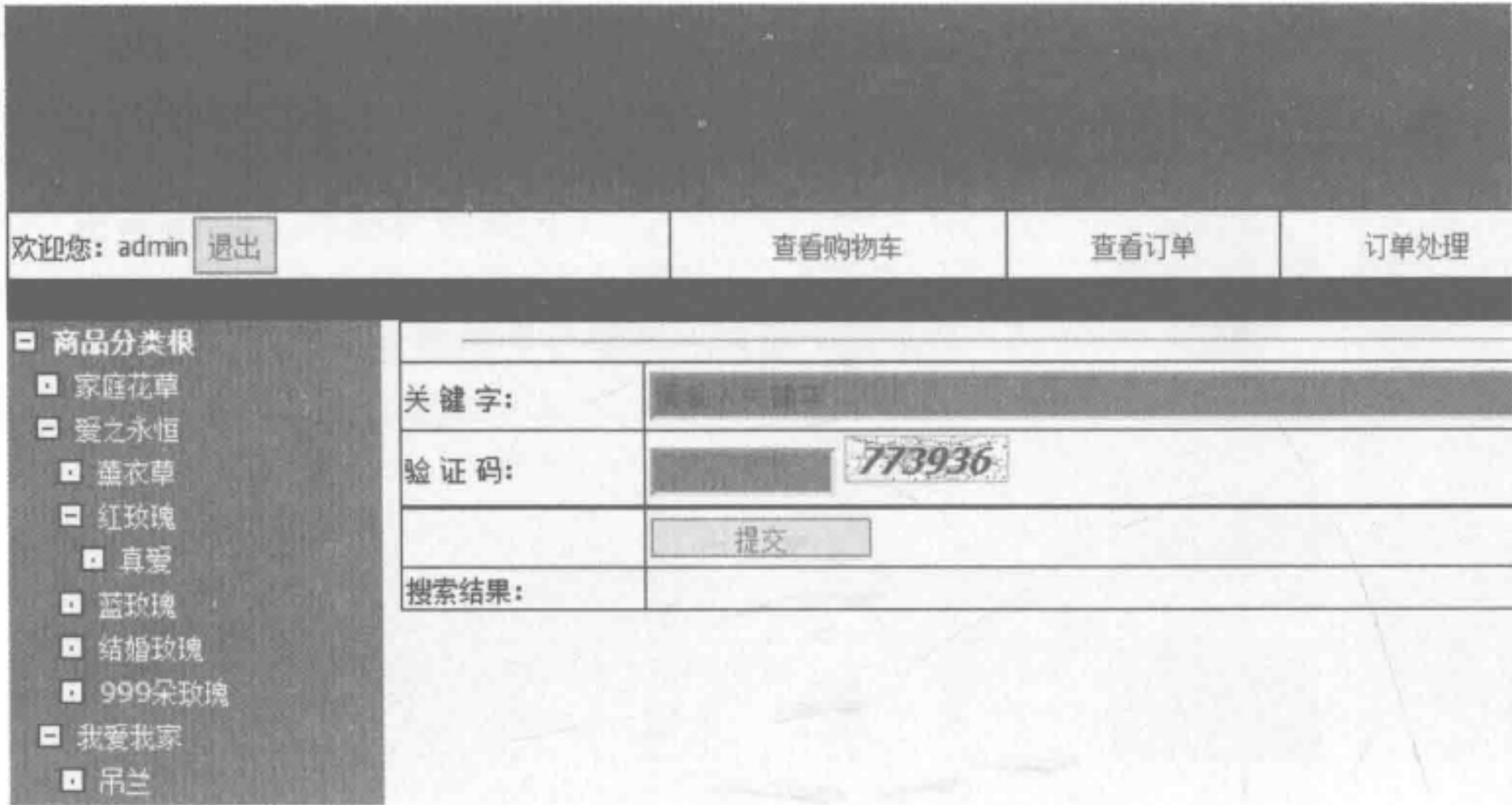


图 7-12 商品搜索界面的效果

商品评论界面的效果如图 7-13 所示。

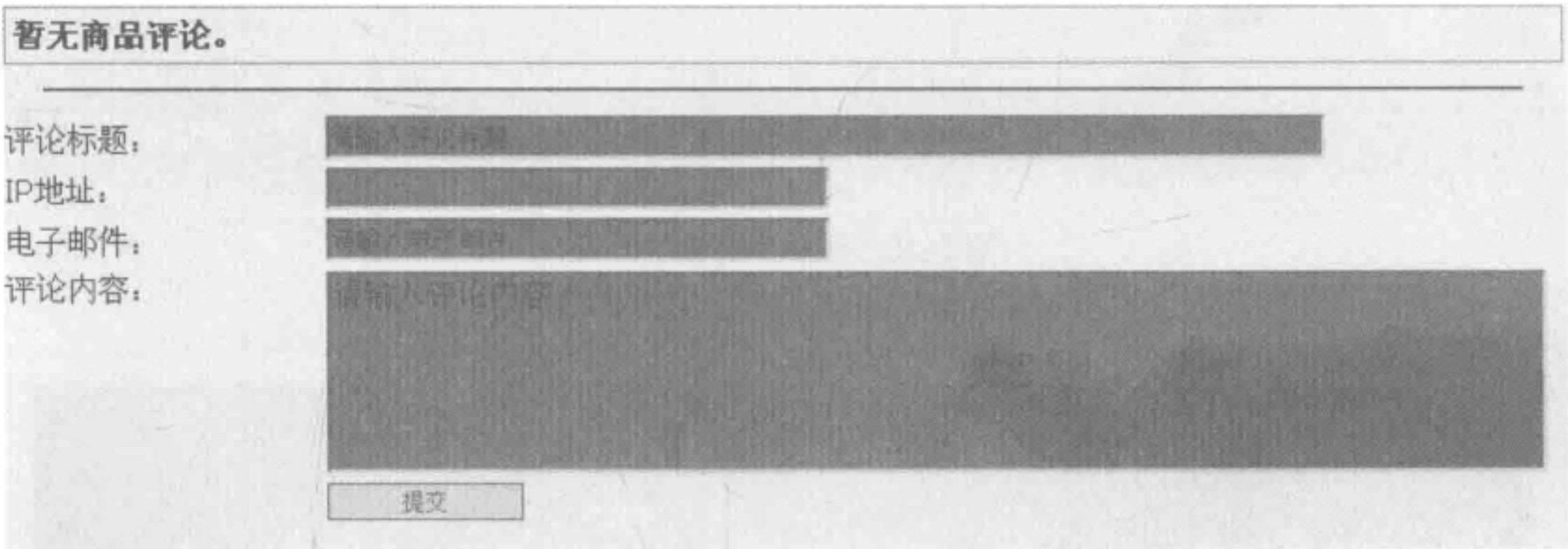


图 7-13 商品评论界面的效果

系统调试成功后，可以将系统文件发布到站点上，供互联网用户浏览。发布的过程很简单，基本实现流程如下。

- (1) 上传文件：使用专用的 FTP 上传工具(例如 CuteFtp)，将文件上传到远程服务器。
- (2) 系统测试：文件上传后，可以在浏览器中输入站点的 IP 来浏览系统效果。为提高站点的易记性，可以为其购买专业的域名。

## 第 8 章 企业交互系统

赢帆项目开发

本章首先对企业的实际需求进行调研和分析，以此为基础提出系统框架，并展开系统设计工作。

在系统设计环节，采用三层(数据层、逻辑层、界面层)松耦合的设计方式，从而提升系统开发和实现效率。在系统实现环节，从三个层次对实现流程进行细化，并据此选择开发语言和工具进行实现。

该系统现已被企业采用，用于部门上下级、员工间围绕本地环境问题的信息交互。经运行和测试，系统成功实现了框架中所涉及的功能模块和业务逻辑，大幅提升了部门内部的信息交互效率和工作效率。



### 赠送的超值电子书

- 071 构造函数概述
- 072 使用私有构造函数
- 073 使用静态构造函数
- 074 使用析构函数
- 075 使用只读字段
- 076 使用 this 关键字
- 077 属性
- 078 索引器
- 079 编译单元
- 080 完全限定名标识

## 8.1 程序的可移植性

视频讲解 光盘：视频\第8章\程序的可移植性.avi

程序可移植性是指将一种计算机上的软件放到其他计算机上运行的能力，也可以称作软件的“自动搬家”。软件移植是实现功能的等价联系，而不是等同联系。软件可移植性的主要标志是：这类软件有个通用的标准文本，它们独立于具体的计算机。开发者要想更加适应客户的各种需求，提高程序的可移植性是必不可少的工作。在本节的内容中，将引领读者一起探讨程序可移植性的奥秘。

### 8.1.1 什么是程序的可移植性

可移植性是指软件从某一环境转移到另一环境的难易程度。为获得较高的可移植性，在设计过程中，常采用通用的程序设计语言和运行支撑环境。尽量不使用与系统的底层相关性强的语言。

从狭义上讲，程序的可移植性，是指某个程序应当独立于计算机的硬件环境。从广义上讲，可移植程序还应独立于计算机的软件，即应该是高级的标准化的软件，它的功能与机器系统结构无关，可跨越很多机器界限。

从一种计算机向另一种计算机移植软件时，首先要考虑所移植的软件对宿主机硬件及操作系统的接口，然后设法用目标机的接口代换之。因此，接口的改造容易与否，是衡量一个软件可移植性高低的主要标志之一。

可移植性是软件质量之一，良好的可移植性可以提高软件的生命周期。可移植性是软件产品的一种能力属性。

编程语言编写的程序首先要被编译器编译成目标代码，然后在目标代码的前面插入启动代码，最终生成一个完整的程序。所以编程语言的可移植性依赖于它们的编译器是否强大，是否在多个平台上都有这种编程语言的编译器。例如，C编译器在大约40种系统上可用，包括从8位微处理器的计算机到Cray超级计算机。

在此要注意的是，程序中为访问特定设备(如显示器)或者操作系统(例如Windows XP的API)的特殊功能而专门编写的代码通常是不能移植的。

综上所述，一个编程语言的可移植性强不强，主要取决于如下两点：

- 不同平台编译器的数量。
- 对特殊硬件或操作系统的依赖性。

### 8.1.2 赢在项目——实现跨开发平台的转换

对于开发人员来说，只要严格遵循C#语言规范来开发程序，就无须考虑太多程序可移植性的问题。要想使自己的程序能够应付挑剔的客户，首先需要确保自己的程序能够在不同的开发平台中运行。

在当今C#语言的程序开发中，最主流的开发平台是Visual Studio .NET。



随着 Visual Studio .NET 开发工具的发展,联合使用第三方插件可以轻松实现跨平台操作。例如,利用 Visual Studio .NET,可以开发 Linux 上的应用程序,可以将 ASP.NET 程序运行在 Linux 平台的 Apache 服务器上,可以让 ASP.NET 页面访问 J2EE 中的组件(如 EJB 等),这些对于以往的程序开发者而言,听起来像是天方夜谭、痴人说梦。Windows 平台和 Linux 平台,Java 平台和 .NET 平台这两对竞争对手在程序开发者眼里似乎永远只是平行线,没有交集。然而,自从 Visual Studio 2010 发布后,这些不可能开始变成了现实。当然,除此之外,Visual Studio 2010 下的 Team Explore Everywhere 对 Java 和 Eclipse 的开发也是不可小觑的。

### (1) TFS 与 Team Explorer Everywhere 跨平台

Team Explorer Everywhere 是 Eclipse 的 Plug-in 元件,让 Eclipse 能够使用 TFS(Team Foundation Server)的版本控制、工作项目(Work Item)、Team Build 以及报表等功能,它提供类似 Team Explorer、Pending Changes 等视窗,操作起来更加便捷。

Cross-Platform Command-Line 运行在基于 Unix 的系统中,很多情况下没有 X-Window,即没有图形化用户界面,唯一方法是只能通过命令进行操作,这个工具可以让我们通过指令进行程序代码版本的管控和 Team Build 等工作。

当然,在安装方式上也很简单,只要将 Team Explorer Everywhere 解压,将解压后的 Features 与 plugins 两个目录复制到 Eclipse 安装目录中即可。当 Eclipse 启动后,再选择 Menu → Windows → Other Perspective → Other → Team Explorer,即可显示。

### (2) Grasshopper 工具跨平台

从 Visual Studio 2010 工具开始,就可以利用 Grasshopper 工具来开发在 Linux 下运行的 Web 应用程序了,还可以利用 Visual MainWin for J2EE Developer Edition 来进行 Java 下的开发工作,这两者设计的目的,就是在 Visual Studio 开发工具和 Linux、Java EE 之间架起一座桥梁。使 Visual Studio 2010 的跨平台作业得以实现。

## 8.2 新的挑战

视频讲解  光盘: 视频\第8章\新的挑战.avi

本项目的客户是一家外资公司,为了提高办公效率,实现无纸化办公,准备做一个企业内部的 OA 交互系统。客户提出了如下所示的两点要求。

- (1) 公司内部人员可以自由组队,组成一个团体,在团体内交流。
- (2) 如果不能在线交流,可以给对方留言。

本项目的团队成员如下所示。

- 项目经理: 负责前期功能分析、策划,构建系统模块,检查项目进度,质量检查。
- 软件工程师 A: 配置系统文件,搭建数据库,实现数据访问层。
- 软件工程师 B: 负责登录验证模块、客户分组模块的编码工作。
- 软件工程师 C: 负责团队处理模块和在线交互模块的编码工作。
- 软件工程师 D: 系统测试和站点发布。

本项目的具体实现流程如图 8-1 所示。

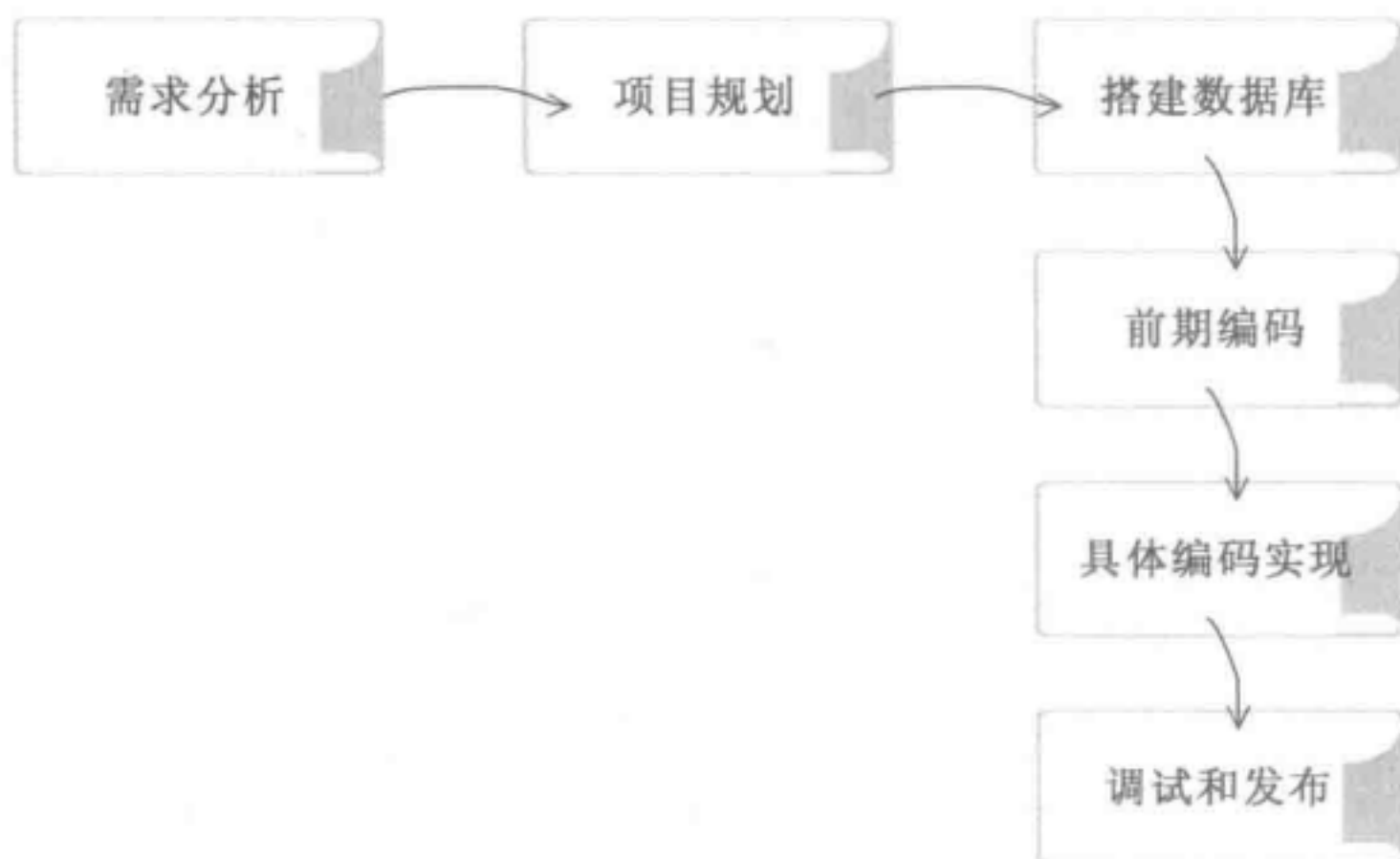


图 8-1 开发流程

## 8.3 项目规划和分析

视频讲解 光盘：视频\第 8 章\项目规划和分析.avi

软件项目规划的目的，是预测未来，确定要达到的目标，估计会遇到的问题，并提出实现目标和解决问题的有效方案、方针、措施和手段的过程。在本节的内容中，将详细讲解本章项目规划和分析的具体过程。

### 8.3.1 在线交互系统的背景

随着经济社会的不断发展，企业信息化水平在不断地提高。良好的企业信息化建设不仅能够改变企业商业模式、经营模式，降低经营成本，同时也能够帮助企业优化管理、提高工作效率。随着电子计算技术和互联网技术的发展，企业内部信息交互方式和手段不断发生新的变革。以互联网技术为基础的信息交互方式代替传统交互方式成为企业改善管理、提高工作效率的利器。但随着企业不断发展和壮大，企业内部信息交互的需求也在不断变化，现有的交互方式已经不能充分满足当前的需求。新型的基于 Web 的信息交互系统为解决该问题提供了新思路。这里从企业的实际需求出发，提出并实现了一种基于 Web 的企业信息交互系统。

### 8.3.2 企业在线交互系统的构成模块

一个典型的企业在线交互系统的构成模块如下。

#### (1) 登录验证模块

为了确保系统的安全，防止非法用户和竞争对手进入系统，在系统中专门设立了登录验证模块。

#### (2) 用户信息分类显示模块

为便于快速实现对不同用户的交互，对系统内的用户进行了细分，方便用户的选择性

交互。例如，在系统中设置了重要客户、一般客户和合作伙伴等不同种类的群体。

### (3) 信息显示模块

为了方便系统用户间的相互了解，系统设立了用户信息详情显示模块，供用户浏览系统内各用户的详细信息。

### (4) 用户检索模块

为方便用户迅速找到自己的目标交流对象，系统设置了信息检索模块，用户可以根据用户的基本信息快速地找到自己的交流目标。

### (5) 团队处理模块

为了便于企业对不同部门或不同工作目标的区分，系统设置了团队处理模块，可以将不同种类的用户加入到各自的团队中，从而发挥集体优势，创造出更高的效益。

### (6) 在线交流模块

在线交流模块是整个系统的核心，系统用户可以与系统内的其他用户进行在线及时交互，进一步实现办公自动化。

上述应用模块的具体运行流程如图 8-2 所示。

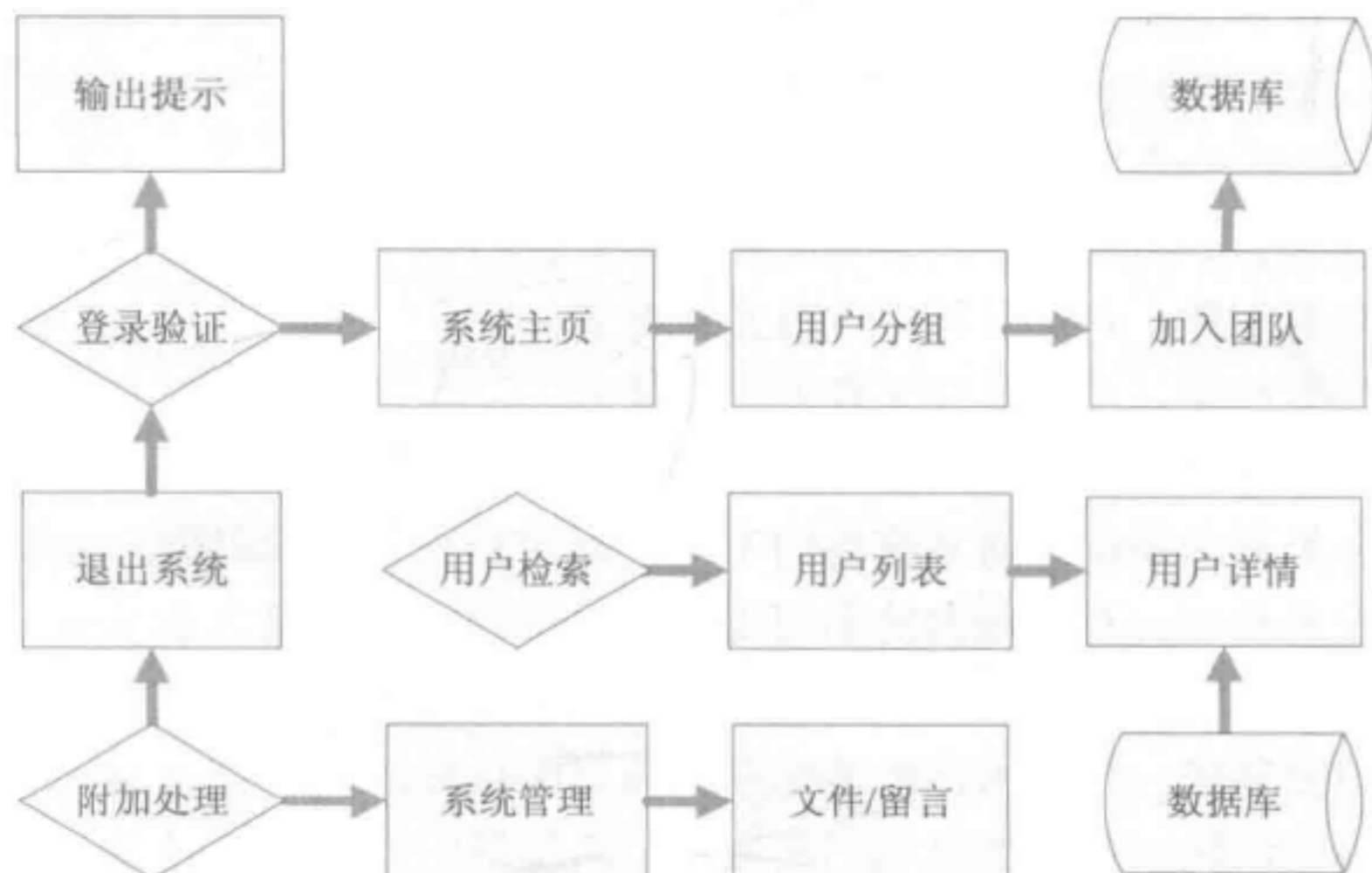


图 8-2 在线交互系统的运行流程

在企业在线交互系统中，最为核心的功能模块是在线信息的交互处理。在现实的 Web 开发中，在线信息交互功能通常使用在线交互系统来代替。所以，只需读者掌握了本书前面介绍的在线交互系统，对于本章的知识，相信读者也会轻松上手。

至此，整个项目的第一阶段完成。在这一阶段，深刻体会到了团队精神在软件开发团队中的重要性，下面是一些关于团队合作的观点。

### (1) 作为一个领导者

领导者是团队的核心，是从全局角度把握整个团队方向的人。作为一个领导者，虽然其权威级别高一些，不过学会熟练地与别人一起完成更多的工作，应该为提升自身价值所能做的最重要的事情。应该做到：

- 分工明确但不呆板。
- 加强团队成员的日常交流。



- 说话时多使用“我们”。
- 让每个人感觉到自己很重要。

## (2) 作为团队成员

每个团队成员都是不可或缺的，每一个团队成员都要具有团队合作的意识。无论自身能力多么强大，团队中少了某个成员却依然会继续运行，所以不要妄自尊大。应该做到：

- 做好自己的事情。
- 信任你的伙伴，即信任其他团队成员。
- 为他人着想，不要事事都从自己的角度考虑。
- 愿意多付出。

## 8.4 规划项目文件

视频讲解 光盘：视频\第8章\规划项目文件.avi

(1) 新建文件夹 kehu 和 data，来保存项目的实现文件，各个文件夹的具体说明如下。

- kehu：保存系统的项目文件。
- data：保存系统的数据库文件。

(2) 预先规划各个构成模块的实现文件，具体说明如下。

- 系统配置文件：对项目程序进行总体配置。
- 样式设置模块：设置系统文件的显示样式。
- 数据库文件：搭建系统数据库平台，保存系统的登录数据。
- 用户分类显示模块：将系统内的不同类别用户列表显示出来。
- 团队处理模块：对系统内的不同用户群体进行团队处理。
- 在线交流模块：实现系统内用户的在线交互处理。
- 用户检索模块：帮助用户迅速检索到自己的目标用户。
- 系统管理模块：帮助当前用户实现对个人信息的管理维护。
- 在线留言模块：使当前用户实现向目标用户的留言发布功能。
- 文件处理模块：使当前用户实现向目标用户的在线文件处理。

## 8.5 系统配置文件

视频讲解 光盘：视频\第8章\系统配置文件.avi

因为客户要求使用 SQL Server 2008 数据库存储数据，并用 Ajax 技术实现无刷新处理，所以需要根据用户的需求来编写配置文件 Web.config，主要功能是设置数据库的连接参数，并配置系统和 Ajax 服务器的相关内容。

在动态 Web 开发应用中，Ajax 技术的主要特点如下所示。

(1) 页面独立性。传统的 Web 应用程序一般由多个页面构成，偕同完成特定的处理功能。而对于一个典型的 Ajax 应用程序来说，用户无须在不同的页面中切换，只要停留在一

个页面中，由 XMLHttpRequest 对象从服务器取得数据，然后由 JavaScript 操作页面上的元素并更新其中的内容即可。

(2) 符合标准性。作为 Ajax 技术的核心，W3C 正在对 XMLHttpRequest 的规范进行标准化处理，XMLHttpRequest 成为标准已经指日可待。而在 Ajax 领域所使用的其他组成技术，包括 JavaScript、XML、CSS 和 DOM 等，均早已成为标准，并被所有的主流浏览器所实现。这样，典型的 Ajax 应用程序无需客户端进行任何形式的安装部署，即可兼容地运行于每一个主流浏览器上。

(3) 能够获取服务器数据。可以灵活更新页面内的指定内容，而不需要刷新整个页面。

(4) 页面与服务器间的数据交互可以通过异步传输来实现，而不需要中断用户当前的操作。

(5) 减少了页面和服务器间的数据传输数量，从而大大提高了应用程序的处理效率。

在本节的内容中，将分别对上述功能的实现进行详细说明。

### 1. 配置连接字符串参数

配置连接字符串参数即设置系统程序连接数据库的参数，对应的实现代码如下所示：

```
<connectionStrings>
  <add name="SQLCONNECTIONSTRING"
    connectionString="data source=GUAN\AAA;user id=sa;pwd=8888888;database=kehu"
    providerName="System.Data.SqlClient"/>
</connectionStrings>
```

其中，source 设置连接的数据库服务器；user id 和 pwd 分别指定数据库的登录名和密码；database 设置连接数据库的名称。

### 2. 配置 Ajax 服务器参数

配置 Ajax 服务器参数即配置 Ajax Control Toolkit 程序集参数，为 AjaxControlToolkit.dll 程序集提供了一个前缀字符串“AjaxControlToolkit”。

这样，系统页面在引用 AjaxControlToolkit.dll 中的控件时，就不再需要额外地添加 <Register> 代码了。上述功能在 <controls> 元素内的对应实现代码如下所示：

```
<pages>
  <controls>
    <add namespace="AjaxControlToolkit" assembly="AjaxControlToolkit"
      tagPrefix="ajaxToolkit"/>
    <add tagPrefix="asp" namespace="System.Web.UI" assembly="System.Web.Extensions,
      Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
  </controls>
</pages>
```

## 8.6 搭建数据库

视频讲解 光盘：视频\第 8 章\搭建数据库.avi

数据库设计是总体设计中一个重要的环节，良好的数据库设计可以简化开发过程，提高系统的性能，使系统功能更加明确。

一个好的数据库结构可以使系统处理速度快、占用空间小、操作处理过程简单、容易查找等。数据库结构的变化会造成编码的改动，所以在编码之前，一定要认真设计好数据库，避免无谓的工作。

在设计数据库的过程中，必须避免后期随着项目的升级出现为数据库设计打补丁的情况，所以需要遵循如下 3 个原则。

(1) 数据库中表的个数越少越好。只有表的个数少了，才能说明系统的 E-R 图少而精，去掉了重复的多余的实体，形成了对客观世界的高度抽象，进行了系统的数据集成，防止了打补丁式的设计。

(2) 表中组合主键的字段个数越少越好。因为主键的作用，一是建主键索引，二是作为子表的外键，所以组合主键的字段个数少了，不仅节省了运行时间，而且节省了索引存储空间。

(3) 一个表中的字段个数越少越好。只有字段的个数少了，才能说明在系统中不存在数据重复，且很少有数据冗余，更重要的，是督促读者学会“列变行”，这样就防止了将子表中的字段拉入到主表中去，在主表中留下许多空余的字段。

所谓“列变行”，就是将主表中的一部分内容拉出去，另外单独建一个子表。这个方法很简单，有的人就是不习惯、不采纳、不执行。

### 8.6.1 数据库设计

本系统实例采用 SQL Server 2008 数据库，名为“kehu”。kehu 库内各表的具体结构分别如下所示。

(1) 表 Caboodle 的具体设计结构如表 8-1 所示。

表 8-1 系统团队信息表(Caboodle)

| 字段名称   | 数据类型          | 是否主键 | 默认值  | 功能描述   |
|--------|---------------|------|------|--------|
| ID     | int           | 是    | 递增 1 | 编号     |
| Name   | varchar(50)   | 否    | Null | 名称     |
| UserID | int           | 否    | Null | 创建用户编号 |
| Remark | varchar(1000) | 否    | Null | 简介     |

(2) 表 CaboodleUser 的具体设计结构如表 8-2 所示。

表 8-2 系统团队关联信息表(CaboodleUser)

| 字段名称       | 数据类型 | 是否主键 | 默认值  | 功能描述   |
|------------|------|------|------|--------|
| CaboodleID | int  | 是    | 递增 1 | 所属团队编号 |
| UserID     | int  | 否    | Null | 所属用户编号 |
| RoleID     | int  | 否    | Null | 角色     |

(3) 表 File 的具体设计结构如表 8-3 所示。



表 8-3 系统发送文件信息表(File)

| 字段名称       | 数据类型         | 是否主键 | 默 认 值 | 功能描述 |
|------------|--------------|------|-------|------|
| ID         | int          | 是    | 递增 1  | 编号   |
| Name       | varchar(200) | 否    | Null  | 文件名  |
| Sender     | int          | 否    | Null  | 发送者  |
| Receiver   | int          | 否    | Null  | 接收者  |
| Url        | varchar(255) | 否    | Null  | 文件地址 |
| Type       | varchar(50)  | 否    | Null  | 类型   |
| Size       | int          | 否    | Null  | 大小   |
| CreateDate | datetime     | 否    | Null  | 事件   |

(4) 表 Group 的具体设计结构如表 8-4 所示。

表 8-4 系统用户分组信息表(Group)

| 字段名称   | 数据类型        | 是否主键 | 默 认 值 | 功能描述 |
|--------|-------------|------|-------|------|
| ID     | int         | 是    | 递增 1  | 编号   |
| Name   | varchar(20) | 否    | Null  | 名称   |
| UserID | int         | 否    | Null  | 所属编号 |

(5) 表 GroupUser 的具体设计结构如表 8-5 所示。

表 8-5 用户分组关系信息表(GroupUser)

| 字段名称    | 数据类型 | 是否主键 | 默 认 值 | 功能描述 |
|---------|------|------|-------|------|
| ID      | int  | 是    | 递增 1  | 编号   |
| GroupID | int  | 否    | Null  | 组编号  |
| UserID  | int  | 否    | Null  | 用户编号 |

(6) 表 Leaveword 的具体设计结构如表 8-6 所示。

表 8-6 系统留言信息表(Leaveword)

| 字段名称       | 数据类型         | 是否主键 | 默 认 值 | 功能描述 |
|------------|--------------|------|-------|------|
| ID         | int          | 是    | 递增 1  | 编号   |
| Body       | varchar(100) | 否    | Null  | 内容   |
| Sender     | int          | 否    | Null  | 发送者  |
| Receiver   | int          | 否    | Null  | 接收者  |
| CreateDate | datetime     | 否    | Null  | 时间   |
| Status     | tinyint      | 否    | Null  | 状态   |

(7) 表 MessageForCaboodle 的具体设计结构如表 8-7 所示。

表 8-7 团队交互信息表(MessageForCaboodle)

| 字段名称       | 数据类型          | 是否主键 | 默 认 值 | 功能描述 |
|------------|---------------|------|-------|------|
| ID         | int           | 是    | 递增 1  | 编号   |
| Body       | varchar(1000) | 否    | Null  | 内容   |
| Sender     | int           | 否    | Null  | 发送者  |
| CaboodleID | int           | 否    | Null  | 群编号  |
| CreateDate | datetime      | 否    | Null  | 事件   |

(8) 表 MessageForSingle 的具体设计结构如表 8-8 所示。

表 8-8 用户交互信息表(MessageForSingle)

| 字段名称       | 数据类型          | 是否主键 | 默 认 值 | 功能描述 |
|------------|---------------|------|-------|------|
| ID         | int           | 是    | 递增 1  | 编号   |
| Body       | varchar(1000) | 否    | Null  | 内容   |
| Sender     | int           | 否    | Null  | 发送者  |
| Receiver   | int           | 否    | Null  | 接收者  |
| CreateDate | datetime      | 否    | Null  | 事件   |

(9) 表 Role 的具体设计结构如表 8-9 所示。

表 8-9 系统用户角色信息表(Role)

| 字段名称   | 数据类型         | 是否主键 | 默 认 值 | 功能描述 |
|--------|--------------|------|-------|------|
| ID     | int          | 是    | 递增 1  | 编号   |
| Name   | varchar(50)  | 否    | Null  | 名    |
| Remark | varchar(100) | 否    | Null  | 说明   |

(10) 表 User 的具体设计结构如表 8-10 所示。

表 8-10 系统用户信息表(User)

| 字段名称         | 数据类型          | 是否主键 | 默 认 值 | 功能描述 |
|--------------|---------------|------|-------|------|
| ID           | int           | 是    | 递增 1  | 编号   |
| Username     | varchar(50)   | 否    | Null  | 用户名  |
| Aliasname    | varchar(50)   | 否    | Null  | 别名   |
| Password     | varchar(255)  | 否    | Null  | 密码   |
| UserIdentity | varchar(10)   | 否    | Null  | 标识   |
| CreateDate   | datetime      | 否    | Null  | 时间   |
| Email        | varchar(255)  | 否    | Null  | 邮箱   |
| PictureUrl   | varchar(255)  | 否    | Null  | 图片   |
| Signing      | varchar(1000) | 否    | Null  | 签名   |

(11) 表 UserLogin 的具体设计结构如表 8-11 所示。

表 8-11 系统用户登录信息表(UserLogin)

| 字段名称       | 数据类型     | 是否主键 | 默 认 值 | 功能描述 |
|------------|----------|------|-------|------|
| ID         | int      | 是    | 递增 1  | 编号   |
| UserID     | int      | 否    | Null  | 用户编号 |
| LoginDate  | datetime | 否    | Null  | 登录时间 |
| LogoffDate | datetime | 否    | Null  | 退出时间 |

在上述数据库中，表 GroupUser 表 CaboodleUser 设计得很科学，其作用在整个系统中十分重要，它们关联了系统的用户信息和团队信息，在具体的数据库操作过程中，起到了桥梁的作用，是一种关系表。

8.6.2 系统参数设置文件

系统参数设置功能是由文件 Global.asax 实现的，功能是定义页面载入、结束和错误初始化，并保存系统的登录数据，实现用户的登录和退出处理。具体的实现代码如下所示：

```
<%@ Application Language="C#" %>
<%@ Import Namespace="System.Collections.Generic" %>
<%@ Import Namespace="ASPNETAJAXWeb.AjaxInstantMessaging" %>
<script runat="server">
    /// 保存登录用户的列表
    public static List<UserInfo> Users = new List<UserInfo>();
    void Application_Start(object sender, EventArgs e)
    {
        Users.Clear(); ///登录用户列表初始化
    }
    void Application_End(object sender, EventArgs e)
    {
        /// 在应用程序关闭时的处理
    }
    void Application_Error(object sender, EventArgs e)
    {
        /// 在出现未处理的错误时处理
    }
    void Session_Start(object sender, EventArgs e) { }
    void Session_End(object sender, EventArgs e)
    {
        if(Session["UserID"] != null) ///用户离开时处理
        {
            string userID = Session["UserID"].ToString();
            foreach(UserInfo ui in Users)
            {
                if(ui.UserID.ToString() == userID) ///根据用户 ID 找到离开的用户
                {
                    Users.Remove(ui);
                    break;
                }
            }
        }
    }
}
</script>
```



通过上述代码可知，每当接手一个新项目时，都要分析项目中的核心模块和具体实现方法。作为一个内部交互项目，核心功能很明确：用户登录验证、团队交互、信息传递。这些功能用对应的函数即可实现，只要实现了上述功能，整个项目也就大体上完成了。

## 8.7 数据访问层

视频讲解 光盘：视频\第8章\数据访问层.avi

因为对方是我们公司的第一个外企客户，所以在本阶段一定要做好系统可扩展性的准备工作，避免出现其他要求措手不及。在此需要特别注意的是，本系统应该提供登录验证模块，这样可以保证只有企业内部的合法用户才能登录这个系统，提高系统的安全性。

当使用 ASP.NET 进行动态 Web 开发时，三层架构是最佳的开发模式。三层架构包含表示层(USL)、业务逻辑层(BLL)、数据访问层(DAL)。

(1) 数据访问层：主要是对原始数据(数据库或者文本文件等存放数据的形式)的操作层，而不是指原始数据，也就是说，是对数据的操作，而不是数据库。具体为业务逻辑层或者表示层提供数据服务。

(2) 业务逻辑层：主要是针对具体问题的操作，也可以理解成对数据层的操作，对数据业务逻辑处理。如果说数据层是积木，那逻辑层就是对这些积木的搭建。

(3) 表示层：主要表示 Web 方式，也可以表示成 WinForm 方式，Web 方式也可以表现成 aspx，如果逻辑层相当强大和完善，无论表示层如何定义和更改，逻辑层都能完善地提供服务。

对于很多初学者来说，最大的困惑是不知当前工作中哪些属于数据访问层，哪些属于逻辑层。其实，辨别的方法很简单。

- 数据访问层：主要看你的数据层里面有没有包含逻辑处理。实际上，数据访问层的各个函数主要完成各个对数据文件的操作，而不必管其他操作。
- 业务逻辑层：主要负责对数据层的操作，也就是说对一些数据层的操作进行组合。
- 表示层：是对用户的请求接受，以及数据的返回，为客户端提供应用程序的访问。

完善的三层结构的要求是：修改表现层而不用修改逻辑层，修改逻辑层而不用修改数据层。否则，你的应用是不是多层结构，或者说是层结构的划分和组织上是不是有问题，就很难说了。

作为整个项目的核心和难点，本项目的数据访问层分为如下所示的 3 个部分。

- (1) 登录验证。
- (2) 客户分组。
- (3) 团队管理。

### 8.7.1 数据访问层——用户登录验证

本功能模块的数据访问层功能是由文件 User.cs 实现的。

该文件的主要功能，是在 ASPNETAJAXWeb.AjaxInstantMessaging 空间内，建立 UserInfo 类和 User 类，并定义多个方法，实现对数据库中用户数据的处理。

在文件 User.cs 中，与用户登录验证模块相关的方法如下：

- GetUserLogin(string username, string password)。
- AddUserLogin(int userID)。
- UpdateUserLogoff(int loginID)。

上述方法的运行流程如图 8-3 所示。

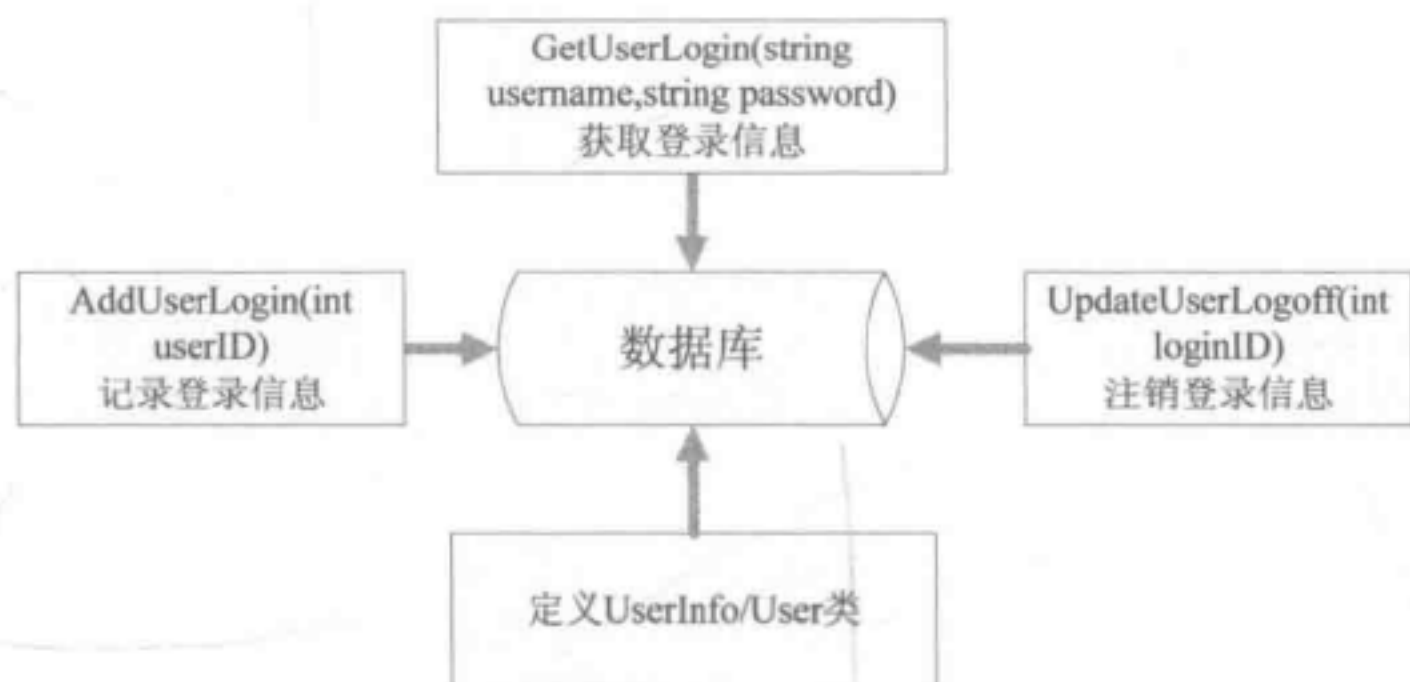


图 8-3 登录验证模块数据访问层的运行流程

## 1. 定义类

定义 UserInfo 类和 User 类的主要实现代码如下所示：

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;
namespace ASPNETAJAXWeb.AjaxInstantMessaging
{
    /// 保存用户登录信息的类
    public class UserInfo
    {
        private int userID;
        private int caboodleID = -1;
        private string username;
        ...
    }
    public class User
    {
        public User()
        {
        }
    }
}
```

## 2. 获取登录信息

获取登录信息即获取登录用户的用户名和密码信息，实现用户的登录。该功能是由方法 GetUserLogin(string username, string password)实现的，其具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数，并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象，实现数据库连接。
- (3) 新建 SQL 查询语句，获取数据库内此登录数据的用户登录信息。
- (4) 创建获取数据的对象 cmd。
- (5) 打开数据库连接，获取数据，将获取的数据保存在 dr 中。



(6) 操作成功, 返回 dr。

上述功能的对应实现代码如下所示:

```
public SqlDataReader GetUserLogin(string username, string password)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText =
        "SELECT ID FROM [User] WHERE Username=@Username AND Password=@Password";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@Username", SqlDbType.VarChar, 50);
    cmd.Parameters.Add("@Password", SqlDbType.VarChar, 255);
    cmd.Parameters[0].Value = username;
    cmd.Parameters[1].Value = password;
    ///定义 SqlDataReader
    SqlDataReader dr;
    try
    {
        con.Open();
        ///读取数据
        dr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    }
    catch (Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    return dr;
}
```

### 3. 添加登录信息

添加登录信息即向系统库内添加新登录用户的用户名和密码信息, 上述功能是由方法 AddUserLogin(int userID)实现的, 具体的实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。

(2) 使用该连接字符串创建 con 对象, 实现数据库连接。

(3) 新建 SQL 插入语句, 向系统库内添加此登录用户的登录数据。

(4) 创建添加数据的对象 cmd。

(5) 打开数据库连接执行插入操作, 将处理后的结果保存在 result 中。

(6) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```
public int AddUserLogin(int userID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "INSERT INTO [UserLogin] (UserID, LoginDate, LogoffDate)
        VALUES (@UserID, GETDATE(), GETDATE()) RETURN @@Identity";
```



```

///创建 SqlCommand
SqlCommand cmd = new SqlCommand(cmdText, con);
///创建参数并赋值
cmd.Parameters.Add("@UserID", SqlDbType.Int, 4);
cmd.Parameters.Add("RETURNVALUE", SqlDbType.Int, 4);
cmd.Parameters[0].Value = userID;
cmd.Parameters[1].Direction = ParameterDirection.ReturnValue;
int result = -1;
try
{
    con.Open();
    ///操作数据
    result = cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    throw new Exception(ex.Message, ex);
}
finally
{
    con.Close();
}
///返回登录的 ID 值
return (int)cmd.Parameters[1].Value;
}

```

#### 4. 注销登录信息

注销登录信息就是把保存的当前登录信息从系统中注销。

该功能是由方法 UpdateUserLogoff(int loginID)实现的, 其具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 字符串中。
- (2) 使用连接字符串创建 con 对象, 实现数据库连接。
- (3) 新建 SQL 更新语句, 对系统数据库内此登录用户的状态进行修改。
- (4) 创建修改数据的对象 cmd。
- (5) 打开数据库连接, 执行更新操作, 将处理后结果保存在 result 中。
- (6) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```

public int UpdateUserLogoff(int loginID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "UPDATE [UserLogin] SET LogoffDate=GETDATE() WHERE ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = loginID;
    int result = -1;
    try
    {
        con.Open();

```

```
    ///操作数据
    result = cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    throw new Exception(ex.Message, ex);
}
finally
{
    con.Close();
}
return result;
}
```

## 8.7.2 数据访问层——客户分组

本模块的数据访问层功能是由文件 Group.cs 实现的。

该文件的主要功能，是在 ASPNETAJAXWeb.AjaxInstantMessaging 空间内建立 Group 类和 GroupUser 类，并定义多个方法，实现对数据库中用户数据的处理。在文件 Group.cs 中，与用户登录验证模块相关的方法如下：

- GetGroupByUser(int userID)。
- GetSingleGroup(int groupID)。
- AddGroup(string name, int userID)。
- UpdateGroup(int groupID, string name)。
- DeleteGroup(int groupID)。
- GetUserbyGroup(int groupID)。
- AddGroupUser(int groupID, int userID)。
- UpdateGroupUser(int oldGroupID, int newGroupID, int userID)。
- DeleteGroupUser(int groupID, int userID)。

其中，上述前 5 种方法是 Group 类实现的，具体的运行流程如图 8-4 所示。

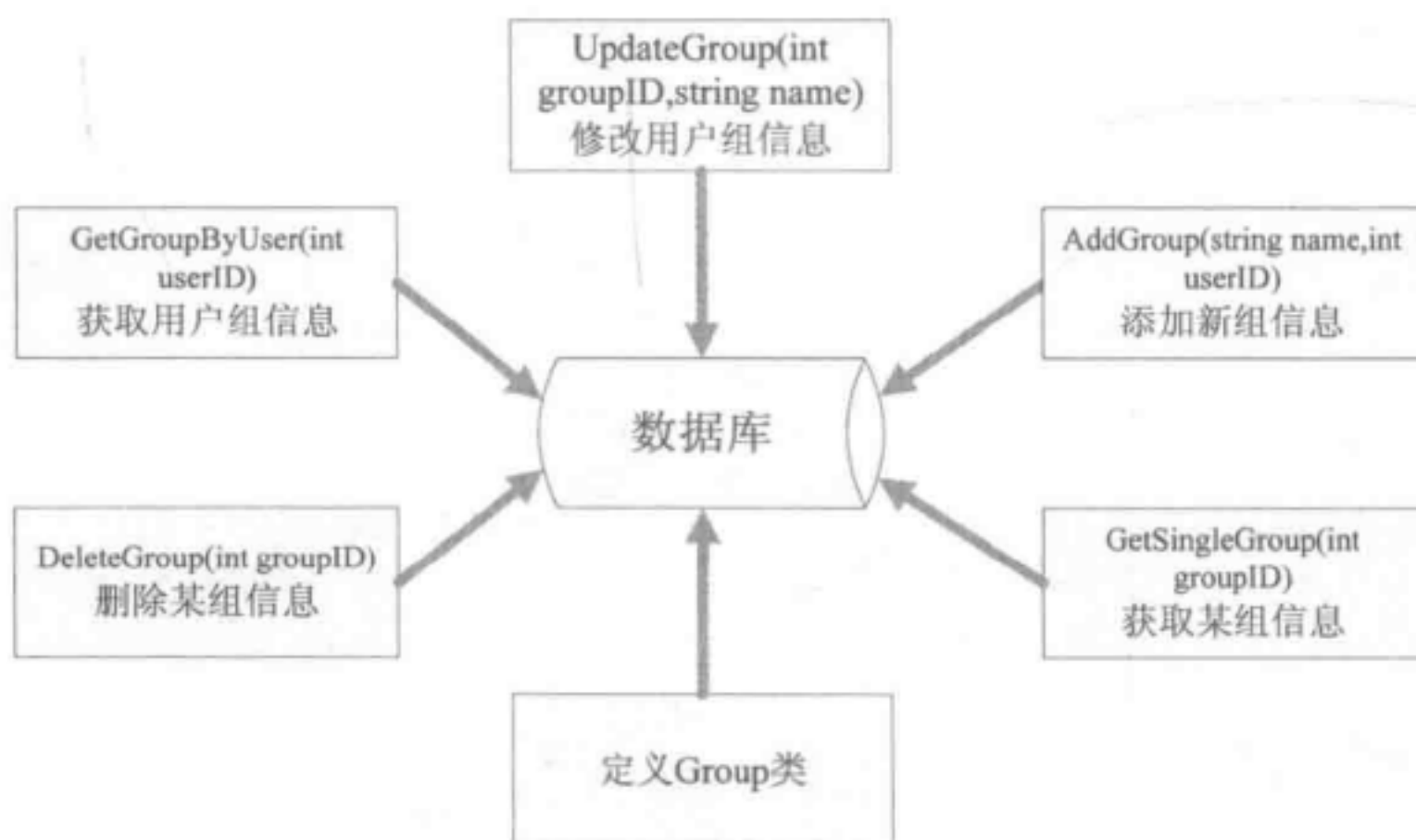


图 8-4 Group 类数据访问层的运行流程

下面将分别介绍上述 Group 类各方法的实现流程。

## 1. 定义 Group 类

定义 Group 类的主要实现代码，如下所示：

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;
namespace ASPNETAJAXWeb.AjaxInstantMessaging
{
    public class Group
    {
        public Group()
        {
        }
    }
}
```

## 2. 获取用户组信息

获取用户组信息就是获取某用户所属的分类组信息。

该功能是由方法 GetGroupByUser(int userID)实现的，具体的实现流程如下。

(1) 从系统配置文件 Web.config 内获取数据库连接参数，并将其保存在 connectionString 字符串中。

(2) 使用该连接字符串创建 con 对象，实现数据库连接。

(3) 新建 SQL 查询语句，获取库内此登录用户所属的用户组信息。

(4) 创建获取数据的对象 da。

(5) 打开数据库连接，获取数据，将获取的数据保存在 ds 中。

(6) 操作成功，返回 ds。

上述功能的对应实现代码如下所示：

```
public DataSet GetGroupByUser(int userID)
{
    ///获取连接字符串
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT G.*, [User].Username, (SELECT COUNT(*) FROM GroupUser WHERE
GroupID=G.ID AND GroupID > 3) AS GroupUserCount FROM [Group] AS G INNER JOIN [User]
ON G.UserID=[User].ID WHERE G.UserID=@UserID OR UserID=0";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@UserID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = userID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        ///打开连接
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch (Exception ex)
```



```

{
    ///抛出异常
    throw new Exception(ex.Message, ex);
}
finally
{
    ///关闭连接
    con.Close();
}
return ds;
}

```

### 3. 获取某组信息

获取某组信息，就是获取系统库内指定编号用户组的信息。

该功能是由方法 `GetSingleGroup(int groupID)` 实现的，其具体实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。
- (2) 使用该连接字符串创建 `con` 对象，实现数据库连接。
- (3) 新建 SQL 查询语句，获取数据库内指定编号的用户组信息。
- (4) 创建获取数据的对象 `cmd`。
- (5) 打开数据库连接，获取数据，将获取的数据保存在 `dr` 中。
- (6) 操作成功，返回 `dr`。

上述功能的对应实现代码如下所示：

```

public SqlDataReader GetSingleGroup(int groupID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT * FROM [Group] WHERE ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = groupID;
    ///定义 SqlDataReader
    SqlDataReader dr;
    try
    {
        con.Open();
        ///读取数据
        dr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    return dr;
}

```

### 4. 添加新组信息

添加新组信息就是向系统库内添加新的用户组信息。

该功能是由方法 AddGroup(string name, int userID)实现的, 具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象, 实现数据库连接。
- (3) 新建 SQL 插入语句, 向系统数据库内添加新的用户组信息。
- (4) 创建添加数据的对象 cmd, 执行插入操作, 将操作结果保存在 result 中。
- (6) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```
public int AddGroup(string name, int userID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "INSERT INTO [Group] (Name,UserID) VALUES (@Name,@UserID) ";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@Name", SqlDbType.VarChar, 50);
    cmd.Parameters.Add("@UserID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = name;
    cmd.Parameters[1].Value = userID;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch(Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}
```

## 5. 修改用户组信息

修改用户组信息就是修改系统库内某编号的用户组信息。

该功能是由方法 UpdateGroup(int groupID, string name)实现的, 具体的实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象, 实现数据库连接。
- (3) 新建 SQL 更新语句, 对系统库内某编号的用户组信息进行修改。
- (4) 创建修改数据的对象 cmd 执行更新操作, 将操作结果保存在 result 中。
- (6) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```

public int UpdateGroup(int groupID, string name)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "UPDATE [Group] SET Name=@Name WHERE ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters.Add("@Name", SqlDbType.VarChar, 50);
    cmd.Parameters[0].Value = groupID;
    cmd.Parameters[1].Value = name;
    int result = -1;
    try
    {
        con.Open();
        ///操作数据
        result = cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return result;
}

```

## 6. 删除用户组信息

删除用户组信息就是删除系统库内某编号的用户组信息。

该功能是由方法 DeleteGroup(int groupID)实现的，具体的实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数，并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象，实现数据库连接。
- (3) 新建 SQL 删除语句，删除系统库内某编号的用户组信息。
- (4) 创建删除数据的对象 cmd，执行删除操作，将操作结果保存在 result 中。
- (5) 操作成功，返回 result。

上述功能的对应实现代码如下所示：

```

public int DeleteGroup(int groupID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "DELETE [Group] WHERE ID = @ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);

```



```

cmd.Parameters[0].Value = groupID;
int result = -1;
try
{
    ///打开连接
    con.Open();
    ///操作数据
    result = cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    throw new Exception(ex.Message, ex);
}
finally
{
    con.Close();
}
return result;
}

```

### 8.7.3 数据访问层——团队模块

本模块的数据访问层功能是由文件 Caboodle.cs 实现的。

该模块的主要功能是在 ASPNETAJAXWeb.AjaxInstantMessaging 空间内建立 Caboodle 类和 CaboodleUser 类，并定义多个方法来实现对数据库中用户数据的处理。

在文件 Caboodle.cs 中，与系统团队处理模块相关的方法如下：

- GetSelfCaboodleByUser(int userID)。
- GetCaboodleByUser(int userID)。
- GetSingleCaboodle(int caboodleID)。
- AddCaboodle(string name, int userID, string remark)。
- UpdateCaboodle(int caboodleID, string name, string remark)。
- DeleteCaboodle(int caboodleID)。
- GetUserbyCaboodle(int caboodleID)。
- AddCaboodleUser(int caboodleID, int userID, int roleID)。
- DeleteCaboodleUser(int caboodleID, int userID)。

其中，前 6 个方法是 Caboodle 类实现的。

#### 1. 定义 Caboodle 类

定义 Caboodle 类的主要实现代码如下所示：

```

using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;
namespace ASPNETAJAXWeb.AjaxInstantMessaging
{
    public class Caboodle
    {
        public Caboodle()
        {
        }
    }
}

```

## 2. 获取用户团队信息

获取用户团队信息就是获取某用户所创建的团队信息。

该功能是由方法 `GetSelfCaboodleByUser(int userID)` 实现的，其具体实现流程如下。

(1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。

(2) 使用该连接字符串创建 `con` 对象，实现数据库连接。

(3) 新建 SQL 查询语句，获取数据库内此登录用户创建的团队信息。

(4) 创建获取数据的对象 `da`。

(5) 打开数据库连接，获取数据，将获取的数据保存在 `ds` 中。

(6) 操作成功，返回 `ds`。

上述功能的对应实现代码如下所示：

```
public DataSet GetSelfCaboodleByUser(int userID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT Caboodle.*, [User].Username, (SELECT COUNT(*) FROM
CaboodleUser WHERE CaboodleID=Caboodle.ID) AS CaboodleUserCount FROM Caboodle INNER
JOIN[User] ON Caboodle.UserID=[User].ID WHERE Caboodle.UserID=@UserID";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@UserID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = userID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        ///打开连接
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch (Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
        con.Close();
    }
    return ds;
}
```

## 3. 获取用户团队的详细信息

获取用户团队的详细信息就是获取某用户所创建的团队和加入的团队信息，该功能是由方法 `GetCaboodleByUser(int userID)` 实现的，其具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数,并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象,实现数据库连接。
- (3) 新建 SQL 查询语句,获取数据库内此登录用户创建的团队信息。
- (4) 创建获取数据的对象 da。
- (5) 打开数据库连接,获取数据,将获取的数据保存在 ds 中。
- (6) 操作成功,返回 ds。

上述功能的对应实现代码如下所示:

```
public DataSet GetCaboodleByUser(int userID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    ///创建连接
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT DISTINCT Caboodle.* FROM Caboodle INNER JOIN CaboodleUser
ON Caboodle.ID=CaboodleUser.CaboodleID WHERE CaboodleUser.UserID=@UserID";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///创建参数并赋值
    da.SelectCommand.Parameters.Add("@UserID", SqlDbType.Int, 4);
    da.SelectCommand.Parameters[0].Value = userID;
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        con.Close();
    }
    return ds;
}
```

#### 4. 获取某团队信息

获取某团队信息即获取某编号团队的信息,该功能是由方法 GetSingleCaboodle(int caboodleID)实现的,其具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数,并将其保存在 connectionString 连接字符串中。
- (2) 使用该连接字符串创建 con 对象,实现数据库连接。
- (3) 新建 SQL 查询语句,获取数据库内指定 ID 的团队信息。
- (4) 创建获取数据的对象 cmd。
- (5) 打开数据库连接,获取数据,将获取的数据保存在 dr 中。



(6) 操作成功, 返回 dr。

上述功能的对应实现代码如下所示:

```
public SqlDataReader GetSingleCaboodle(int caboodleID)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    string cmdText = "SELECT * FROM [Caboodle] WHERE ID=@ID";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
    ///创建参数并赋值
    cmd.Parameters.Add("@ID", SqlDbType.Int, 4);
    cmd.Parameters[0].Value = caboodleID;
    ///定义 SqlDataReader
    SqlDataReader dr;
    try
    {
        con.Open();
        ///读取数据
        dr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    }
    catch (Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    return dr;
}
```

## 5. 添加团队信息

添加团队信息就是向系统库内添加新的团队的信息。

该功能是由方法 AddCaboodle(string name, int userID, string remark)实现的, 其具体实现流程如下。

- (1) 从系统配置文件 Web.config 内获取数据库连接参数, 并将其保存在 connectionString 连接字符串中。
- (2) 使用连接字符串创建 con 对象, 实现数据库连接。
- (3) 新建 SQL 插入语句, 向数据库内添加新的团队的信息。
- (4) 创建添加数据的对象 cmd。
- (5) 打开数据库连接, 执行添加处理, 将操作后的数据保存在 result 中。
- (6) 操作成功, 返回 result。

上述功能的对应实现代码如下所示:

```
public int AddCaboodle(string name, int userID, string remark)
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText =
        "INSERT INTO [Caboodle] (Name, UserID, Remark) VALUES (@Name, @UserID, @Remark) ";
    ///创建 SqlCommand
    SqlCommand cmd = new SqlCommand(cmdText, con);
```

```

///创建参数并赋值
cmd.Parameters.Add("@Name", SqlDbType.VarChar, 50);
cmd.Parameters.Add("@UserID", SqlDbType.Int, 4);
cmd.Parameters.Add("@Remark", SqlDbType.VarChar, 1000);
cmd.Parameters[0].Value = name;
cmd.Parameters[1].Value = userID;
cmd.Parameters[2].Value = remark;
int result = -1;
try
{
    con.Open();
    ///操作数据
    result = cmd.ExecuteNonQuery();
}
catch (Exception ex)
{
    throw new Exception(ex.Message, ex);
}
finally
{
    con.Close();
}
return result;
}

```

在此，只简要介绍了 Caboodle 类上述数据库访问层方法的实现流程，至于其他方法的实现过程，在此将不做详细介绍，读者只须参阅本书光盘中的对应文件即可了解。

## 8.8 实现身份验证模块

视频讲解 光盘：视频\第8章\实现身份验证模块.avi

本系统的身份验证具有很强的代表性，既包含用户登录验证，也包含用户注销登录模块。在本节的内容中，将详细讲解实现身份验证模块的具体实现过程。

### 8.8.1 用户登录验证模块

用户登录验证模块的功能，是对登录用户的数据进行验证，确保只有系统合法的用户才能登录系统。上述功能的实现文件如下：

- Login.aspx。
- Login.aspx.cs。

登录验证处理页面文件 Login.aspx.cs 的功能，是对获取的登录表单数据进行验证，确保只有是合法用户才能登录系统。具体的实现流程如下。

- (1) 引入命名空间，声明 UserLogin 类。
- (2) Page\_Load 载入初始化处理。
- (3) 激活 btnLogin\_Click(object sender, EventArgs e)事件，验证码验证处理。
- (4) 查询此登录数据，验证登录数据是否合法。
- (5) Session 保存合法登录数据。
- (6) 重定向系统主页。

文件 Login.aspx.cs 的主要实现代码如下所示:

```
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using ASPNETAJAXWeb.ValidateCode.Page;
using System.Data.SqlClient;
public partial class UserLogin : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void btnLogin_Click(object sender, EventArgs e)
    {
        if(Session[ValidateCode.VALIDATECODEKEY] != null)
        {
            ///验证验证码是否相等
            if(tbCode.Text != Session[ValidateCode.VALIDATECODEKEY].ToString())
            {
                lbMessage.Text = "验证码输入错误, 请重新输入";
                return;
            }
            ///判断用户的密码和名称是否正确
            ASPNETAJAXWeb.AjaxInstantMessaging.User user =
                new ASPNETAJAXWeb.AjaxInstantMessaging.User();
            SqlDataReader dr = user.GetUserLogin(tbUsername.Text, tbPassword.Text);
            if(dr == null) return;
            bool isLogin = false;
            if(dr.Read())
            {
                ///读取用户的登录信息, 并保存
                UserInfo ui = new UserInfo();
                ui.UserID = Int32.Parse(dr["ID"].ToString());
                ui.Username = tbUsername.Text;
                ///保存到 Session 中
                Session["UserID"] = ui.UserID;
                Session["Username"] = ui.Username;
                ///保存到全局信息中
                ASP.global_asax.Users.Add(ui);
                isLogin = true;
            }
            dr.Close();

            ///如果用户登录成功
            if(isLogin == true)
            {
                Response.Redirect("~/Default.aspx");
                return;
            }
        }
    }

    protected void btnReturn_Click(object sender, EventArgs e)
    {
        ///清空各种输入框中的信息
        tbUsername.Text = tbPassword.Text = tbCode.Text = string.Empty;
    }
}
```



## 8.8.2 登录用户注销模块

登录用户注销模块的功能，是使系统内的当前登录用户安全地退出当前系统。对应的实现文件如下：

- LogOff.aspx
- LogOff.aspx.cs

其中，文件 LogOff.aspx 是一个简单的中间页面，它通过调用其本身的后台隐藏文件 LogOff.aspx.cs，实现登录数据的注销处理功能。

文件 LogOff.aspx 实现隐藏代码调用的代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LogOff.aspx.cs"
    Inherits="LogOff" %>
```

文件 LogOff.aspx.cs 的功能，是引入命名空间并声明 LogOff 类，注销当前用户的登录数据。文件 LogOff.aspx.cs 的主要实现代码如下所示：

```
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class LogOff : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        ///清空用户信息
        Session["UserID"] = null;
        Session["Username"] = null;
        ///停止当前会话
        Session.Clear();
        Session.Abandon();
        ///重定向到用户登录页面
        Response.Redirect("~/Login.aspx");
    }
}
```

到此为止，完成了登录验证模块的工作。在一些登录验证系统中，都使用了验证码。但是，究竟验证码有什么作用呢？验证码一般是防止批量注册的，人眼看起来都费劲，何况是机器呢。例如，在百度贴吧中，如果处于未登录状态，发帖时要求输入验证码，这就是防止大规模匿名回帖情况的发生。目前，不少网站为了防止用户利用机器人自动注册、登录、灌水，都采用了验证码技术。所谓验证码，就是利用一串随机产生的数字或符号，生成一幅图片，图片里加上一些干扰像素，由用户肉眼识别其中的验证码信息，输入表单并提交网站验证，验证成功后才能使用某项功能。

## 8.9 客户分组处理模块

视频讲解  光盘：视频\第8章\客户分组处理模块.avi

客户分组即客户分类，最初我们不明白客户为什么要分组。难道同一个企业内还要分三六九等？其实，当今企业的客户分类功能势在必行，特别是在外资企业中，部门区分十

分严格，财务部、市场部、行政部等，每个部门可以作为一个小组，这样就能够实现部门内成员的交流了。

### 8.9.1 用户分组添加模块

用户分组添加模块的功能，是向系统内添加新的用户组。上述功能的实现文件如下：

- AddGroup.aspx。
- AddGroup.aspx.cs。

AddGroup.aspx.cs 的功能，是对获取的登录表单数据进行验证，确保只有合法用户才能登录系统。

具体的实现流程如下。

- (1) 引入命名空间，声明 Hailfellow\_AddGroup 类。
- (2) Page\_Load 载入初始化处理。
- (3) 登录验证处理。
- (4) 激活事件 btnCommit\_Click(object sender, EventArgs e)，调用方法 AddGroup 实现数据添加。
- (5) 重定向用户组管理列表界面。

文件 AddGroup.aspx.cs 的主要实现代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
public partial class Hailfellow_AddGroup : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        userID = Int32.Parse(Session["UserID"].ToString());
    }

    protected void btnCommit_Click(object sender, EventArgs e)
    {
        ///添加组
        Group group = new Group();
        if(group.AddGroup(tbName.Text, userID) > 0)
        {
            Response.Redirect("~/Hailfellow/GroupManage.aspx");
        }
    }
}
```

### 8.9.2 用户分组修改模块

用户分组修改模块的功能，是对系统库内的某用户组信息进行修改。该功能的实现文件如下：

- UpdateGroup.aspx。
- UpdateGroup.aspx.cs。

用户组修改处理文件 UpdateGroup.aspx.cs 的功能, 是对获取的登录表单数据进行验证, 确保只有是合法用户才能登录系统。其具体实现流程如下。

- (1) 引入命名空间, 声明 Hailfellow\_UpdateGroup 类。
- (2) Page\_Load 载入初始化处理。
- (3) 获取组编号。
- (4) 调用 BindPageData(int groupID), 获取并显示此用户组的数据。
- (5) 激活 btnCommit\_Click(object sender, EventArgs e), 通过方法 UpdateGroup 进行用户组更新处理。
- (6) 重定向, 返回组管理列表页面。

文件 UpdateGroup.aspx.cs 的主要代码如下所示:

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
public partial class Hailfellow_UpdateGroup : System.Web.UI.Page
{
    int groupID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取数据的ID值
        if(Request.Params["GroupID"] != null)
        {
            groupID = Int32.Parse(Request.Params["GroupID"].ToString());
        }
        if(!Page.IsPostBack && groupID>0)
        {
            BindPageData(groupID);
        }
    }
    private void BindPageData(int groupID)
    {
        ///读取数据
        Group group = new Group();
        SqlDataReader dr = group.GetSingleGroup(groupID);
        if(dr == null) return;
        ///显示数据
        if(dr.Read())
        {
            tbName.Text = dr["Name"].ToString();
        }
        dr.Close();
    }
    protected void btnCommit_Click(object sender, EventArgs e)
    {
        ///修改组
        Group group = new Group();
        if(group.UpdateGroup(groupID, tbName.Text) > 0)
        {
            Response.Redirect("~/Hailfellow/GroupManage.aspx");
        }
    }
}

```



### 8.9.3 用户组管理列表模块

用户组管理列表模块的功能，是以列表的样式将系统库内的用户组显示出来，并提供管理链接，对各用户组进行管理和维护。上述功能的实现文件如下：

- GroupManage.aspx。
- GroupManage.aspx.cs。

用户组列表处理文件 GroupManage.aspx.cs 的功能，是进行页面初始化处理，显示系统内的用户组信息。其具体实现流程如下。

- (1) 引入命名空间，声明 Hailfellow\_GroupManage 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证处理。
- (4) 获取用户组 ID。
- (5) 调用 BindPageData(int groupID)，获取并显示此用户组的信息。
- (6) 根据用户操作重定向到对应的处理页面。
- (7) 弹出“删除确认”对话框。
- (8) 删除指定编号的用户组信息。

文件 GroupManage.aspx.cs 的主要代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
public partial class Hailfellow_GroupManage : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户的ID值
        userID = Int32.Parse(Session["UserID"].ToString());
        if(!Page.IsPostBack)
        {
            BindPageData(userID);
        }
    }
    private void BindPageData(int userID)
    {
        ///读取数据
        Group group = new Group();
        DataSet ds = group.GetGroupByUser(userID);
        ///显示数据
        gvGroup.DataSource = ds;
        gvGroup.DataBind();
    }
    protected void btnAdd_Click(object sender, EventArgs e)
    {
        Response.Redirect("~/Hailfellow/AddGroup.aspx");
    }
}
```

```
protected void gvGroup_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName.ToLower() == "update")
    {
        ///重定向到修改组页面
        Response.Redirect(
            "~/Hailfellow/UpdateGroup.aspx?GroupID=" + e.CommandArgument.ToString());
        return;
    }
    if (e.CommandName.ToLower() == "del")
    {
        ///删除选择的组
        Group group = new Group();
        if (group.DeleteGroup(Int32.Parse(e.CommandArgument.ToString())) > 0)
        {
            BindPageData(userID);
        }
        return;
    }
}

protected void gvGroup_RowDataBound(object sender, GridViewRowEventArgs e)
{
    ///添加删除确认的对话框
    ImageButton imgDelete = (ImageButton)e.Row.FindControl("imgDelete");
    if (imgDelete != null)
    {
        imgDelete.Attributes.Add(
            "onclick", "return confirm(\"您确认要删除当前行的组吗? \");");
    }
}
}
```

#### 8.9.4 客户检索模块

客户检索模块的功能，是提供系统用户检索表单，将指定关键字的用户信息迅速检索出来。上述功能的实现文件如下：

- SearchFellow.aspx。
- SearchFellow.aspx.cs。

信息检索处理页面文件 SearchFellow.aspx.cs 的功能，是将系统库内满足搜索表单关键字和搜索方式的个人信息检索出来。具体的实现流程如下。

- (1) 引入命名空间，声明 Hailfellow\_SearchFellow 类。
- (2) Page\_Load 载入初始化处理。
- (3) 调用方法 GetUsers()，获取系统库内的用户数据。
- (4) 根据搜索方式参数进行检索语句定义，具体说明如下。
  - 参数 0：按照用户名称进行检索。
  - 参数 1：按照用户别名称进行检索。
  - 参数 2：按照用户名号码进行检索。
- (5) 验证码验证，开始进行检索处理。
- (6) 调用 ShowSearchResult()显示检索结果。

文件 SearchFellow.aspx.cs 的主要代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
using ASPNETAJAXWeb.ValidateCode.Page;
public partial class Hailfellow_SearchFellow : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    private void ShowSearchResult()
    {
        ///获取数据
        ASPNETAJAXWeb.AjaxInstantMessaging.User user =
            new ASPNETAJAXWeb.AjaxInstantMessaging.User();
        DataSet ds = user.GetUsers();
        if(ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        if(string.IsNullOrEmpty(tbKey.Text) == true) return;
        ///搜索给定条件的用户
        DataView dv = ds.Tables[0].DefaultView;
        switch(ddlMethod.SelectedValue)
        {
            case "0": ///按用户名称搜索
            {
                dv.RowFilter = "Username LIKE '%" + tbKey.Text + "%'";
                break;
            }
            case "1": ///按用户别名搜索
            {
                dv.RowFilter = "Aliasname LIKE '%" + tbKey.Text + "%'";
                break;
            }
            case "2": ///按用户号码搜索
            {
                dv.RowFilter = "UserIdentity LIKE '%" + tbKey.Text + "%'";
                break;
            }
            default: break;
        }
        dv.Sort = "UserIdentity";
        ///显示数据
        gvUser.DataSource = dv;
        gvUser.DataBind();
    }
    protected void btnCommit_Click(object sender, EventArgs e)
    {
        ///初始化搜索结果数据
        gvUser.DataSource = null;
        gvUser.DataBind();
        lbMessage.Visible = false;
        if(Session[ValidateCode.VALIDATECODEKEY] != null)
        {
            ///验证验证码是否相等
            if(tbCode.Text != Session[ValidateCode.VALIDATECODEKEY].ToString())
            {
                lbMessage.Text = "验证码输入错误, 请重新输入";
                lbMessage.Visible = true;
                return;
            }
        }
        ///搜索用户
        ShowSearchResult();
    }
}

```



```

}
protected void gvUser_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if(e.CommandName.ToLower() == "add")
    {
        ///重定向到添加客户页面
        Response.Redirect(
            "~/Hailfellow/AddFellow.aspx?FellowID=" + e.CommandArgument.ToString());
        return;
    }
}
protected void gvUser_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    ///重新显示数据
    gvUser.PageIndex = e.NewPageIndex;
    ShowSearchResult();
}
}

```

方法 `GetUsers()` 是文件 `User.cs` 内的一个数据库访问层方法，其功能是查询系统内所有用户的数据。具体的实现流程如下。

- (1) 从系统配置文件 `Web.config` 内获取数据库连接参数，并将其保存在 `connectionString` 连接字符串中。
- (2) 使用该连接字符串创建 `con` 对象，实现数据库连接。
- (3) 新建 SQL 查询语句，获取系统数据库内所有用户的信息。
- (4) 创建获取数据的对象 `da`。
- (5) 打开数据库连接，获取数据，将获取的数据保存在 `ds` 中。
- (6) 操作成功，返回 `ds`。

上述功能的对应实现代码如下所示：

```

public DataSet GetUsers()
{
    string connectionString =
        ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
    SqlConnection con = new SqlConnection(connectionString);
    ///创建 SQL 语句
    string cmdText = "SELECT * FROM [User]";
    ///创建 SqlDataAdapter
    SqlDataAdapter da = new SqlDataAdapter(cmdText, con);
    ///定义 DataSet
    DataSet ds = new DataSet();
    try
    {
        ///打开连接
        con.Open();
        ///填充数据
        da.Fill(ds, "DataTable");
    }
    catch(Exception ex)
    {
        ///抛出异常
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        ///关闭连接
    }
}

```

```

        con.Close();
    }
    return ds;
}

```

### 8.9.5 客户管理列表模块

用户管理列表模块的功能，是以列表的样式将系统库内某用户的客户信息显示出来，并提供管理链接，对各用户组进行管理和维护。该功能的实现文件如下：

- FellowManage.aspx。
- FellowManage.aspx.cs。

客户管理列表处理文件 FellowManage.aspx.cs 的功能，是初始化载入页面，将指定用户的客户信息读取并显示出来。其具体实现流程如下。

- (1) 引入命名空间，声明 Hailfellow\_FellowManage 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证判断。
- (4) 定义 BindUserData，获取并显示对应的客户数据。
- (5) 定义事件 gvUser\_RowCommand(object sender, GridViewCommandEventArgs e)，进行对应的操作处理。
- (6) 弹出“删除确认”对话框。
- (7) 执行删除处理。

文件 FellowManage.aspx.cs 主要实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
public partial class Hailfellow_FellowManage : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户的ID值
        userID = Int32.Parse(Session["UserID"].ToString());
        if(!Page.IsPostBack)
        {
            BindPageData(userID);
        }
    }
    private void BindPageData(int userID)
    {
        ///读取数据
        Group group = new Group();
        DataSet ds = group.GetGroupByUser(userID);
        ///显示数据
        gvGroup.DataSource = ds;
        gvGroup.DataBind();
    }
}

```

```

}
private void BindUserData(Gridview gv, int groupID)
{
    ///获取组的用户
    GroupUser gu = new GroupUser();
    DataSet ds = gu.GetUserbyGroup(groupID);
    ///显示组用户信息
    gv.DataSource = ds;
    gv.DataBind();
}
protected void gvGroup_RowDataBound(object sender, GridViewRowEventArgs e)
{
    ///显示组的用户
    GridView gvUser = (GridView)e.Row.FindControl("gvUser");
    if(gvUser != null)
    {
        BindUserData(gvUser,
            Int32.Parse(gvGroup.DataKeys[e.Row.RowIndex].Value.ToString()));
        gvUser.EmptyDataText = gvGroup.DataKeys[e.Row.RowIndex].Value.ToString();
    }
}
protected void gvUser_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if(e.CommandName.ToLower() == "del")
    {
        ///删除组中的客户
        GroupUser groupUser = new GroupUser();
        int groupID = Int32.Parse(((GridView)sender).EmptyDataText);
        if(groupUser.DeleteGroupUser(groupID,
            Int32.Parse(e.CommandArgument.ToString())) > 0)
        {
            BindUserData((GridView)sender, groupID);
        }
        return;
    }
}
protected void gvUser_RowDataBound(object sender, GridViewRowEventArgs e)
{
    ///添加删除确认的对话框
    ImageButton imgDelete = (ImageButton)e.Row.FindControl("imgDelete");
    if(imgDelete != null)
    {
        imgDelete.Attributes.Add("onclick",
            "return confirm(\"您确认要删除当前行的用户吗? \");");
    }
}
}

```

## 8.9.6 客户移动转换模块

客户移动转换模块的功能，是使系统用户能对自己的客户类别进行转换处理，以灵活地对客户信息进行维护。上述功能的实现文件如下：

- MoveFellow.aspx。
- MoveFellow.aspx.cs。

客户管理列表处理文件 MoveFellow.aspx.cs 的功能，是初始化载入页面，将指定用户的客户信息读取并显示出来。其具体实现流程如下。



- (1) 引入命名空间，声明 Hailfellow\_MoveFellow 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证判断。
- (4) 获取客户 ID 参数。
- (5) 定义 BindUserData，获取并显示对应的客户数据。
- (6) 定义事件 btnCommit\_Click(object sender, EventArgs)，进行移动操作处理。
- (7) 输出“移动成功”提示对话框。

文件 MoveFellow.aspx.cs 的实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
public partial class Hailfellow_MoveFellow : System.Web.UI.Page
{
    int userID = -1;
    int fellowID = -1;
    int oldGroupID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户的ID值
        userID = Int32.Parse(Session["UserID"].ToString());
        if(Request.Params["UserID"] != null)
        {
            fellowID = Int32.Parse(Request.Params["UserID"].ToString());
        }
        if(Request.Params["GroupID"] != null)
        {
            oldGroupID = Int32.Parse(Request.Params["GroupID"].ToString());
        }
        if(!Page.IsPostBack && userID>0 && fellowID>0)
        {
            BindPageData(userID, fellowID);
        }
    }
    private void BindPageData(int userID, int fellowID)
    {
        ///获取客户信息
        ASPNETAJAXWeb.AjaxInstantMessaging.User user =
            new ASPNETAJAXWeb.AjaxInstantMessaging.User();
        SqlDataReader dr = user.GetSingleUser(fellowID);
        if(dr == null) return;
        if(dr.Read())
        {
            ///显示客户名称
            lbUsername.Text = dr["Username"].ToString();
        }
        dr.Close();
        ///读取数据
        Group group = new Group();
        DataSet ds = group.GetGroupByUser(userID);
        ///显示数据
    }
}

```

```

ddlGroup.DataSource = ds;
ddlGroup.DataTextField = "Name";
ddlGroup.DataValueField = "ID";
ddlGroup.DataBind();
}
protected void btnCommit_Click(object sender, EventArgs e)
{
    ///判断用户是否选择组
    if(ddlGroup.SelectedIndex <= -1)
    {
        AjaxInstantMessagingSystem.ShowAjaxDialog(
            (Button)sender, "请选择客户移动的组");
        return;
    }
    ///添加客户
    GroupUser gu = new GroupUser();
    if(gu.UpdateGroupUser(
        oldGroupID, Int32.Parse(ddlGroup.SelectedValue), fellowID) > 0)
    {
        ///显示添加客户成功信息
        AjaxInstantMessagingSystem.ShowAjaxDialog(
            (Button)sender, "恭喜您, 移动客户成功!");
    }
}
}

```

### 8.9.7 客户信息显示模块

客户信息显示模块的功能, 是将系统内某用户的客户信息详细地显示出来。对应的实现文件如下:

- ShowFellowInfo.aspx。
- ShowFellowInfo.aspx.cs。

客户详情处理文件 ShowFellowInfo.aspx.cs 的功能, 是初始化载入页面, 将指定编号的客户信息读取并显示出来。其具体的实现流程如下。

- (1) 引入命名空间, 声明 Hailfellow\_ShowFellowInfo 类。
- (2) Page\_Load 载入初始化处理。
- (3) 获取客户 ID 参数。
- (4) 定义 BindUserData, 获取并显示对应的客户数据。
- (5) 定义页面关闭事件 btnCommit\_Click(object sender, EventArgs e)。

文件 ShowFellowInfo.aspx.cs 的主要实现代码如下所示:

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
public partial class Hailfellow_ShowFellowInfo : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取用户的 ID 值
        if(Request.Params["UserID"] != null)
        {
            userID = Int32.Parse(Request.Params["UserID"].ToString());
        }
    }
}

```

```
    }  
    if(!Page.IsPostBack && userID>0)  
    {  
        ///显示用户的信息  
        BindPageData(userID);  
    }  
}  
private void BindPageData(int userID)  
{  
    ///获取用户信息  
    ASPNETAJAXWeb.AjaxInstantMessaging.User user =  
        new ASPNETAJAXWeb.AjaxInstantMessaging.User();  
    SqlDataReader dr = user.GetSingleUser(userID);  
    if(dr == null) return;  
    if(dr.Read())  
    {  
        ///显示用户信息  
        lbUsername.Text = dr["Username"].ToString();  
        lbAliasname.Text = dr["Aliasname"].ToString();  
        lbUserIdentity.Text = dr["UserIdentity"].ToString();  
        lbEmail.Text = dr["Email"].ToString();  
    }  
    dr.Close();  
}  
protected void btnCommit_Click(object sender, EventArgs e)  
{  
    Response.Write("<script>window.close();</script>");  
}  
}
```

到此为止，完成了类别管理模块的编码工作。类别管理在 Web 项目中很常见，例如，新闻系统中有新闻类别，电子商务系统中有产品类别……看似很复杂，其实它们的实现原理都是一致的，只是在数据库中增加了一个类别表。

我们能通过后台对类别信息进行管理。在添加那些分类信息时，增加一个类别值。例如添加新闻信息时，原来的添加表单可能分为标题、内容和时间，但是现在得分为标题、类别、内容和时间，即在数据库中增加了“类别”列。

## 8.10 系统团队处理模块

视频讲解  光盘：视频\第 8 章\系统团队处理模块.avi

系统团队处理模块的功能，是根据现实的客观需要，在系统内创建专门的团队，来实现企业特定的任务。并且根据现实状况的变化，而对团队进行及时的管理和调整。在下面的内容中，将对上述功能文件的实现过程进行详细介绍。

### 8.10.1 添加团队模块

添加团队模块的功能，是向系统库内添加新的团队信息。该功能的实现文件如下：

- AddCaboodle.aspx。
- AddCaboodle.aspx.cs。

客户添加处理文件 AddCaboodle.aspx.cs 的功能，是初始化载入页面，将获取表单的数



据添加到系统库中。具体的实现流程如下。

- (1) 引入命名空间，声明 Caboodle\_AddCaboodle 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录判断处理。
- (4) Session 保存登录数据。
- (5) 定义 btnCommit\_Click(object sender, EventArgs e)，执行添加处理。
- (6) 重定向管理列表界面。

文件 AddCaboodle.aspx.cs 的主要实现代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
public partial class Caboodle_AddCaboodle : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户信息
        userID = Int32.Parse(Session["UserID"].ToString());
    }
    protected void btnCommit_Click(object sender, EventArgs e)
    {
        ///添加群
        Caboodle caboodle = new Caboodle();
        if(caboodle.AddCaboodle(tbName.Text, userID, tbRemark.Text) > 0)
        {
            Response.Redirect("~/Caboodle/CaboodleManage.aspx");
        }
    }
}
```

## 8.10.2 修改团队处理模块

修改团队处理模块的功能，是对系统库内某编号的团队信息进行修改。该功能的实现文件如下：

- UpdateCaboodle.aspx。
- UpdateCaboodle.aspx.cs。

团队修改处理文件 UpdateCaboodle.aspx.cs 的功能，是初始化载入页面，对指定编号的团队信息进行更新处理。具体的实现流程如下。

- (1) 引入命名空间，声明 Caboodle\_UpdateCaboodle 类。
- (2) Page\_Load 载入初始化处理。
- (3) 获取团队的 ID 值。
- (4) 定义 BindPageData(int caboodleID)，获取并显示原信息。
- (5) 定义 btnCommit\_Click(object sender, EventArgs e)，执行更新处理。

## (6) 重定向管理列表页面。

文件 UpdateCaboodle.aspx.cs 的主要实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
public partial class Caboodle_UpdateCaboodle : System.Web.UI.Page
{
    int caboodleID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///获取群的 ID 值
        if(Request.Params["CaboodleID"] != null)
        {
            caboodleID = Int32.Parse(Request.Params["CaboodleID"].ToString());
        }
        if(!Page.IsPostBack && caboodleID>0)
        {
            ///显示群信息
            BindPageData(caboodleID);
        }
    }
    private void BindPageData(int caboodleID)
    {
        ///读取数据
        Caboodle caboodle = new Caboodle();
        SqlDataReader dr = caboodle.GetSingleCaboodle(caboodleID);
        if(dr == null) return;
        ///显示数据
        if(dr.Read())
        {
            tbName.Text = dr["Name"].ToString();
            tbRemark.Text = dr["Remark"].ToString();
        }
        dr.Close();
    }
    protected void btnCommit_Click(object sender, EventArgs e)
    {
        ///修改群
        Caboodle caboodle = new Caboodle();
        if(caboodle.UpdateCaboodle(caboodleID, tbName.Text, tbRemark.Text) > 0)
        {
            Response.Redirect("~/Caboodle/CaboodleManage.aspx");
        }
    }
}

```

### 8.10.3 团队管理列表模块

团队管理列表模块的功能，是对系统数据库内某用户的团队信息进行管理。上述功能的实现文件如下：

- CaboodleManage.aspx
- CaboodleManage.aspx.cs

团队列表处理文件 CaboodleManage.aspx.cs 的功能，是初始化载入页面，将指定编号的团队信息进行更新处理。其具体实现流程如下。

- (1) 引入命名空间, 声明 Caboodle\_CaboodleManage 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证判断。
- (4) 定义 BindPageData(int userID), 获取并显示信息。
- (5) 定义 gvCaboodle\_RowCommand(object sender, GridViewCommandEventArgs e), 执行对应的处理。
- (6) 重定向, 返回列表页面。

文件 CaboodleManage.aspx.cs 的主要实现代码如下所示:

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
public partial class Caboodle_CaboodleManage : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        userID = Int32.Parse(Session["UserID"].ToString()); ///获取用户的ID值
        if(!Page.IsPostBack)
        {
            BindPageData(userID);
        }
    }
    private void BindPageData(int userID)
    {
        ///读取数据
        Caboodle caboodle = new Caboodle();
        DataSet ds = caboodle.GetSelfCaboodleByUser(userID);
        if(ds == null || ds.Tables.Count <= 0) return;
        ///显示数据
        gvCaboodle.DataSource = ds;
        gvCaboodle.DataBind();
        if(ds.Tables[0].Rows.Count <= 0) btnAdd.Enabled = true;
        else btnAdd.Enabled = false;
    }
    protected void btnAdd_Click(object sender, EventArgs e)
    {
        Response.Redirect("~/Caboodle/AddCaboodle.aspx");
    }
    protected void gvCaboodle_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        if(e.CommandName.ToLower() == "update")
        {
            ///重定向到修改组页面
            Response.Redirect("~/Caboodle/UpdateCaboodle.aspx?CaboodleID="
                + e.CommandArgument.ToString());
            return;
        }
        if(e.CommandName.ToLower() == "del")
        {
            ///删除选择的组
            Caboodle caboodle = new Caboodle();

```



```

        if (caboodle.DeleteCaboodle(Int32.Parse(e.CommandArgument.ToString())) > 0)
        {
            BindPageData(userID);
        }
        return;
    }
}
protected void gvCaboodle_RowDataBound(object sender, GridViewRowEventArgs e)
{
    ///添加删除确认的对话框
    ImageButton imgDelete = (ImageButton)e.Row.FindControl("imgDelete");
    if (imgDelete != null)
    {
        imgDelete.Attributes.Add("onclick",
            "return confirm(\"您确认要删除当前行的组吗? \");");
    }
}
}

```

### 8.10.4 加入团队处理模块

加入团队处理模块的功能，是使系统当前登录用户加入到客户的团队中。对应的实现文件如下：

- AddCaboodleUser.aspx
- AddCaboodleUser.aspx.cs

加入团队处理文件 AddCaboodleUser.aspx.cs 的功能，是初始化载入页面，将用户加入到其指定的团队中。具体的实现流程如下。

- (1) 引入命名空间，声明 Caboodle\_AddCaboodleUser 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证判断。
- (4) 定义 BindPageData(int userID)，获取并显示客户信息。
- (5) 定义事件 gvUser\_RowCommand(object sender, GridViewCommandEventArgs e)，执行对应的处理。
- (6) 重定向，返回列表页面。

文件 AddCaboodleUser.aspx.cs 的主要实现代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
public partial class Caboodle_AddCaboodleUser : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if (Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户信息
        userID = Int32.Parse(Session["UserID"].ToString());
    }
}

```

```

        if(!Page.IsPostBack && userID>0)
        {
            BindPageData(userID);
        }
    }
    private void BindPageData(int userID)
    {
        ///获取信息
        Caboodle caboodle = new Caboodle();
        DataSet ds = caboodle.GetSelfCaboodleByUser(userID);
        if(ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        ///显示名称
        lbGroupName.Text = ds.Tables[0].Rows[0]["Name"].ToString();
        ///保存 ID 值
        ViewState["CaboodleIDKey"] = ds.Tables[0].Rows[0]["ID"].ToString();
        ///获取数据
        ASPNETAJAXWeb.AjaxInstantMessaging.User user =
            new ASPNETAJAXWeb.AjaxInstantMessaging.User();
        ///绑定并显示数据
        gvUser.DataSource = user.GetFellowNotInCaboodleByUser(userID);
        gvUser.DataBind();
    }
    protected void gvUser_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        if(e.CommandName.ToLower() == "add")
        {
            if(ViewState["CaboodleIDKey"] != null)
            {
                CaboodleUser caboodleUser = new CaboodleUser();
                ///获取群的 ID 值
                int caboodleID = Int32.Parse(ViewState["CaboodleIDKey"].ToString());
                ///添加客户到团队
                caboodleUser.AddCaboodleUser(caboodleID,
                    Int32.Parse(e.CommandArgument.ToString()), 4);
                ///显示操作结果
                AjaxInstantMessagingSystem.ShowAjaxDialog((Button)e.CommandSource,
                    "恭喜您, 添加客户到群成功。");
                BindPageData(userID); ///重新显示数据
            }
            return;
        }
    }
    protected void gvUser_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        ///重新显示数据
        gvUser.PageIndex = e.NewPageIndex;
        BindPageData(userID);
    }
}

```

## 8.11 在线交互模块

视频讲解  光盘：视频\第8章\在线交互模块.avi

在本节的内容中，将开始实现整个项目的核心——在线交互模块的编码工作。其实，在线交互与聊天系统的交互原理是一致的，当用户发表信息后，只须通过无刷新技术将信

息显示出来即可。

## 8.11.1 系统主页显示模块

系统主页是一个框架页面，功能是调用框架页显示系统的用户分组列表，并实现用户的在线交互。主页内各构成框架文件如下：

- Default.aspx。
- Default.aspx.cs。
- Fellow.aspx。
- Fellow.aspx.cs。
- Header.aspx。
- Desktop.aspx。
- Desktop.aspx.cs。

### 1. 主框架处理页面

主框架处理页面文件 Default.aspx.cs 的功能，是对登录用户进行登录判断处理，如果没有登录，则返回登录表单页面。主要代码如下所示：

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
    }
}
```

### 2. 分组列表显示处理页面

分组列表显示处理文件 Fellow.aspx.cs 的功能，是初始化载入页面，将用户加入到其指定的团队中。具体的实现流程如下。

- (1) 引入命名空间，声明 Hailfellow\_Fellow 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证判断。
- (4) 定义 BindPageData(int userID)，获取并显示客户组信息。
- (5) 定义 BindUserData(DataList dl, int groupID)，获取并显示客户组内的对应客户信息。

文件 Fellow.aspx.cs 的主要代码如下所示：

```
///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
public partial class Hailfellow_Fellow : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
```



```

{
    ///判断用户是否登录
    if(Session["UserID"] == null)
    {
        Response.Redirect("~/Login.aspx");
        return;
    }
    ///获取用户的ID值
    userID = Int32.Parse(Session["UserID"].ToString());
    if(!Page.IsPostBack)
    {
        BindPageData(userID);
    }
}
private void BindPageData(int userID)
{
    ///读取数据
    Group group = new Group();
    DataSet ds = group.GetGroupByUser(userID);
    ///显示数据
    gvGroup.DataSource = ds;
    gvGroup.DataBind();
}
private void BindUserData(DataList dl,int groupID)
{
    ///获取组中的用户
    GroupUser gu = new GroupUser();
    DataSet ds = gu.GetUserbyGroup(groupID);
    ///显示组中的用户
    dl.DataSource = ds;
    dl.DataBind();
}
protected void gvGroup_RowDataBound(object sender,GridViewRowEventArgs e)
{
    ///显示每一个组的用户
    DataList dlUser = (DataList)e.Row.FindControl("dlUser");
    if(dlUser != null)
    {
        ///绑定数据
        BindUserData(
            dlUser,Int32.Parse(gvGroup.DataKeys[e.Row.RowIndex].Value.ToString()));
    }
}
}

```

### 8.11.2 一对一交互处理模块

一对一交互处理模块的功能,是实现系统内用户一对一在线交互功能。上述功能的实现文件如下:

- Messaging.aspx。
- Messaging.aspx.cs。

一对一交互处理页面文件 Messaging.aspx.cs 的功能,是初始化载入页面,对用户的交互数据进行处理。具体的实现流程如下。

- (1) 引入命名空间,声明 Messaging 类。
- (2) Page\_Load 载入初始化处理。

- (3) 用户登录验证判断。
- (4) 获取用户 ID, 并显示用户数据。
- (5) 保存当前用户进入系统的时间。
- (6) 定义 ShowMessageData(), 获取并显示交互信息。
- (7) 定义 ShowMessageData(), 将新发布的数据显示出来。

文件 Messaging.aspx.cs 的主要代码如下所示:

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
using System.Text;
public partial class Messaging : System.Web.UI.Page
{
    int userID = -1;
    int fellowID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户的ID值
        userID = Int32.Parse(Session["UserID"].ToString());
        if(Request.Params["UserID"] != null)
        {
            fellowID = Int32.Parse(Request.Params["UserID"].ToString());
        }
        if(!Page.IsPostBack && userID > 0 && fellowID > 0)
        {
            ///保存进入系统的时间
            ViewState["StartDate"] = DateTime.Now.ToString();
            BindPageData(userID, fellowID);
        }
        btnCommit.Enabled = (userID > 0 && fellowID > 0) ? true : false;
    }
    private void BindPageData(int userID, int fellowID)
    {
        ///获取客户信息
        ASPNETAJAXWeb.AjaxInstantMessaging.User user =
            new ASPNETAJAXWeb.AjaxInstantMessaging.User();
        SqlDataReader dr = user.GetSingleUser(fellowID);
        if(dr == null) return;
        if(dr.Read())
        {
            ///显示客户名称
            lbUsername.Text = "正在与团队: " + dr["Username"].ToString() + " 交互...";
        }
        dr.Close();
    }
    private void ShowMessageData()
    {
        ///获取所有消息
        MessageForSingle message = new MessageForSingle();
        DataSet ds = message.GetMessageByUser(userID, fellowID);
        if(ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        ///过滤进入该交互室之前的消息, 保留进入该交互室之后的消息
    }
}

```

```

DataView dv = ds.Tables[0].DefaultView;
dv.RowFilter = string.Format("CreateDate >= '{0}'",
    DateTime.Parse(ViewState["StartDate"].ToString()));
///构建交互的消息
StringBuilder sbMessage = new StringBuilder();
foreach(DataRowView row in dv)
{
    ///设置一条消息
    string singleMessage = row["SenderName"].ToString() + " 在["
        + row["CreateDate"].ToString() + "]发表: \n";
    singleMessage += " " + row["Body"].ToString() + "\n";
    sbMessage.Append(singleMessage);
}
///显示交互消息
tbChatMessage.Text = sbMessage.ToString();
}
protected void btnCommit_Click(object sender, EventArgs e)
{
    ///发送消息
    MessageForSingle message = new MessageForSingle();
    if(message.AddMessage(tbMessage.Text, userID, fellowID) > 0)
    {
        ///重新显示消息
        ShowMessageData();
    }
}
protected void tUser_Tick(object sender, EventArgs e)
{
    ///显示最新消息
    ShowMessageData();
}
}

```

### 8.11.3 团队交互处理模块

团队交互处理模块的功能，是共同实现系统内某团队用户的在线交互功能。对应的实现文件如下：

- SelectCaboodle.aspx。
- SelectCaboodle.aspx.cs。
- CaboodleMessaging.aspx。
- CaboodleMessaging.aspx.cs。

#### 1. 页面初始化处理

文件 SelectCaboodle.aspx.cs 的功能，是将页面进行初始化处理，进入用户选定团队的交互界面。其具体实现流程如下。

- (1) 引入命名空间，声明 Caboodle\_CaboodleManage 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证判断。
- (4) 定义 BindPageData(int userID)，获取并显示团队信息。
- (5) 重定向页面文件。

文件 SelectCaboodle.aspx.cs 的主要代码如下所示：



```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
public partial class Caboodle_CaboodleManage : System.Web.UI.Page
{
    int userID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户的ID值
        userID = Int32.Parse(Session["UserID"].ToString());
        if(!Page.IsPostBack)
        {
            BindPageData(userID);
        }
    }
    private void BindPageData(int userID)
    {
        ///读取数据
        Caboodle caboodle = new Caboodle();
        DataSet ds = caboodle.GetSelfCaboodleByUser(userID);
        if(ds == null || ds.Tables.Count <= 0) return;
        ///显示数据
        gvCaboodle.DataSource = ds;
        gvCaboodle.DataBind();
        if(ds.Tables[0].Rows.Count <= 0) btnAdd.Enabled = true;
        else btnAdd.Enabled = false;
    }
}

```

## 2. 团队交互处理页面

团队交互处理页面文件 CaboodleMessaging.aspx.cs 的功能，是初始化载入页面，对用户的交互数据进行处理。具体的实现流程如下。

- (1) 引入命名空间，声明 CaboodleMessaging 类。
  - (2) Page\_Load 载入初始化处理。
  - (3) 用户登录验证判断。
  - (4) 获取用户 ID，并显示用户数据。
  - (5) 保存进入系统的时间。
  - (6) 定义 BindPageData(int userID, int caboodleID)，初始化页面信息。
  - (7) 定义 ShowMessageData(int caboodleID)，获取并显示用户的交互信息。
- 文件 CaboodleMessaging.aspx.cs 的主要代码如下所示：

```

///引入新的命名空间
using ASPNETAJAXWeb.AjaxInstantMessaging;
using System.Data.SqlClient;
using System.Text;
using System.Collections.Generic;
public partial class CaboodleMessaging : System.Web.UI.Page
{
    int userID = -1;

```

```

int caboodleID = -1;
protected void Page_Load(object sender, EventArgs e)
{
    ///判断用户是否登录
    if(Session["UserID"] == null)
    {
        Response.Redirect("~/Login.aspx");
        return;
    }
    ///获取用户的ID值
    userID = Int32.Parse(Session["UserID"].ToString());
    if(Request.Params["CaboodleID"] != null)
    {
        caboodleID = Int32.Parse(Request.Params["CaboodleID"].ToString());
    }
    if(!Page.IsPostBack && userID>0 && caboodleID>0)
    {
        ///保存进入交互室的时间
        ViewState["StartDate"] = DateTime.Now.ToString();
        ///设置用户登录当前团队
        InitCaboodleUser();
        ///显示消息和用户
        BindPageData(userID, caboodleID);
        ShowUserData();
    }
    btnCommit.Enabled = (userID > 0 && caboodleID > 0) ? true : false;
}
private void InitCaboodleUser()
{
    ///设置用户进入的团队
    for(int i=0; i<ASP.global_asax.Users.Count; i++)
    {
        if(ASP.global_asax.Users[i].UserID.ToString()
            == Session["UserID"].ToString())
        {
            ASP.global_asax.Users[i].CaboodleID = caboodleID;
            break;
        }
    }
}
private void ShowUserData()
{
    ///获取群交互团队的用户
    List<UserInfo> users = new List<UserInfo>();
    foreach(UserInfo ui in ASP.global_asax.Users)
    {
        if(ui.CaboodleID == caboodleID)
        {
            users.Add(ui);
        }
    }
    ///显示群交互室的用户
    lbUser.DataSource = users;
    lbUser.DataValueField = "UserID";
    lbUser.DataTextField = "Username";
    lbUser.DataBind();
}
private void BindPageData(int userID, int caboodleID)
{
    ///获取用户信息
    ASPNETAJAXWeb.AjaxInstantMessaging.User user =

```

```

        new ASPNETAJAXWeb.AjaxInstantMessaging.User();
        SqlDataReader dr = user.GetSingleUser(userID);
        if(dr == null) return;
        string username = string.Empty;
        if(dr.Read())
        {
            ///获取用户名称
            username = dr["Username"].ToString();
        }
        dr.Close();
        ///获取群信息
        Caboodle caboodle = new Caboodle();
        SqlDataReader drc = caboodle.GetSingleCaboodle(caboodleID);
        if(drc == null) return;
        if(drc.Read())
        {
            ///读取并显示群的信息
            lbInfoMessage.Text = "用户 " + username + " 正在 "
                + drc["Name"].ToString() + " 群中交互...";
        }
        drc.Close();
    }
    private void ShowMessageData(int caboodleID)
    {
        ///获取所有消息
        MessageForCaboodle message = new MessageForCaboodle();
        DataSet ds = message.GetMessageByCaboodle(caboodleID);
        if(ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        ///过滤进入该交互室之前的消息, 保留进入该交互室之后的消息
        DataView dv = ds.Tables[0].DefaultView;
        dv.RowFilter = string.Format(
            "CreateDate >= '{0}'", DateTime.Parse(ViewState["StartDate"].ToString()));
        ///构建交互的消息
        StringBuilder sbMessage = new StringBuilder();
        foreach(DataRowView row in dv)
        {
            ///设置一条消息
            string singleMessage = row["SenderName"].ToString() + " 在["
                + row["CreateDate"].ToString() + "]发表: \n";
            singleMessage += " " + row["Body"].ToString() + "\n";
            sbMessage.Append(singleMessage);
        }
        ///显示交互消息
        tbChatMessage.Text = sbMessage.ToString();
    }
    protected void btnCommit_Click(object sender, EventArgs e)
    {
        ///发送消息
        MessageForCaboodle message = new MessageForCaboodle();
        if(message.AddMessage(tbMessage.Text, userID, caboodleID) > 0)
        {
            ///重新显示消息
            ShowMessageData(caboodleID);
        }
    }
    protected void tUser_Tick(object sender, EventArgs e)
    {
        ///显示最新消息和在线用户
        ShowMessageData(caboodleID);
        ShowUserData();
    }

```



### 8.11.4 文件发送模块

在.NET 框架类库的 System.IO 命名空间内，提供了专用的类，来实现对文件系统的操作管理，包括常见的复制、删除、文件移动等操作。在 System.IO 命名空间中，与文件管理类相关的主要信息如下。

- System.MarshalByRefObject：访问远程处理的应用程序，可以在分布的不同程序之间调用数据。
- System.IO.FileSystemInfo：是类 FileInfo 和 DirectoryInfo 的基类，封装了文件和目录操作的全部方法。
- System.IO.File：提供了创建、复制、删除、移动等文件处理的静态方法，并协同创建对象 FileStream。
- System.IO.FileInfo：提供了创建、复制、删除、移动等文件处理的静态方法，并协同创建对象 FileStream。
- System.IO.Directory：提供了创建、复制、删除、移动等文件处理的实例方法，并协同创建对象 FileStream。
- System.IO.DirectoryInfo：提供了创建、移动和枚举目录和子目录的实例方法。
- System.IO.Path：对包含文件或目录路径的 String 实例进行跨平台方式操作。
- System.Environment：设置当前运行环境平台的信息。

上述各个文件操作类型的对应关系如图 8-5 所示。

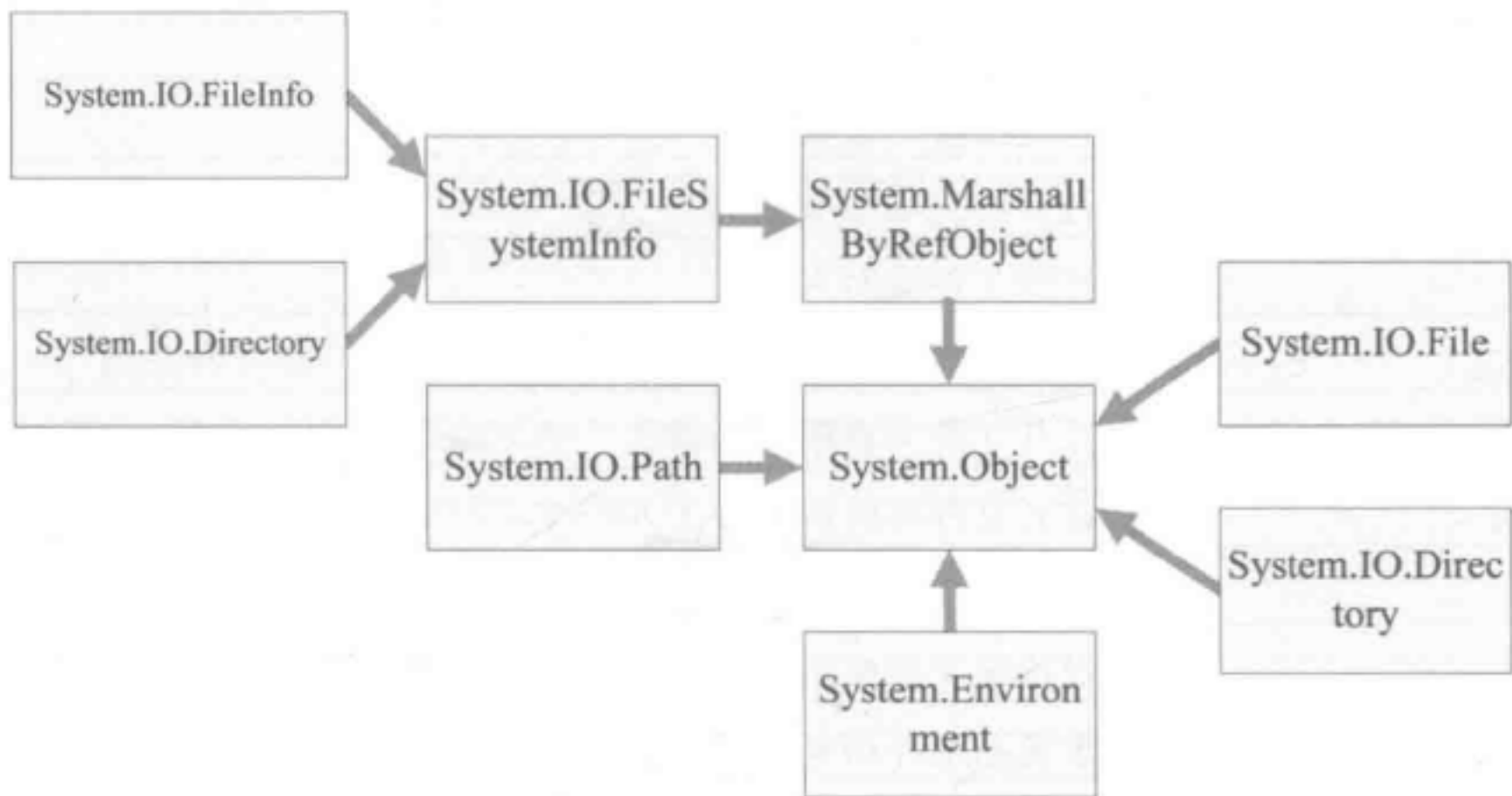


图 8-5 System.IO 命名空间文件操作类型的关系

因为上述各个文件操作类型都是静态类或封闭类，所以不能被继承。在 IO 内的 File 和 FileInfo 类的功能类似，能够实现对文件的复制、移动、重命名、创建和打开等基本操作，并能获取和设置文件属性及文件创建、访问和写入操作的 DateTime 信息。

类 File 中主要方法的具体说明如表 8-12 所示。

表 8-12 类 File 方法的信息

| 方 法                  | 说 明                                |
|----------------------|------------------------------------|
| AppendAllText        | 将指定字符串添加到文件中，如果文件不存在，则创建该文件        |
| AppendText           | 创建一个 Stream，将 WriteUTF-8 编码文件追加    |
| Copy                 | 将现有文件复制到新文件                        |
| Create               | 创建新文件                              |
| CreateText           | 创建或打开一个 UTF-8 编码的文件                |
| Decrypt              | 解密用 Encrypt 加密的文件                  |
| Delete               | 删除指定文件                             |
| Encrypt              | 加密某文件                              |
| Exists               | 验证文件是否存在                           |
| GetAccessControl     | 获取一个 FileSecurity 对象               |
| GetAttributes        | 获取文件的属性                            |
| GetCreationTime      | 获取文件或目录的创建时间                       |
| GetCreationTimeUtc   | 获取文件或目录的创建 UTC 格式时间                |
| GetLastAccessTime    | 获取文件或目录的最后被访问时间                    |
| GetLastAccessTimeUtc | 获取文件或目录的最后被访问的 UTC 格式时间            |
| GetLastWriteTime     | 获取文件或目录的上次被写入的时间                   |
| GetLastWriteTimeUtc  | 获取文件或目录的上次被写入的 UTC 格式时间            |
| Move                 | 移动某文件                              |
| Open                 | 打开某路径上的 FileStream                 |
| OpenRead             | 打开文件并读取                            |
| OpenText             | 打开 UTF-8 编码文件并读取                   |
| OpenWrite            | 打开文件并写入                            |
| ReadAllBytes         | 打开文件并将内容读入一个字符串，然后将文件关闭            |
| ReadAllLines         | 打开文件并将所有行读入一个字符串数组，然后将文件关闭         |
| ReadAllText          | 打开文件并将所有行读入一个字符串，然后将文件关闭           |
| Replace              | 替换某文件内容                            |
| SetAccessControl     | 对指定文件应用由 FileSecurity 对象描述的访问控制列表项 |
| SetAttributes        | 设置某文件指定的 FileAttributes            |
| SetCreationTime      | 设置创建文件的时间                          |
| SetCreationTimeUtc   | 设置创建文件的 UTC 格式时间                   |
| SetLastAccessTime    | 设置上次访问文件的时间                        |
| SetLastAccessTimeUtc | 设置上次访问文件的 UTC 格式时间                 |
| SetLastWriteTime     | 设置上次写入文件的时间                        |
| SetLastWriteTimeUtc  | 设置上次写入文件的 UTC 格式时间                 |
| WriteAllBytes        | 创建新文件，并写入指定的字节数组，然后关闭文件            |

续表

| 方 法           | 说 明                    |
|---------------|------------------------|
| WriteAllLines | 创建新文件，并写入指定的字符串，然后关闭文件 |
| WriteAllText  | 创建新文件，并写入内容，然后关闭文件     |

类 FileInfo 的主要属性信息如表 8-13 所示。

表 8-13 类 FileInfo 的属性信息

| 属 性               | 说 明                                    |
|-------------------|----------------------------------------|
| Attributes        | 获取设置当前 FileSystemInfo 的 FileAttributes |
| CreationTime      | 获取设置当前 FileSystemInfo 的创建时间            |
| CreationTimeUtc   | 获取设置当前 FileSystemInfo 的 UTC 格式时间       |
| Directory         | 解密用 Encrypt 加密的文件                      |
| DirectoryName     | 获取父目录实例                                |
| Exists            | 验证文件是否存在                               |
| Extension         | 获取文件扩展名字符串                             |
| FullName          | 获取文件或目录的完整名或目录                         |
| IsReadOnly        | 设置文件为只读                                |
| LastAccessTime    | 获取文件或目录的上次被访问的时间                       |
| LastAccessTimeUtc | 获取文件或目录的上次被访问的 UTC 格式时间                |
| LastWriteTime     | 获取文件或目录的上次被写入的时间                       |
| LastWriteTimeUtc  | 获取文件或目录的上次被写入的 UTC 格式时间                |
| Length            | 获取文件的大小                                |
| Name              | 获取文件名                                  |

类 FileInfo 的主要方法信息如表 8-14 所示。

表 8-14 类 FileInfo 方法信息

| 方 法              | 说 明                    |
|------------------|------------------------|
| AppendText       | 创建一个 StreamWriter      |
| CopyTo           | 将现有文件复制到新文件            |
| Create           | 创建新文件                  |
| CreateObjRef     | 创建远程代理对象               |
| CreateText       | 创建、写入新文件的 StreamWriter |
| Decrypt          | 解密用 Encrypt 加密的文件      |
| Delete           | 删除指定文件                 |
| Encrypt          | 加密某文件                  |
| GetAccessControl | 获取 FileSecurity 对象     |
| MoveTo           | 移动某文件                  |



续表

| 方 法              | 说 明                                |
|------------------|------------------------------------|
| Open             | 打开某文件                              |
| OpenRead         | 创建只读 FileStream                    |
| OpenText         | 打开 UTF-8 编码文件并读取                   |
| OpenWrite        | 打开只写 FileStream                    |
| Replace          | 替换某文件内容                            |
| SetAccessControl | 对指定文件应用由 FileSecurity 对象描述的访问控制列表项 |

本系统文件发送模块的功能，实现系统内用户间的文件传送功能。上述功能的实现文件如下：

- File.aspx。
- File.aspx.cs。

文件发送处理文件 File.aspx.cs 的功能，是初始化载入页面，对获取的传送文件数据进行处理。具体的实现流程如下。

- (1) 引入命名空间，声明 FilePage 类。
- (2) Page\_Load 载入初始化处理。
- (3) 用户登录验证判断。
- (4) 获取用户 ID，并显示用户数据。
- (5) 定义 BindPageData(int userID, int fellowID)，获取客户信息。
- (6) 定义 btnCommit\_Click(object sender, EventArgs e)事件，进行文件上载处理。
- (7) 处理完毕，输出成功提示。

文件 File.aspx.cs 的主要代码如下所示：

```
public partial class FilePage : System.Web.UI.Page
{
    int userID = -1;
    int fellowID = -1;
    protected void Page_Load(object sender, EventArgs e)
    {
        ///判断用户是否登录
        if(Session["UserID"] == null)
        {
            Response.Redirect("~/Login.aspx");
            return;
        }
        ///获取用户的 ID 值
        userID = Int32.Parse(Session["UserID"].ToString());
        if(Request.Params["UserID"] != null)
        {
            fellowID = Int32.Parse(Request.Params["UserID"].ToString());
        }
        if(!Page.IsPostBack && userID>0 && fellowID>0)
        {
            BindPageData(userID, fellowID);
        }
    }
    private void BindPageData(int userID, int fellowID)
    {
```

```

///获取好友信息
ASPNETAJAXWeb.AjaxInstantMessaging.User user =
    new ASPNETAJAXWeb.AjaxInstantMessaging.User();
SqlDataReader dr = user.GetSingleUser(fellowID);
if(dr == null) return;
if(dr.Read())
{
    ///显示好友名称
    lbUsername.Text = dr["Username"].ToString();
}
dr.Close();
}
protected void btnCommit_Click(object sender, EventArgs e)
{
    ///判断上传文件的内容是否为空
    if(fuFile.HasFile == false || fuFile.PostedFile.ContentLength<=0)
    {
        lbMessage.Text = "上传文件的内容为空, 请重新选择文件! ";
        return;
    }
    ///获取上传文件的属性, 如类型、大小、名称等
    string type = fuFile.PostedFile.ContentType;
    int size = fuFile.PostedFile.ContentLength;
    string oldFileName =
        Path.GetFileNameWithoutExtension(fuFile.PostedFile.FileName);
    ///创建基于时间的文件名称
    string fileName = AjaxInstantMessagingSystem.CreateDateTimeString();
    string extension = Path.GetExtension(fuFile.PostedFile.FileName);
    ///构建保存文件位置的路径
    string url = "Files/" + fileName + extension;
    ///映射为物理路径
    string fullPath = Server.MapPath(url);
    ///判断文件是否存在
    if(System.IO.File.Exists(fullPath) == true)
    {
        lbMessage.Text = "上传文件的已经存在, 请重新选择文件! ";
        return;
    }
    try
    {
        ///上传文件
        fuFile.SaveAs(fullPath);
        ASPNETAJAXWeb.AjaxInstantMessaging.File file =
            new ASPNETAJAXWeb.AjaxInstantMessaging.File();
        ///添加到数据库中
        if(file.AddFile(oldFileName, userID, fellowID, url, type, size) > 0)
        {
            lbMessage.Text = "恭喜您, 发送文件(" + oldFileName + ")给团队"
                + lbUsername.Text + "成功.";
        }
    }
    catch(Exception ex)
    {
        ///显示错误信息
        lbMessage.Text = "上传文件错误, 错误原因为: " + ex.Message;
        return;
    }
}
}

```

## 8.12 项目调试

视频讲解 光盘：视频\第8章\项目调试.avi

系统登录表单页面的效果如图 8-6 所示。



图 8-6 系统登录表单页面的效果

系统主页的显示效果如图 8-7 所示。



图 8-7 系统主页的效果

系统在线交流界面的效果如图 8-8 所示。

发送文件界面的效果如图 8-9 所示。

查看信息界面的效果如图 8-10 所示。



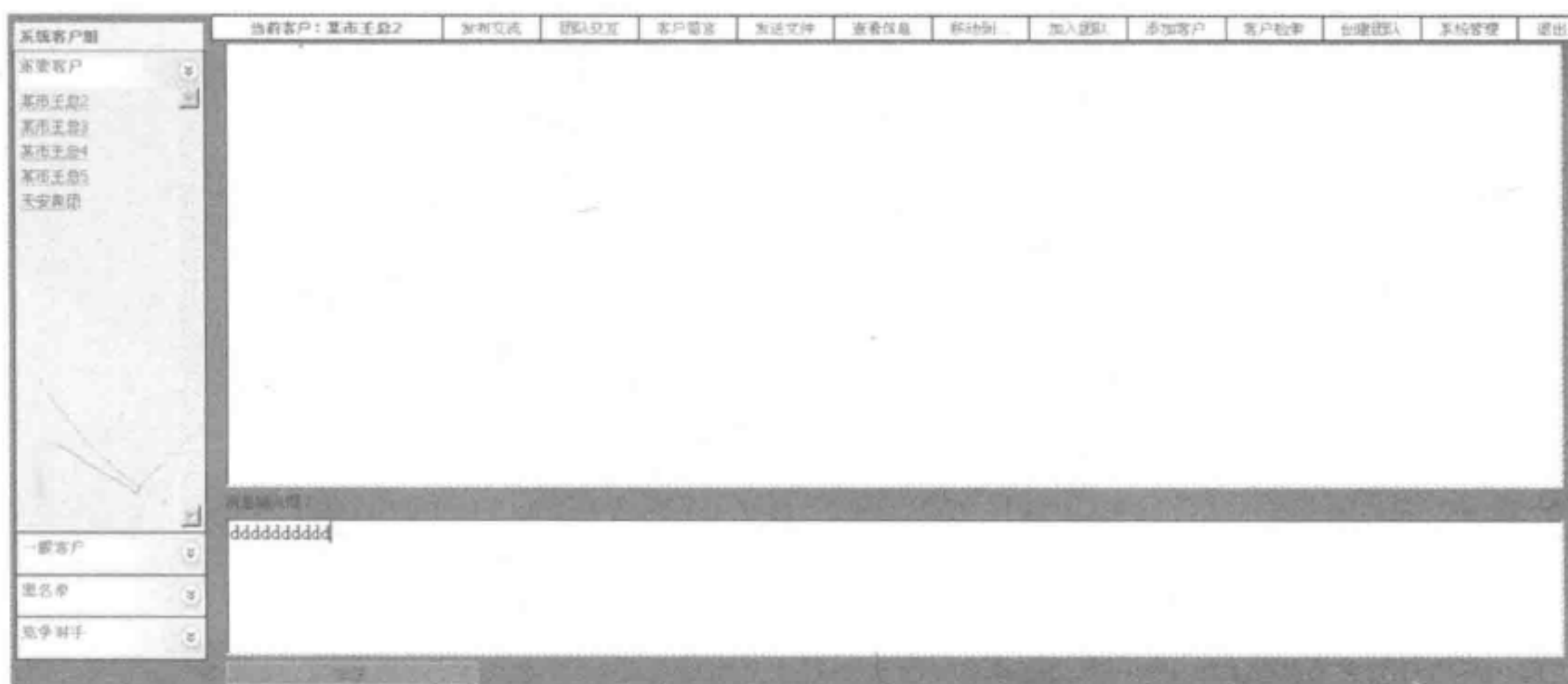


图 8-8 在线交流界面的效果

|       |                                   |
|-------|-----------------------------------|
| 发送到:  | 客户。                               |
| 选择文件: | <input type="text"/> 浏览...        |
|       | <input type="button" value="发送"/> |

图 8-9 发送文件界面的效果

|       |                                   |
|-------|-----------------------------------|
| 用户名称: | 某市王总2                             |
| 用户别名: | User2                             |
| 惟一标识: | 100002                            |
| 电子邮件: | admin@web.com                     |
|       | <input type="button" value="关闭"/> |

图 8-10 查看信息界面的效果



## 第9章 餐饮管理系统

高帆项目开发

餐饮管理系统是根据我国餐饮行业的现状及未来发展的趋势开发的一套基于 SaaS(软件即服务)的计算机管理系统。系统包含餐饮管理的各个方面,是餐饮实现管理科学化、信息化、现代化的重要标志。该系统使餐厅的餐台预定、消费管理、前台接待、收银结账、财务处理、营业查询、餐台使用情况更新等全面信息化,大大提高了工作效率,减少了因人为因素造成的差错及不必要的成本,提高了餐厅的档次、服务水平及销售收入。

在本章的内容中,将使用 C#语言,在 Visual Studio + SQL Server 平台上开发一个完整的餐饮管理系统。



### 赠送的超值电子书

- 081 using 指令
- 082 内部类
- 083 分布类
- 084 面向对象的计算模型和可计算性的联系
- 085 常量与只读字段的区别
- 086 使用 get 和 set 访问器的注意事项
- 087 索引器与属性的比较
- 088 不允许派生类的可访问性高于基类的可访问性
- 089 不要在密封类型中声明虚拟成员
- 090 不要在密封类型中声明受保护的成员



## 9.1 考虑所有可能会发生的情形

视频讲解 光盘：视频\第9章\考虑所有可能会发生的情形

一款好的软件程序，能够面对所有可能发生的情形而获得正确的结果。在评价软件质量高低的规范中，健壮性是其中重要的一条。在本节的内容中，将引领读者一起探讨提高程序健壮性的奥秘。

### 9.1.1 一段代码所引发的思考

曾经在了一本经典教科书中看到过一则教程，里面有位老师提到了印度人的软件业要比中国的好，其中的原因除了印度人母语是英语外，更重要的，是因为“印度人严谨”，他们的程序更有健壮性。印度的一个老程序员，月代码量在一千行左右，这一千行代码，算法平实，但都是经过仔细推敲和实战检验的代码，不会轻易崩溃。而国内的很多程序员，一天就可以写出一千行代码，写的代码简短精干，算法非常有技巧性，但往往是不安全的、不完善的。

平常写一段功能性的程序，可能一百行代码即可实现；但如果写一段健壮的程序，至少需要 300 行代码。例如写一个房贷计算器程序，算法十分简单，十多行就完成了。在提示用户输入金额一栏中，要求从用户界面读取利率、年限和贷款额三个数据，大多数人的写法十分简单，只需如下一句代码即可：

```
doubleNum = Double.parseDouble(JOptionPane.showInputDialog(null, "请输入"+StrChars));
```

但是，上述程序完全不具备健壮性，因为现实中，输入的金额字符是不受限制的，输入的金额字符会有很多种情形，例如：

- 输入了负数。
- 输入超出了 double 类型所能涵盖的范围。
- 输入了标点符号。
- 输入了中文。
- 没有任何输入。
- 选择了取消或者单击了右上角的关闭按钮。

上述这一切都是有可能发生的情形，而且超出了程序的处理范围。这种情形本不该发生，但是在使用程序时，一切输入都是有可能的。作为一个程序员，如何让自己代码在执行的时候确保输入字符的合法性呢？可以编写一个独立的方法来验证输入的数据，限定输入的只能是正实数，否则就报错，需要用户单击取消或者关闭按钮。这个验证方法就是为了提高程序的健壮性而推出的，由此可见，程序的健壮性就是要求我们的程序需要考虑各种各样的运行环境和情形。

### 9.1.2 赢在项目——提高程序的健壮性

程序的健壮性是指在异常情况下，软件能够正常运行的能力。正确性与健壮性的区别

是：前者描述软件在需求范围之内的行为，而后者描述软件在需求范围之外的行为。可是正常情况与异常情况并不容易区分，开发者往往要么没想到异常情况，要么把异常情况错当成正常情况而不做处理，结果降低了健壮性。要知道用户是不管正确性与健壮性的区别的，反正软件出了差错都是开发方的错。所以，提高软件的健壮性也是开发者的义务。

程序的健壮性有两层含义：一是容错能力，二是恢复能力。

容错是指发生异常情况时系统不出错误的能力。对于应用于航空航天、武器、金融等领域的这类高风险系统，容错设计非常重要。容错是非常健壮的意思，比如 Unix 的容错能力很强，很难使系统出问题。而恢复则是指软件发生错误后(不论死活)，重新运行时，能否恢复到没有发生错误前的状态的能力。

恢复能力从语义上理解，不及容错那么健壮。例如，同样是被打伤，非常健壮的人一点事儿都没有(表示有容错能力)；比较健壮的人，虽然被打倒在地，过了一会儿还能爬起来，除了皮肉之痛外，倒也不用去医院(表示恢复能力比较强)；而虚弱的人可能短期恢复不过来，得在病床上躺很久。

恢复能力是很有价值的。Microsoft 公司早期的窗口系统如 Windows 3.x 和 Windows 9x，动不动就死机，其容错性的确比较差。但它们的恢复能力还不错，机器重新启动后，一般都能正常运行(看在这个份儿上，当时人们也愿意将就着用)。

假如一个软件可以正确地运行在不同的环境下，则认为软件可移植性高，也可以叫“软件在不同平台下是健壮的”。

一个软件能够检测自己内部的设计或者编码错误，并得到正确的执行结果，这是软件的正确性标准，也可以说，软件有内部的保护机制，是模块级健壮的。

软件健壮性是一个比较模糊的概念，但是，却是非常重要的软件外部量度标准。软件设计得健壮与否，直接反映了分析设计和编码人员的水平，即所谓的“高手写的程序不容易死”。

那么，究竟如何提高程序的健壮性呢？需要从如下所示的三个方面着手。

#### (1) 解决面向对象要求的黑箱操作

由于面向对象要求各个部分是彼此独立的，所以各个部分就要足够强劲，以应付输入参数的不合理性。虽然现代编程都讲究预处理，一般是将输入格式转换为统一的格式，然后进行处理(比如说，现在网上的搜索引擎，都是将输入转换为 Unicode 的格式)，但这并不是说我们的处理函数就不需要错误处理了。预处理能够大大减少程序出错的概率和编写错误处理的复杂度。

但是，考虑到单独模块越来越趋向于智能化，各个黑箱应该具有独立的行为、错误处理，以及错误纠正的功能。

#### (2) 实现错误捕捉和错误信息

C#语言的错误处理机制比较健全，这里需要注意的问题，是如何书写错误信息的问题。错误信息要完整：包括在什么地方，因为什么，出现了什么样的错误。不完整的错误信息没有任何可用价值。

另外一点就是使用运行日志(Log)。在关键的步骤上输入一些信息到 Log 文件内，提示当前程序运行到什么地方去了(如有可能，得到系统的当前错误码)。这样，当程序意外中断的时候，就可以使用这个 Log 进行一定的判断。



### (3) 实现程序的自我防御，预防二义性

好的程序应该是尽可能自动纠错的，这在程序的输入不可预测的情况下尤为重要。其实，由于合作开发的原因，这样的情况很多。一个程序员编写的模块很有可能对输入有特定的要求，那么当另外一个程序员调用这个模块的时候，就会出现问題。

实现程序的自我防御的处理办法，是在模块内对输入进行判断，如果有二义性，则进行合理纠错，并有效地提示(在 Debug 版本下)。这种解决办法通常是在循环语句中使用 if 语句来处理二义性的情形。其中最简单的方法是，在 if 语句中设置一旦程序有任何错误发生，就退出当前的程序或单个线程。

## 9.2 新的项目

视频讲解 光盘：视频\第 9 章\新的项目.avi

本餐饮管理系统项目的客户提出了如下两点要求。

- (1) 需要登录验证，确保合法人员才能使用这个系统。
- (2) 能够实现开台、点菜和结账处理。

本项目是老同学获得的私活儿，整个团队成员如下所示。

- 软件工程师 A：负责前期功能分析，策划系统模块，做系统设计，数据库设计。
- 软件工程师 B：负责整个项目的窗体界面设计、具体编码、后期项目调试和发布。

整个项目的实现流程如图 9-1 所示。

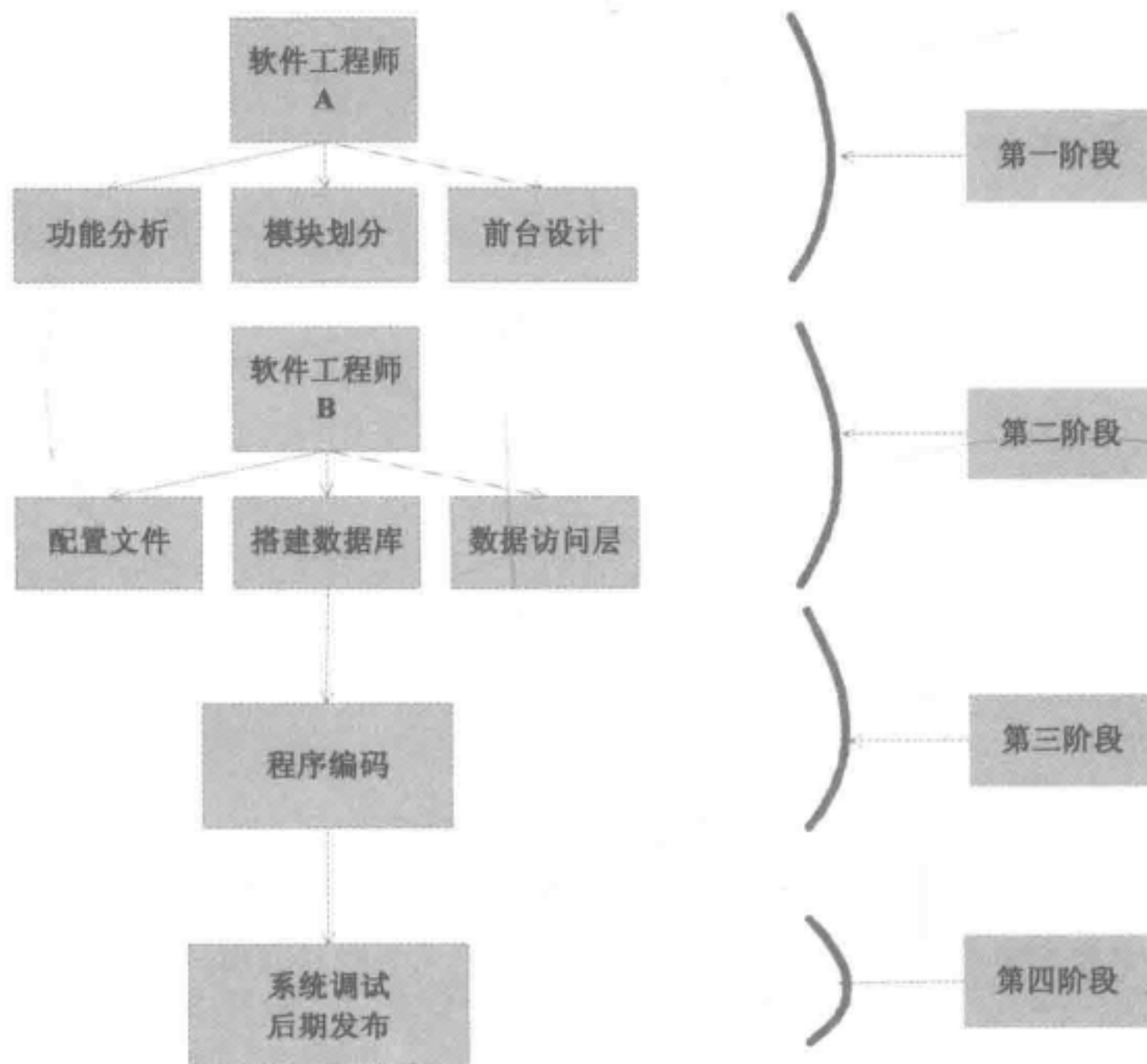


图 9-1 项目的实现流程



## 9.3 项目规划分析

视频讲解 光盘：视频\第9章\项目规划分析.avi

餐饮系统是一个综合性的系统，不仅涉及用餐管理，而且还会涉及到用餐、订餐等一些操作，幸亏这些操作都是基于数据库数据处理的，所以比较容易实现。

### 9.3.1 开发背景

近年来，随着计算机网络技术的飞速发展，餐饮业的竞争也越来越激烈。要想在激烈竞争环境下生存，必须使用科学的管理思想，将点餐和管理实现一体化操作。这样会大大提高工作效率，避免原来的手工操作的麻烦，从而使整个过程更加精确和有效。

### 9.3.2 项目模块分析

作为一个餐饮管理系统，应该具有如下功能：

- 用人机交互的方式，实现界面友好、信息灵活、查询方便、数据安全可靠。
- 具备对餐厅顾客开台、点菜、加菜、账目查询和结账等功能。
- 能够对顾客的消费历史进程进行查询。
- 最大限度地实现易维护性和易操作性。

### 9.3.3 构成模块

一个典型的餐饮管理系统的基本模块结构如图 9-2 所示。



图 9-2 餐饮管理系统模块的结构

## 9.4 搭建数据库

视频讲解 光盘：视频\第9章\搭建数据库.avi

考虑到用户上传的图片可能会比较多,所以决定采用 SQL Server 2008 作为数据库工具,并创建了一个名为“eat”的数据库。

### 9.4.1 数据库概念设计

根据系统需求,规划出整个系统需要的实体,有商品信息实体、商品类别信息实体、顾客消费信息实体、桌台信息实体、用户信息实体、职员信息实体。

(1) 商品信息实体的 E-R 图如图 9-3 所示。

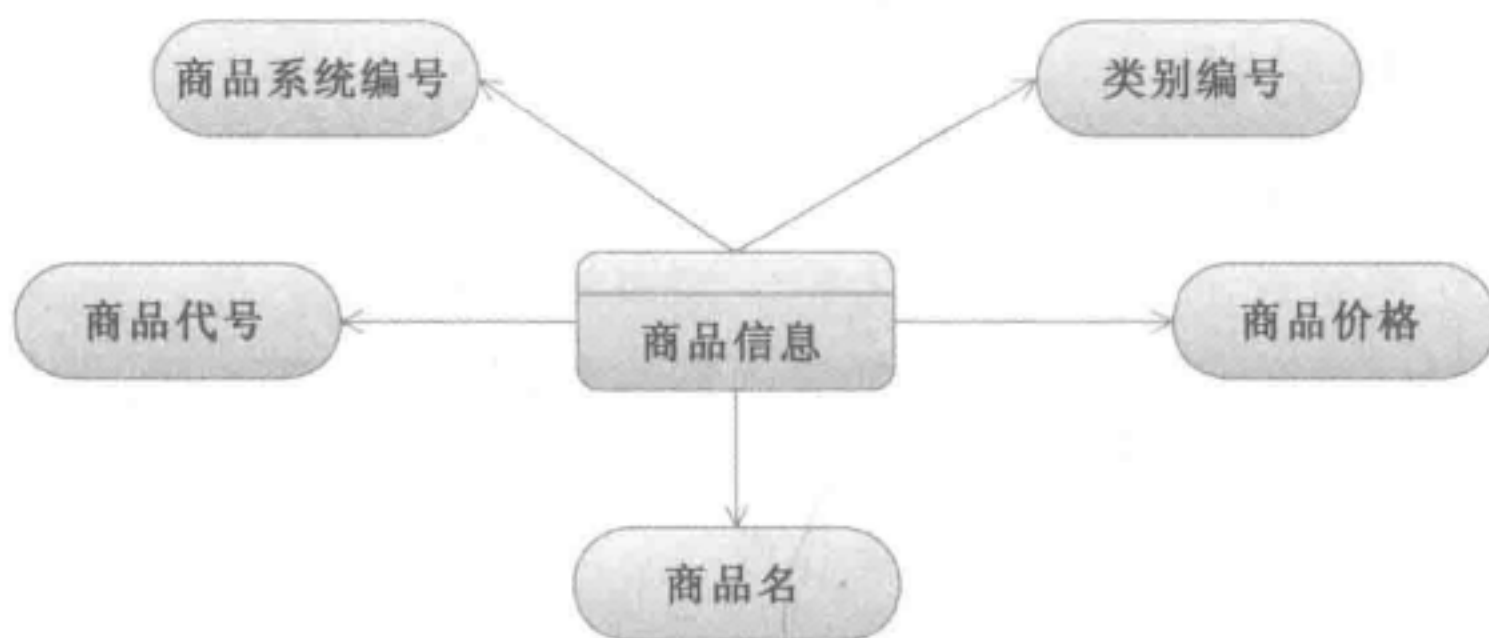


图 9-3 商品信息实体的 E-R 图

(2) 顾客消费信息实体的 E-R 图如图 9-4 所示。



图 9-4 顾客消费信息实体的 E-R 图

(3) 桌台信息实体的 E-R 图如图 9-5 所示。



图 9-5 桌台信息实体的 E-R 图

(4) 职员信息实体的 E-R 图如图 9-6 所示。



图 9-6 职员信息实体的 E-R 图

(5) 用户信息实体的 E-R 图如图 9-7 所示。

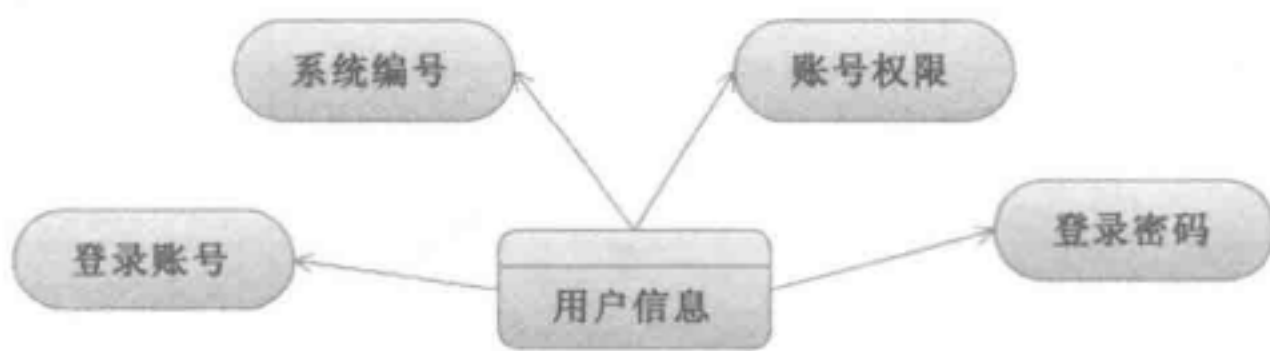


图 9-7 用户信息实体的 E-R 图

(6) 商品类别信息实体的 E-R 图如图 9-8 所示。



图 9-8 商品类别信息实体的 E-R 图



9.4.2 数据库逻辑结构设计

(1) 表 tb\_foodtype 用于保存商品类别信息，具体的设计结构如表 9-1 所示。

表 9-1 表 tb\_foodtype

| 字 段 名    | 数据类型        | 描 述 |
|----------|-------------|-----|
| ID       | int         | 编号  |
| foodtype | varchar(50) | 类别名 |

(2) 表 tb\_food 用于保存商品类别信息，具体的设计结构如表 9-2 所示。

表 9-2 表 tb\_food

| 字 段 名     | 数据类型           | 描 述  |
|-----------|----------------|------|
| ID        | int            | 编号   |
| foodty    | char(10)       | 类别编号 |
| foodnum   | char(10)       | 商品代号 |
| foodname  | varchar(50)    | 商品名  |
| foodprice | decimal(18, 0) | 商品价格 |

(3) 表 tb\_GuestFood 用于保存顾客的消费信息，具体的设计结构如表 9-3 所示。

表 9-3 表 tb\_GuestFood

| 字 段 名        | 数据类型           | 描 述   |
|--------------|----------------|-------|
| ID           | int            | 编号    |
| foodnum      | char(10)       | 商品代号  |
| foodname     | varchar(50)    | 商品名   |
| foodsum      | char(10)       | 消费数量  |
| foodallprice | decimal(18, 0) | 商品价格  |
| waitername   | varchar(50)    | 操作员姓名 |
| beizhu       | varchar(50)    | 备注    |
| zhuotai      | char(10)       | 消费桌台  |
| datetime     | varchar(50)    | 消费时间  |

(4) 表 tb\_Room 用于保存桌台信息，具体的设计结构如表 9-4 所示。

表 9-4 表 tb\_Room

| 字 段 名    | 数据类型        | 描 述 |
|----------|-------------|-----|
| ID       | char(10)    | 编号  |
| RoomName | char(10)    | 名称  |
| RoomJC   | varchar(50) | 简称  |

续表

| 字 段 名        | 数据类型           | 描 述   |
|--------------|----------------|-------|
| RoomBJF      | decimal(18, 0) | 包间费   |
| RoomWZ       | char(10)       | 位置    |
| RoomZT       | char(10)       | 状态    |
| RoomType     | varchar(50)    | 类型    |
| RoomBZ       | varchar(50)    | 备注    |
| RoomQT       | varchar(50)    | 其他信息  |
| GuestName    | varchar(50)    | 顾客姓名  |
| zhangdanDate | varchar(50)    | 开台时间  |
| Num          | int            | 人数    |
| WaiterName   | varchar(50)    | 操作员姓名 |

(5) 表 tb\_User 用于保存系统用户信息，具体的设计结构如表 9-5 所示。

表 9-5 表 tb\_User

| 字 段 名    | 数据类型        | 描 述 |
|----------|-------------|-----|
| ID       | int         | 编号  |
| UserName | varchar(50) | 登录名 |
| UserPwd  | varchar(50) | 密码  |
| power    | char(10)    | 权限  |

(6) 表 tb\_Waiter 用于保存职员信息，具体的设计结构如表 9-6 所示。

表 9-6 表 tb\_Waiter

| 字 段 名      | 数据类型        | 描 述  |
|------------|-------------|------|
| ID         | int         | 系统编号 |
| WaiterName | varchar(50) | 职员名字 |
| CardNum    | varchar(50) | 身份证号 |
| WaiterNum  | char(10)    | 职员编号 |
| Sex        | char(10)    | 性别   |
| Age        | char(10)    | 年龄   |
| Tel        | varchar(50) | 电话   |

在上述设计数据库的过程中，整个处理过程还是基于数据库这个中间存储媒介的，流程如下。

- (1) 查询系统数据库，将某一条信息以表单方式显示出来。
- (2) 在窗体中显示或修改某条信息。
- (3) 能够删除数据库中的某条信息。

## 9.5 设计窗体

视频讲解 光盘：视频\第9章\设计窗体.avi

### 1. 登录界面

登录界面实现用户登录验证，老同学的设计方案如图 9-9 所示。



图 9-9 登录界面窗体

### 2. 日历界面

日历界面显示一个日历信息，老同学的设计方案如图 9-10 所示。

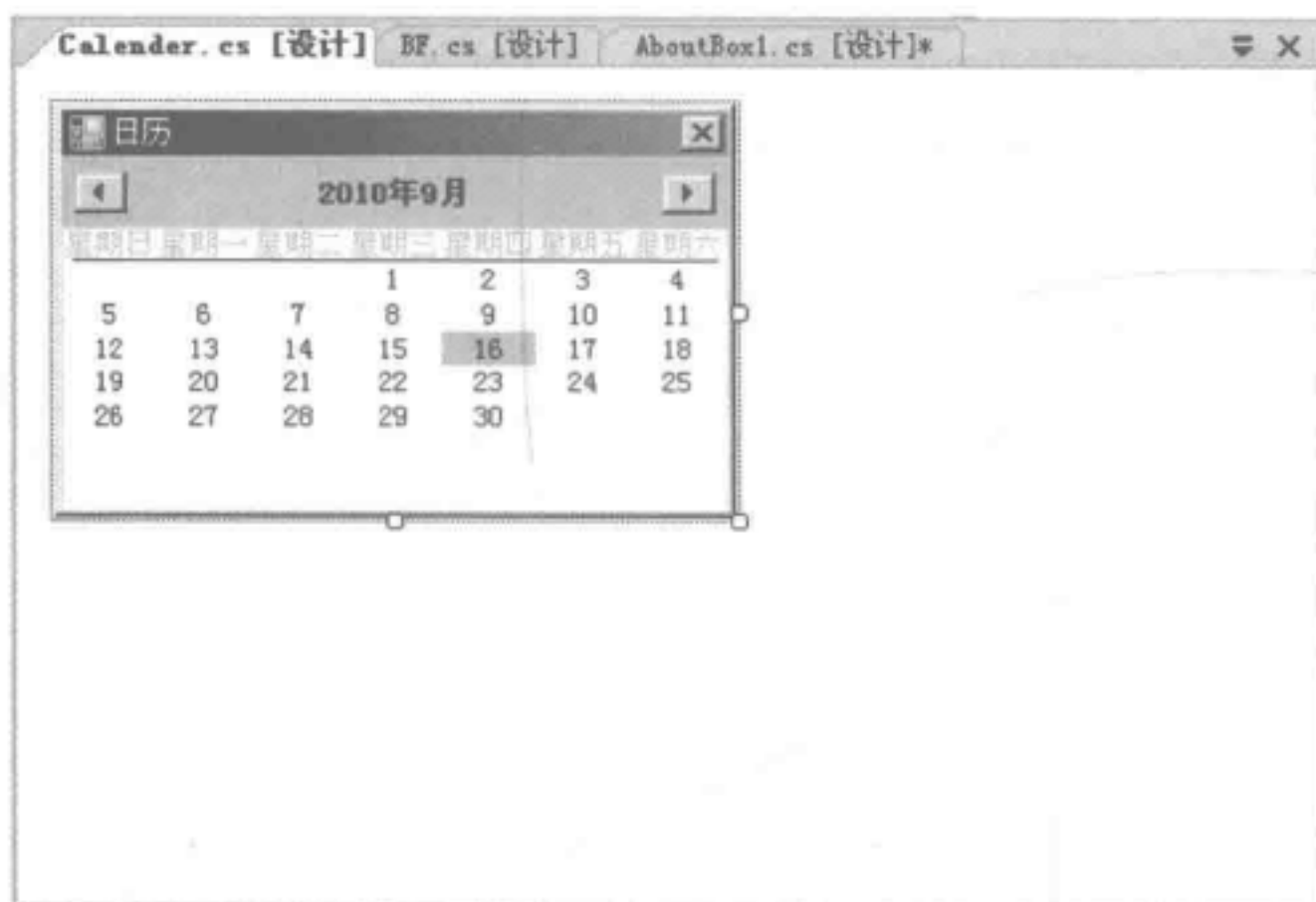


图 9-10 日历界面窗体

### 3. 菜单界面

菜单界面显示相关的菜单信息，老同学的设计方案如图 9-11 所示。





图 9-11 菜单界面窗体

#### 4. 桌台界面

桌台界面显示桌台的具体信息，老同学的设计方案如图 9-12 所示。



图 9-12 桌台界面窗体

#### 5. 结账界面

结账界面显示具体的结账信息，老同学的设计方案如图 9-13 所示。

#### 6. 主界面

主界面是系统执行并登录后默认显示的主界面，老同学的设计方案如图 9-14 所示。

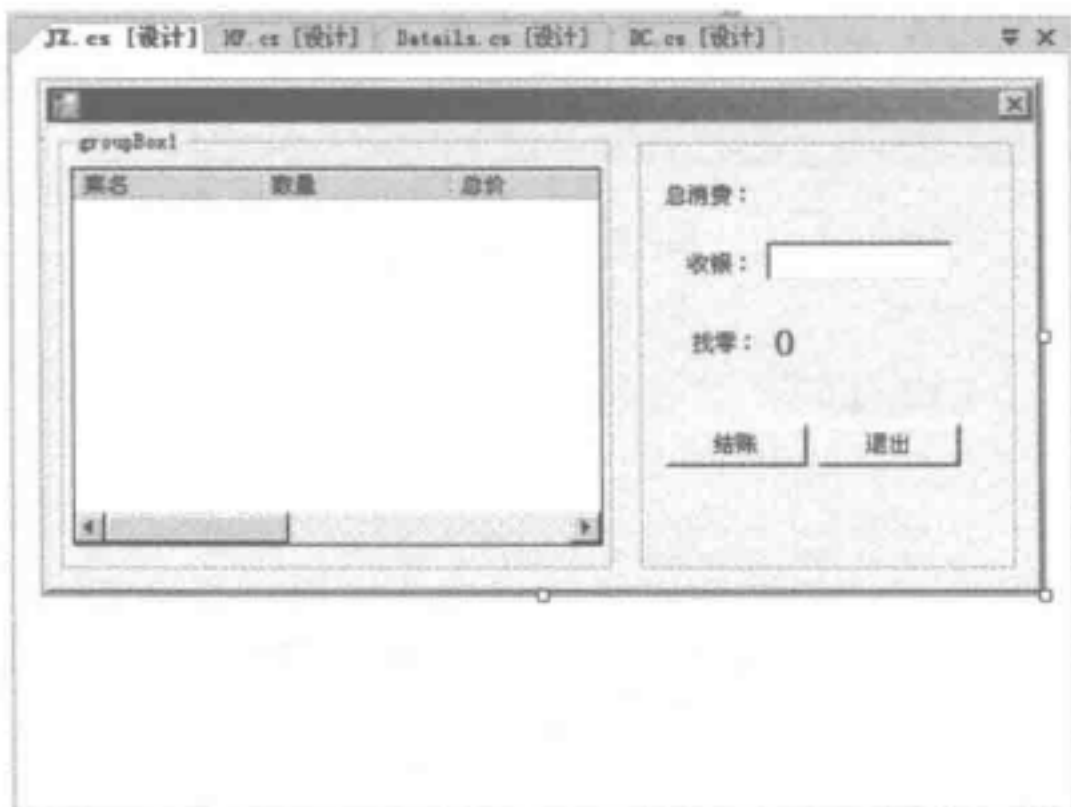


图 9-13 结账界面窗体

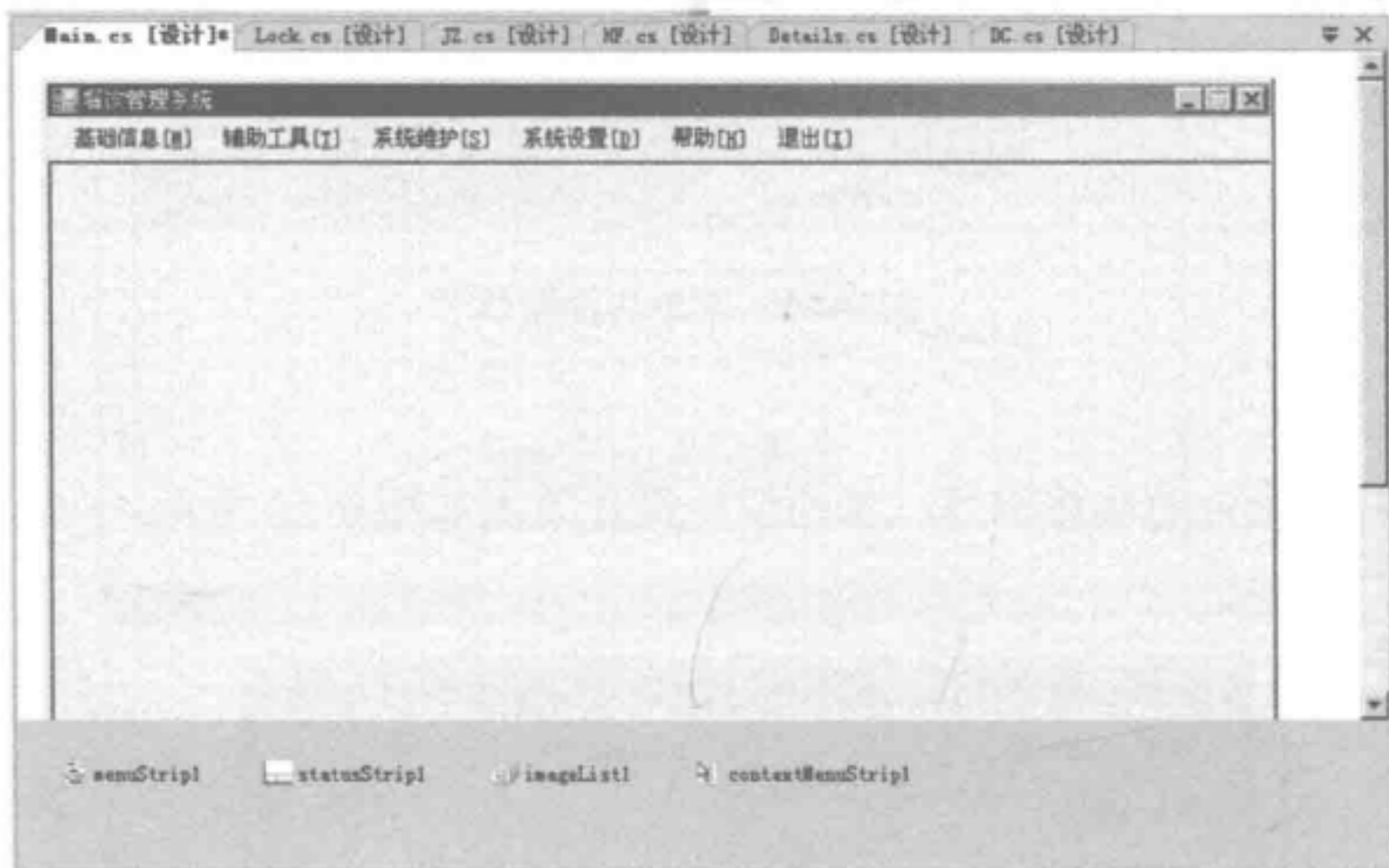


图 9-14 系统主界面窗体

## 9.6 具体编码

视频讲解 光盘：视频\第 9 章\具体编码.avi

在具体编码之前，一定要仔细做好系统可扩展性的准备工作，避免有意外发生。在本节的内容中，将详细讲解本项目的具体编码过程。

### 9.6.1 数据库连接

首先编写数据流连接代码，实现文件是 DBConn.cs，对应的代码如下所示：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
namespace MrCy.BaseClass
{
    class DBConn
```

```
{
    public static SqlConnection CyCon()
    {
        return new SqlConnection("server=(local);database=eat;uid=sa;pwd=888888");
    }
}
```

## 9.6.2 登录模块

此模块的功能，是确保只有是系统的合法用户才能登录本系统。本模块的实现文件是 Login.cs，下面开始讲解其具体的实现流程。

(1) 引入命名空间，确保能够使用 SQL Server 数据库。对应的代码如下所示：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

(2) 当单击“登录”按钮后，首先验证是否输入了用户名和密码，没有则输出提示；如果输入了，则要对输入的信息进行验证，只有是合法的用户才能登录系统。对应的代码如下所示：

```
private void btnSubmit_Click(object sender, EventArgs e)
{
    if (txtName.Text == "")
    {
        MessageBox.Show(
            "请输入用户名", "警告", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        if (txtPwd.Text == "")
        {
            MessageBox.Show(
                "请输入密码", "警告", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
        else
        {
            SqlConnection conn = BaseClass.DBConn.CyCon();
            conn.Open();
            SqlCommand cmd =
                new SqlCommand("select count(*) from tb_User where UserName='"
                    + txtName.Text + "' and UserPwd='" + txtPwd.Text + "'", conn);
            int i = Convert.ToInt32(cmd.ExecuteScalar());
            if (i > 0)
            {
                cmd = new SqlCommand("select * from tb_User where UserName='"
                    + txtName.Text + "'", conn);
                SqlDataReader sdr = cmd.ExecuteReader();
                sdr.Read();
                string UserPower = sdr["power"].ToString().Trim();
                conn.Close();
                Main main = new Main();
            }
        }
    }
}
```



```

        main.power = UserPower;
        main.Names = txtName.Text;
        main.Times = DateTime.Now.ToShortDateString();
        main.Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("用户名或密码错误");
    }
}
}

```

(3) 输入合法数据后能够登录系统，按下 Enter 键后，会触发处理事件，进入系统。对应的代码如下所示：

```

private void txtPwd_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        btnSubmit_Click(sender, e);
    }
}

```

### 9.6.3 主窗体模块

主窗体是用户登录后显示的主界面，主要分为如下 3 部分。

- 顶部菜单：实现一些对应的操作处理。
- 中间桌台显示：显示系统内的桌台信息。
- 底部状态信息：显示系统的当前状态信息。

主界面的效果如图 9-15 所示。

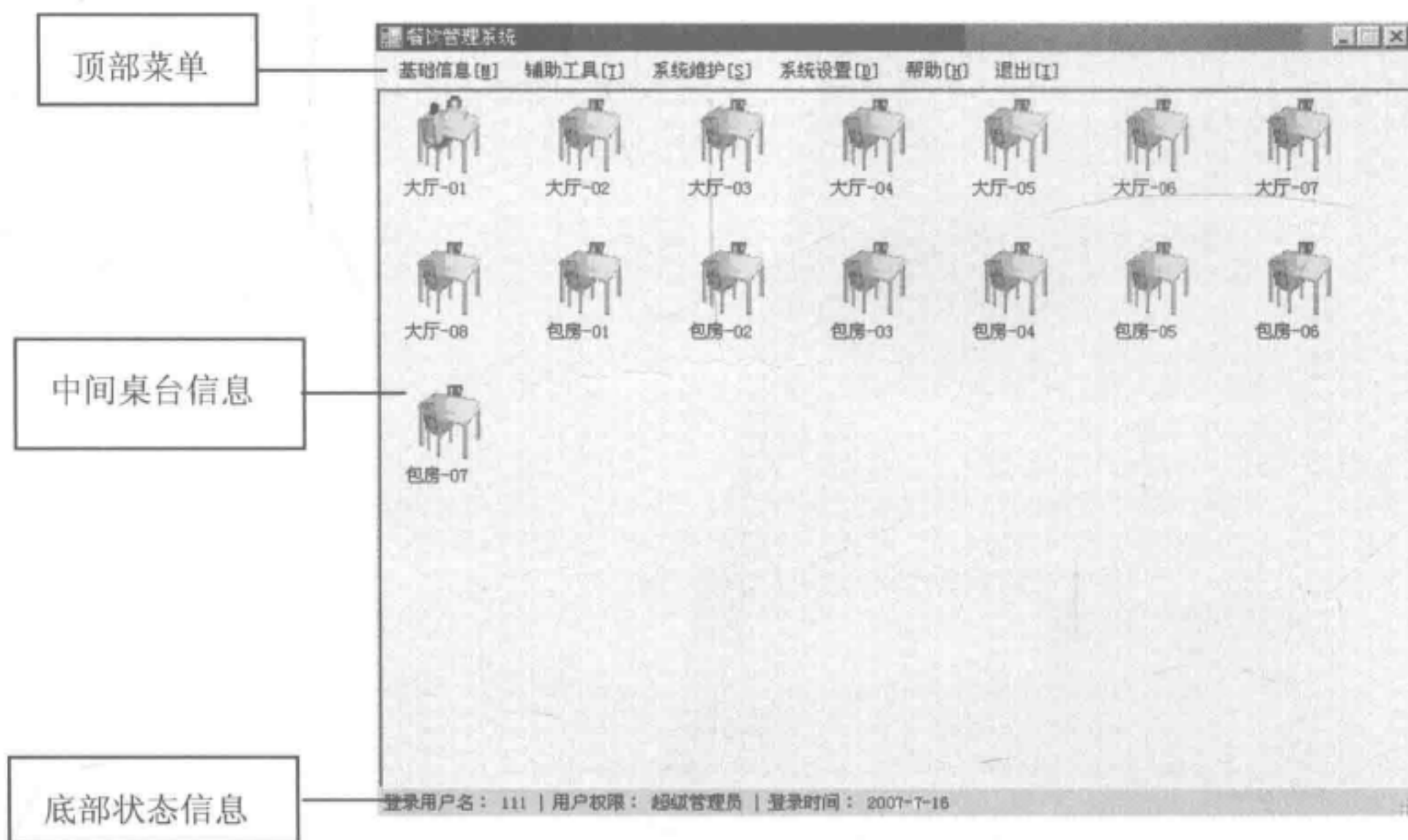


图 9-15 系统主界面

主界面的实现文件是 Main.cs, 下面开始讲解其具体的实现流程。

(1) 定义4个公共变量, 对应的代码如下所示:

```
namespace MrCy
{
    public partial class Main : Form
    {
        public Main()
        {
            InitializeComponent();
        }
        public SqlDataReader sdr;
        public string power;
        public string Names;
        public string Times;
    }
}
```

(2) 定义加载窗体事件, 首先判断用户权限, 并根据不同的权限实现不同的功能。对应的代码如下所示:

```
private void frmMain_Load(object sender, EventArgs e)
{
    switch (power)
    {
        case "0": toolStripStatusLabel13.Text = "超级管理员"; break;
        case "1": toolStripStatusLabel13.Text = "经理"; break;
        case "2": toolStripStatusLabel13.Text = "一般用户"; break;
    }
    toolStripStatusLabel10.Text = Names;
    toolStripStatusLabel16.Text = Times;
    if (power == "2")
    {
        系统维护 SToolStripMenuItem.Enabled = false;
        基础信息 MToolStripMenuItem.Enabled = false;
    }
    if (power == "1")
    {
        系统维护 SToolStripMenuItem.Enabled = false;
    }
}
```

(3) 当窗体焦点被激发时, 系统从数据库中检索所有的桌台状态信息, 并调用自定义的 AddItems(zt)方法为 ListView 控件添加项目。对应的代码如下所示:

```
private void frmMain_Activated(object sender, EventArgs e)
{
    lvDesk.Items.Clear();
    SqlConnection conn = BaseClass.DBConn.CyCon();
    conn.Open();
    SqlCommand cmd = new SqlCommand("select * from tb_Room", conn);
    sdr = cmd.ExecuteReader();
    while (sdr.Read())
    {
        string zt = sdr["RoomZT"].ToString().Trim();
        AddItems(zt);
    }
    conn.Close();
}
```

(4) 自定义方法 AddItems(string rzt), 根据不同状态为 ListView 添加不同的图片。对应

的代码如下所示:

```
private void AddItems(string rzt)
{
    if (rzt == "使用")
    {
        lvDesk.Items.Add(sdr["RoomName"].ToString(), 1);
    }
    else
    {
        lvDesk.Items.Add(sdr["RoomName"].ToString(), 0);
    }
}
```

(5) 定义处理事件 lvDesk\_Click(object sender, EventArgs e), 当用户右击某个桌台时, 会根据桌台的状态弹出不同的右键菜单。对应的代码如下所示:

```
private void lvDesk_Click(object sender, EventArgs e)
{
    string names = lvDesk.SelectedItems[0].SubItems[0].Text;
    SqlConnection conn = BaseClass.DBConn.CyCon();
    conn.Open();
    SqlCommand cmd =
        new SqlCommand("select * from tb_Room where RoomName='" + names + "'", conn);
    SqlDataReader sdr = cmd.ExecuteReader();
    sdr.Read();
    string zt = sdr["RoomZT"].ToString().Trim();
    sdr.Close();
    if (zt == "使用")
    {
        this.contextMenuStrip1.Items[0].Enabled = false;
        this.contextMenuStrip1.Items[1].Enabled = true;
        this.contextMenuStrip1.Items[3].Enabled = true;
        this.contextMenuStrip1.Items[5].Enabled = true;
        this.contextMenuStrip1.Items[6].Enabled = true;
    }
    if (zt == "待用")
    {
        this.contextMenuStrip1.Items[0].Enabled = true;
        this.contextMenuStrip1.Items[1].Enabled = false;
        this.contextMenuStrip1.Items[3].Enabled = false;
        this.contextMenuStrip1.Items[5].Enabled = false;
        this.contextMenuStrip1.Items[6].Enabled = false;
    }
    conn.Close();
}
```

(6) 最后讲解单击顶部菜单后的对应处理事件和中间界面右键单击菜单命令的对应处理事件, 对应的代码如下所示:

```
private void 开台ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (lvDesk.SelectedItems.Count != 0)
    {
        string names = lvDesk.SelectedItems[0].SubItems[0].Text;
        Open openroom = new Open();
        openroom.name = names;
        openroom.ShowDialog();
    }
    else
```



```

{
    MessageBox.Show("请选择桌台");
}
}
private void 点菜ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (lvDesk.SelectedItems.Count != 0)
    {
        string names = lvDesk.SelectedItems[0].SubItems[0].Text;
        DC dc = new DC();
        dc.RName = names;
        dc.ShowDialog();
    }
    else
    {
        MessageBox.Show("请选择桌台");
    }
}
private void 消费查询ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (lvDesk.SelectedItems.Count != 0)
    {
        string names = lvDesk.SelectedItems[0].SubItems[0].Text;
        Serch serch = new Serch();
        serch.RName = names;
        serch.ShowDialog();
    }
    else
    {
        MessageBox.Show("请选择桌台");
    }
}
private void 结账ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (lvDesk.SelectedItems.Count != 0)
    {
        string names = lvDesk.SelectedItems[0].SubItems[0].Text;
        JZ jz = new JZ();
        jz.Rname = names;
        jz.ShowDialog();
    }
    else
    {
        MessageBox.Show("请选择桌台");
    }
}
private void 取消开台toolStripMenuItem_Click(object sender, EventArgs e)
{
    if (lvDesk.SelectedItems.Count != 0)
    {
        string names = lvDesk.SelectedItems[0].SubItems[0].Text;
        SqlConnection conn = BaseClass.DBConn.CyCon();
        conn.Open();
        SqlCommand cmd = new SqlCommand(
            "update tb_Room set RoomZT='待用',Num=0 where RoomName='" + names + "'", conn);
        cmd.ExecuteNonQuery();
        cmd = new SqlCommand(
            "delete from tb_GuestFood where zhuotai='" + names + "'", conn);
        cmd.ExecuteNonQuery();
        conn.Close();
        frmMain_Activated(sender, e);
    }
}

```

```
    }
    else
    {
        MessageBox.Show("请选择桌台");
    }
}

private void 桌台信息ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    Desk desk = new Desk();
    desk.ShowDialog();
}

private void 职员信息ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    User users = new User();
    users.ShowDialog();
}

private void 日历ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    Calender calender = new Calender();
    calender.ShowDialog();
}

private void 记事本ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    System.Diagnostics.Process.Start("notepad.exe");
}

private void 计算器ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    System.Diagnostics.Process.Start("calc.exe");
}

private void 权限管理ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    QxGl qx = new QxGl();
    qx.ShowDialog();
}

private void 系统备份ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    BF bf = new BF();
    bf.ShowDialog();
}

private void 系统恢复ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    HF hf = new HF();
    hf.ShowDialog();
}

private void 口令设置ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    Pwd pwd = new Pwd();
    pwd.names = Names;
    pwd.ShowDialog();
}

private void 锁定系统ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    Lock locksystem = new Lock();
    locksystem.Owner = this;
    locksystem.ShowDialog();
}

private void 关于ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    AboutBox1 ab = new AboutBox1();
    ab.ShowDialog();
}
```

```

}
private void 退出系统ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("确定退出本系统吗?", "提示", MessageBoxButtons.OKCancel,
        MessageBoxIcon.Exclamation) == DialogResult.OK)
    {
        Application.Exit();
    }
}
private void 系统维护ToolStripMenuItem_Click(object sender, EventArgs e)
{
}
private void lvDesk_SelectedIndexChanged(object sender, EventArgs e)
{
}

```

在接下来的内容中, 将总结在 C#窗体之间传值的几种方法。

方法 1, 所有权法:

```

//Form1:
//需要有一个公共的刷新方法
public void Refresh_Method()
{
    //...
}
//在调用 Form2 时, 要把 Form2 的所有者设为 Form1
Form2 f2 = new Form2();
f2.Owner = this;
f2.ShowDialog();
//Form2:
//在需要对其调用者(父)刷新时
Form1 f1;
f1 = (Form1)this.Owner;
f1.Refresh_Method();

```

方法 2, 自身传递法:

```

//Form1:
//需要有一个公共的刷新方法
public void Refresh_Method()
{
    //...
}
Form2 f2 = new Form2();
f2.ShowDialog(this);
//Form2:
private Form1 p_f1;
public Form2(Form1 f1)
{
    InitializeComponent();
    p_f1 = f1;
}
//刷新时
p_f1.Refresh_Method();

```

方法 3, 属性法:

```

//Form1:
//需要有一个公共的刷新方法
public void Refresh_Method()
{

```



```
//...
}
//调用时
Form2 f2 = new Form2();
f2.P_F1 = this;
f2.Show();

//Form2:
private Form1 p_f1;
public Form1 P_F1
{
    get{return p_f1;}
    set{p_f1 = value;}
}
//刷新时
p_f1.Refresh_Method();
```

#### 方法 4: 委托法

```
//声明一个委托
public delegate void DisplayUpdate();
//Form1:
//需要有一个公共的刷新方法
public void Refresh_Method()
{
    //...
}
//调用时
Form2 f2 = new Form2();
f2.ShowUpdate += new DisplayUpdate(Refresh_Method);
f2.Show();
//Form2:
//声明事件
public event DisplayUpdate ShowUpdate;
//刷新时, 放在需要执行刷新的事件里
ShowUpdate();
Form1 中放一个 Label1, 一个 Button1, Form1 为主窗口, 当单击 Button1 时, 执行:
Form2 f2 = new Form2();
f2.ShowDialog(this);
```

Form2 中放一个 TextBox1, 一个 Button1, 当单击 Button1 时执行:

```
Form1 f1 = (Form1)this.Owner;
f1.Label1.Text = this.TextBox1.Text;
```

### 9.6.4 开台模块

当顾客进行消费时, 首先会看是否有合适的位子。如果有空闲的, 则需要为顾客开台。在开台之后, 才可以继续进行点菜、查询和结账处理。本模块的实现文件是 Open.cs, 其实现流程如下。

- (1) 定义公共变量 name 和 conn, 以便在程序中调用。
  - (2) 加载窗体时, 将数据库中所有桌台信息和职员信息检索出来, 显示在 ComboBox 控件上。
  - (3) 输入用餐人数, 单击“保存”按钮之后, 即可对指定的桌台进行开台操作。
- 文件 Open.cs 的具体实现代码如下所示:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace MrCy
{
    public partial class Open : Form
    {
        public Open()
        {
            InitializeComponent();
        }
        public string name;
        public SqlConnection conn;
        private void frmOpen_Load(object sender, EventArgs e)
        {
            conn = BaseClass.DBConn.CyCon();
            conn.Open();
            SqlCommand cmd = new SqlCommand("select * from tb_Room", conn);
            SqlDataReader sdr = cmd.ExecuteReader();
            while (sdr.Read())
            {
                cbNum.Items.Add(sdr["RoomName"].ToString().Trim());
            }
            cbNum.SelectedItem = name.Trim();
            sdr.Close();
            cmd = new SqlCommand("select * from tb_Waiter", conn);
            sdr = cmd.ExecuteReader();
            while (sdr.Read())
            {
                cbWaiter.Items.Add(sdr["WaiterName"].ToString().Trim());
            }
            cbWaiter.SelectedIndex = 0;
            sdr.Close();
        }
        private void txtNum_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((e.KeyChar!=8 && !char.IsDigit(e.KeyChar)) && e.KeyChar!=13)
            {
                MessageBox.Show("请输入数字");
                e.Handled = true;
            }
        }
        private void btnSave_Click(object sender, EventArgs e)
        {
            if (txtNum.Text == "" || Convert.ToInt32(txtNum.Text)<=0)
            {
                MessageBox.Show("请输入用餐人数");
            }
            else
            {
                string RoomName = cbNum.SelectedItem.ToString();
                SqlCommand cmd1 = new SqlCommand("update tb_Room set GuestName='"
                    + txtName.Text + "', zhangdanDate='" + dateTimePicker1.Value.ToString()
                    + "', Num='" + Convert.ToInt32(txtNum.Text) + "', WaiterName='"
                    + cbWaiter.SelectedItem.ToString() + "', RoomZT='使用' where RoomName='"

```

```
        + name + "'", conn);  
        cmd1.ExecuteNonQuery();  
        this.Close();  
    }  
}  
private void btnExit_Click(object sender, EventArgs e)  
{  
    this.Close();  
}  
private void groupBox1_Enter(object sender, EventArgs e)  
{  
}  
}  
}
```

### 9.6.5 点菜模块

当顾客选好桌台之后，并且在开台之后，就可以点菜操作了。在点菜时，系统会显示一些店内的菜，用户可以在上面选择不同的菜。

本模块的实现文件是 DC.cs，其实现流程如下。

- (1) 定义公共变量 RName，用于接收指定桌台的名称。
- (2) 加载窗体时，将数据库中的所有菜系名称检索出来并显示在 TreeView 控件上，以供用户选择。
- (3) 当用户双击某个菜时，将在右侧显示此菜的详细信息。
- (4) 当用户更改商品数量时，商品价格也随之改变。
- (5) 自定义 GetData 方法用于显示所有的点菜信息。
- (6) 完成点菜操作后单击“保存”按钮，将用户所点的才进行保存。
- (7) 用户可以退掉某一个菜。

文件 DC.cs 的具体实现代码如下所示：

```
namespace MrCy  
{  
    public partial class DC : Form  
    {  
        public DC()  
        {  
            InitializeComponent();  
        }  
        public string RName;  
        private void frmDC_Load(object sender, EventArgs e)  
        {  
            this.Text = RName + "点/加菜";  
            TreeNode newnode1 = tvFood.Nodes.Add("锅底");  
            TreeNode newnode2 = tvFood.Nodes.Add("配菜");  
            TreeNode newnode3 = tvFood.Nodes.Add("烟酒");  
            TreeNode newnode4 = tvFood.Nodes.Add("主食");  
            SqlConnection conn = BaseClass.DBConn.CyCon();  
            conn.Open();  
            SqlCommand cmd =  
                new SqlCommand("select * from tb_food where foodty='1'", conn);  
            SqlDataReader sdr = cmd.ExecuteReader();  
            while (sdr.Read())  
            {  

```



```

        newnode1.Nodes.Add(sdr[3].ToString().Trim());
    }
    sdr.Close();
    cmd = new SqlCommand("select * from tb_food where foodty='2'", conn);
    sdr = cmd.ExecuteReader();
    while (sdr.Read())
    {
        newnode2.Nodes.Add(sdr[3].ToString().Trim());
    }
    sdr.Close();
    cmd = new SqlCommand("select * from tb_food where foodty='3'", conn);
    sdr = cmd.ExecuteReader();
    while (sdr.Read())
    {
        newnode3.Nodes.Add(sdr[3].ToString().Trim());
    }
    sdr.Close();
    cmd = new SqlCommand("select * from tb_food where foodty='4'", conn);
    sdr = cmd.ExecuteReader();
    while (sdr.Read())
    {
        newnode4.Nodes.Add(sdr[3].ToString().Trim());
    }
    sdr.Close();
    cmd = new SqlCommand("select * from tb_Waiter", conn);
    sdr = cmd.ExecuteReader();
    while (sdr.Read())
    {
        cbWaiter.Items.Add(sdr["WaiterName"].ToString().Trim());
    }
    cbWaiter.SelectedIndex = 0;
    sdr.Close();
    cmd = new SqlCommand(
        "select RoomZT from tb_Room where RoomName='" + RName + "'", conn);
    string zt = Convert.ToString(cmd.ExecuteScalar());
    if (zt.Trim() == "待用")
    {
        groupBox1.Enabled = false;
        groupBox2.Enabled = false;
        groupBox3.Enabled = false;
        groupBox4.Enabled = false;
    }
    conn.Close();
    GetData();
    tvFood.ExpandAll();
}

private void treeView1_DoubleClick(object sender, EventArgs e)
{
    string foodname = tvFood.SelectedNode.Text;
    if (foodname == "锅底" || foodname == "配菜"
        || foodname == "烟酒" || foodname == "主食")
    {
    }
    else
    {
        SqlConnection conn = BaseClass.DBConn.CyCon();
        conn.Open();
        SqlCommand cmd = new SqlCommand(
            "select * from tb_food where foodname='" + foodname + "'", conn);
        SqlDataReader sdr = cmd.ExecuteReader();
    }
}

```

```

        sdr.Read();
        txtNum.Text = sdr["foodnum"].ToString().Trim();
        txtName.Text = foodname;
        txtprice.Text = sdr["foodprice"].ToString().Trim();
        conn.Close();
        if (txtpnum.Text == "")
        {
            MessageBox.Show("数量不能为空");
            return;
        }
        else
        {
            txtallprice.Text = Convert.ToString(
                Convert.ToInt32(txtprice.Text) * Convert.ToInt32(txtpnum.Text));
        }
    }
}

private void txtpnum_TextChanged(object sender, EventArgs e)
{
    if (txtpnum.Text == "")
    {
        MessageBox.Show("数量不能为空");
        return;
    }
    else
    {
        if (Convert.ToInt32(txtpnum.Text) < 1)
        {
            MessageBox.Show("不能为小于1的数字");
            return;
        }
        else
        {
            txtallprice.Text = Convert.ToString(
                Convert.ToInt32(txtprice.Text) * Convert.ToInt32(txtpnum.Text));
        }
    }
}

private void GetData()
{
    SqlConnection conn = BaseClass.DBConn.CyCon();
    SqlDataAdapter sda = new SqlDataAdapter(
        "select foodname,foodsum,foodallprice,waitername,beizhu,zhuotai,
        datatime from tb_GuestFood where zhuotai='"
        + RName + "'order by ID desc", conn);
    DataSet ds = new DataSet();
    sda.Fill(ds);
    dgvFoods.DataSource = ds.Tables[0];
}

private void txtpnum_KeyPress(object sender, KeyPressEventArgs e)
{
    if ((e.KeyChar != 8 && !char.IsDigit(e.KeyChar)) && e.KeyChar != 13)
    {
        MessageBox.Show("请输入数字");
        e.Handled = true;
    }
}

private void btnDelete_Click(object sender, EventArgs e)
{
    if (dgvFoods.SelectedRows.Count > 0)
    {

```

```

string names = dgvFoods.SelectedCells[0].Value.ToString();
SqlConnection conn = BaseClass.DBConn.CyCon();
conn.Open();
SqlCommand cmd = new SqlCommand(
    "delete from tb_GuestFood where foodname='"
    + names + "' and zhuotai='" + RName + "'", conn);
cmd.ExecuteNonQuery();
conn.Close();
GetData();
}
}
private void btnSave_Click(object sender, EventArgs e)
{
    if (txtName.Text == "" || txtNum.Text == "" || txtprice.Text == "")
    {
        MessageBox.Show("请将选择菜系");
        return;
    }
    else
    {
        if (txtpnum.Text == "")
        {
            MessageBox.Show("数量不能为空");
            return;
        }
        else
        {
            if (Convert.ToInt32(txtpnum.Text) <= 0)
            {
                MessageBox.Show("请输入消费数量");
                return;
            }
            else
            {
                SqlConnection conn = BaseClass.DBConn.CyCon();
                conn.Open();
                SqlCommand cmd = new SqlCommand(
                    "insert into tb_GuestFood(foodnum, foodname, foodsum,
                    foodallprice, waitername, beizhu, zhuotai, datetime) values('"
                    + txtNum.Text.Trim() + "', '" + txtName.Text.Trim() + "', '"
                    + txtpnum.Text.Trim() + "', '"
                    + Convert.ToDecimal(txtallprice.Text.Trim()) + "', '"
                    + cbWaiter.SelectedItem.ToString() + "', '" + txtbz.Text.Trim()
                    + "', '" + RName + "', '" + DateTime.Now.ToString() + "')", conn);
                cmd.ExecuteNonQuery();
                conn.Close();
                GetData();
            }
        }
    }
}
private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
private void tvFood_AfterSelect(object sender, TreeViewEventArgs e)
{
}
}

```



数据库是实现动态软件项目的中间媒介，但是在使用前，需要建立连接。对于 C# 来说，与不同数据库的连接方式不同。具体如下：

#### C#连接连接 Access:

```
using System.Data;
using System.Data.OleDb;
...
string strConnection = "Provider=Microsoft.Jet.OleDb.4.0;";
strConnection += @"Data Source=C:\BegASPNET\Northwind.mdb";
OleDbConnection objConnection = new OleDbConnection(strConnection);
...
objConnection.Open();
objConnection.Close();
```

#### C#连接 SQL Server:

```
using System.Data;
using System.Data.SqlClient;
...
string strConnection = "user id=sa;password=;";
strConnection += "initial catalog=Northwind;Server=YourSQLServer;";
strConnection += "Connect Timeout=30";
SqlConnection objConnection = new SqlConnection(strConnection);
...
objConnection.Open();
objConnection.Close();
```

#### C#连接 Oracle:

```
using System.Data.OracleClient;
using System.Data;
//在窗体上添加一个按钮，叫 Button1，双击 Button1，输入以下代码
private
void Button1_Click(object sender, System.EventArgs e)
{
    string ConnectionString="Data Source=sky;user=system; password=manager;";//写连接串
    OracleConnection conn = new OracleConnection(ConnectionString); //创建一个新连接
    try
    {
        conn.Open();
        OracleCommand cmd = conn.CreateCommand();

        cmd.CommandText = "select * from MyTable"; //在这儿写 SQL 语句
        OracleDataReader odr = cmd.ExecuteReader(); //创建一个 OracleDateReader 对象
        while(odr.Read()) //读取数据，如果 odr.Read() 返回为 false 的话，就说明到记录集的尾部了
        {
            //输出字段 1，这个数是字段索引
            Response.Write(odr.GetOracleString(1).ToString());
        }
        odr.Close();
    }
    catch(Exception ee)
    {
        Response.Write(ee.Message); //如果有错误，输出错误信息
    }
    finally
    {
        conn.Close(); //关闭连接
    }
}
```

## C#连接 MySQL:

```
using MySQLDriverCS;
// 建立数据库连接
MySQLConnection DBConn;
DBConn =
new MySQLConnection(new MySQLConnectionString(
    "localhost", "mysql", "root", "", 3306).AsString);
DBConn.Open();
// 执行查询语句
MySQLCommand DBComm;
DBComm = new MySQLCommand("select Host,User from user", DBConn);
// 读取数据
MySQLDataReader DBReader = DBComm.ExecuteReaderEx();
// 显示数据
try
{
    while (DBReader.Read())
    {
        Console.WriteLine(
            "Host = {0} and User = {1}", DBReader.GetString(0), DBReader.GetString(1));
    }
}
finally
{
    DBReader.Close();
    DBConn.Close();
}
//关闭数据库连接
DBConn.Close();
```

## C#连接 IBM DB2:

```
OleDbConnection1.Open();
//打开数据库连接
OleDbDataAdapter1.Fill(dataSet1, "Address");
//将得来的数据填入 dataSet
DataGrid1.DataBind();
//绑定数据
OleDbConnection1.Close();
//关闭连接
//增加数据库数据
```

在 Web Form 上新增对应字段数量个数的 TextBox, 及一个 button, 为该按钮增加 Click 响应事件代码如下:

```
this.OleDbInsertCommand1.CommandText =
    "INSERTintosADDRESS(NAME, EMAIL, AGE, ADDRESS) VALUES ('"
    + TextBox1.Text + "','" + TextBox2.Text + "','"
    + TextBox3.Text + "','" + TextBox4.Text + "')";
OleDbInsertCommand1.Connection.Open();
//打开连接
OleDbInsertCommand1.ExecuteNonQuery();
//执行该 SQL 语句
OleDbInsertCommand1.Connection.Close();
//关闭连接
```

## C#连接 SyBase:

```
Provider = Sybase.ASEOLEDBProvider.2;
```

```
Initial Catalog = 数据库名;
User ID = 用户名;
Data Source = 数据源;
Extended Properties = "";
Server Name = ip 地址;
Network Protocol = Winsock;
Server Port Address = 5000;
```

## 9.6.6 结账模块

当顾客消费之后，要对顾客的消费信息进行统计，算出他们的消费总额。此功能是通过结账模块实现的。本模块的实现文件是 JZ.cs，其实现流程如下。

- (1) 定义变量 Rname、price 和 bjf，分别用于接收主窗体模块中传递的桌台名和根据名称查询消费的总额。
- (2) 当加载窗体后，先显示桌台名，然后通过桌台名检索出消费的所有账目并显示在 DataGridView，最后将查询出的消费金额显示在 Label 控件上。
- (3) 判断输入金额是否正确数据。
- (4) 在收银文本框中输入金额后，系统将自动计算出需要退还给用户的金额数。
- (5) 当消费者支付后，单击“结账”按钮后，完成整个结账操作，并将当前顾客所使用的桌台状态设置为“可用”。

文件 JZ.cs 的具体实现代码如下所示：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace MrCy
{
    public partial class JZ : Form
    {
        public JZ()
        {
            InitializeComponent();
        }
        public string Rname;
        public string price;
        public string bjf;
        private void frmJZ_Load(object sender, EventArgs e)
        {
            this.Text = Rname + "结账";
            groupBox1.Text = "当前桌台-" + Rname;
            SqlConnection conn = BaseClass.DBConn.CyCon();
            SqlDataAdapter sda = new SqlDataAdapter(
                "select foodname,foodsum,foodallprice,waitername,beizhu,zhuotai,
                datetime from tb_GuestFood where zhuotai='"
                + Rname + "'order by ID desc", conn);
            DataSet ds = new DataSet();
            sda.Fill(ds);
            dgvRecord.DataSource = ds.Tables[0];
            conn.Open();
        }
    }
}
```



```

SqlCommand cmd = new SqlCommand(
    "select sum(foodallprice) from tb_GuestFood where zhuotai='"
    + Rname + "'", conn);
price = Convert.ToString(cmd.ExecuteScalar());
if (price == "")
{
    lblprice.Text = "0";
    btnJZ.Enabled = false;
}
else
{
    cmd = new SqlCommand("select RoomBJF from tb_Room where RoomName='"
        + Rname + "'", conn);
    bjf = cmd.ExecuteScalar().ToString();
    if (bjf == "0")
    {
        btnJZ.Enabled = true;
        lblprice.Text = price + "*95%" + "+" + bjf + "="
            + (Convert.ToDecimal(Convert.ToDouble(price)
                * Convert.ToDouble(0.95))).ToString("C");
    }
    else
    {
        btnJZ.Enabled = true;
        lblprice.Text = price + "*95%" + "+" + bjf + "="
            + (Convert.ToDecimal(Convert.ToDouble(price)
                * Convert.ToDouble(0.95))
            + Convert.ToDecimal(bjf)).ToString("C");
    }
    conn.Close();
}
}

private void txtmoney_KeyPress(object sender, KeyPressEventArgs e)
{
    if ((e.KeyChar!=8 && !char.IsDigit(e.KeyChar)) && e.KeyChar!=13)
    {
        MessageBox.Show("请输入数字");
        e.Handled = true;
    }
}

private void txtmoney_TextChanged(object sender, EventArgs e)
{
    if (price == "")
    {
        lbl0.Text = "0";
    }
    else
    {
        if (txtmoney.Text == "")
        {
            txtmoney.Text = "0";
            lbl0.Text = "0";
        }
        else
        {
            lbl0.Text = Convert.ToDecimal(
                Convert.ToDouble(txtmoney.Text.Trim())
                - Convert.ToDouble(price) * Convert.ToDouble(0.95)
                - Convert.ToDouble(bjf)).ToString("C");
        }
    }
}

```

```

    }
}
private void btnJZ_Click(object sender, EventArgs e)
{
    if (txtmoney.Text==" " || lbl0.Text=="0")
    {
        MessageBox.Show("请先结账");
        return;
    }
    else
    {
        if (lbl0.Text.Substring(1, 1) == "-")
        {
            MessageBox.Show("金额不足");
            return;
        }
        else
        {
            SqlConnection conn = BaseClass.DBConn.CyCon();
            conn.Open();
            SqlCommand cmd =
                new SqlCommand("delete from tb_GuestFood where zhuotai='"
                    + Rname + "'", conn);
            cmd.ExecuteNonQuery();
            cmd = new SqlCommand(
                "update tb_Room set RoomZT='待用',Num=0,WaiterName='' where RoomName='"
                + Rname + "'", conn);
            cmd.ExecuteNonQuery();
            conn.Close();
            this.Close();
        }
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
}
}

```

### 9.6.7 员工管理模块

此模块的功能是管理系统内员工的信息，包括修改、查询和删除等操作。本模块的实现文件是 User.cs，其实现流程如下。

- (1) BindData()绑定显示员工的具体信息。
- (2) button1\_Click 实现“充填”按钮的处理事件。
- (3) button2\_Click 实现“修改”按钮的处理事件。
- (4) button3\_Click 实现“保存”按钮的处理事件。
- (5) button4\_Click 实现“取消”按钮的处理事件。
- (6) button6\_Click 实现“删除”按钮的处理事件。

文件 User.cs 的具体实现代码如下所示：

```
namespace MrCy
```

```

public partial class User : Form
{
    public User()
    {
        InitializeComponent();
    }
    private void BindData()
    {
        SqlConnection conn = BaseClass.DBConn.CyCon();
        SqlDataAdapter sda = new SqlDataAdapter(
            "select WaiterName,CardNum,WaiterNum,Sex,Age,Tel,
            ID from tb_Waiter order by ID desc", conn);
        DataSet ds = new DataSet();
        sda.Fill(ds);
        dataGridView1.DataSource = ds.Tables[0];
    }
    private void button7_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void frmUser_Load(object sender, EventArgs e)
    {
        comboBox1.SelectedIndex = 0;
    }

    private void button5_Click(object sender, EventArgs e)
    {
        BindData();
    }

    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        txtname.Text = dataGridView1.SelectedCells[0].Value.ToString();
        txtjc.Text = dataGridView1.SelectedCells[1].Value.ToString();
        txtbjf.Text = dataGridView1.SelectedCells[2].Value.ToString();
        comboBox1.SelectedItem =
            dataGridView1.SelectedCells[3].Value.ToString().Trim();
        txtlx.Text = dataGridView1.SelectedCells[4].Value.ToString();
        txtbz.Text = dataGridView1.SelectedCells[5].Value.ToString();
        button2.Enabled = true;
        button6.Enabled = true;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        txtname.Text = "";
        txtlx.Text = "";
        txtjc.Text = "";
        txtbz.Text = "";
        txtbjf.Text = "";
        txtname.Enabled = true;
        txtjc.Enabled = true;
        txtbjf.Enabled = true;
        comboBox1.Enabled = true;
        txtlx.Enabled = true;
        txtbz.Enabled = true;
        button3.Enabled = true;
        button4.Enabled = true;
        button2.Enabled = false;
    }
}

```



```

private void button2_Click(object sender, EventArgs e)
{
    button1.Enabled = false;
    button3.Enabled = true;
    button4.Enabled = true;
    txtname.Enabled = false;
    txtjc.Enabled = true;
    txtbjf.Enabled = true;
    this.comboBox1.Enabled = true;
    txtlx.Enabled = true;
    txtbz.Enabled = true;
}

private void button3_Click(object sender, EventArgs e)
{
    SqlConnection conn = BaseClass.DBConn.CyCon();
    conn.Open();
    SqlCommand cmd = new SqlCommand(
        "select count(*) from tb_Waiter where WaiterName='"
        + txtname.Text + "'", conn);
    int i = Convert.ToInt32(cmd.ExecuteScalar());
    if (i > 0)
    {
        cmd = new SqlCommand("update tb_Waiter set WaiterName='"
            + txtname.Text + "',CardNum='" + txtjc.Text + "',WaiterNum='"
            + txtbjf.Text + "',Sex='" + comboBox1.SelectedItem.ToString()
            + "',Age='" + txtlx.Text + "',Tel='" + txtbz.Text + "' where ID='"
            + dataGridView1.SelectedCells[6].Value.ToString() + "'", conn);
        cmd.ExecuteNonQuery();
        conn.Close();
        BindData();
        button1.Enabled = true;
        button2.Enabled = false;
        button3.Enabled = false;
        button4.Enabled = false;
        button5.Enabled = true;
        button6.Enabled = false;
        button9.Enabled = true;
        txtname.Enabled = false;
    }
    else
    {
        cmd = new SqlCommand(
            "insert into tb_Waiter(WaiterName,CardNum,WaiterNum, Sex, Age, Tel)
            values('" + txtname.Text + "', '" + txtjc.Text + "', '" + txtbjf.Text
            + "', '" + comboBox1.SelectedItem.ToString() + "', '" + txtlx.Text
            + "', '" + txtbz.Text + "')", conn);
        cmd.ExecuteNonQuery();
        conn.Close();
        BindData();
        button1.Enabled = true;
        button2.Enabled = false;
        button3.Enabled = false;
        button4.Enabled = false;
        button5.Enabled = true;
        button6.Enabled = false;
        button9.Enabled = true;
        txtname.Enabled = false;
    }
}

private void button4_Click(object sender, EventArgs e)
{

```

```

        button1.Enabled = true;
        button2.Enabled = false;
        button3.Enabled = false;
        button4.Enabled = false;
        button6.Enabled = false;
        txtname.Enabled = false;
        txtjc.Enabled = false;
        txtbjf.Enabled = false;
        this.comboBox1.Enabled = false;
        txtlx.Enabled = false;
        txtbz.Enabled = false;
    }
    private void button6_Click(object sender, EventArgs e)
    {
        SqlConnection conn = BaseClass.DBConn.CyCon();
        conn.Open();
        SqlCommand cmd = new SqlCommand("delete from tb_Waiter where ID='"
            + dataGridview1.SelectedCells[6].Value.ToString() + "'", conn);
        cmd.ExecuteNonQuery();
        conn.Close();
        BindData();
    }
}

```

### 9.6.8 修改密码模块

此模块的功能是修改系统用户的密码信息，本模块的实现文件是 Pwd.cs，其实现流程如下。

- (1) 通过 button1\_Click 设置输入的密码不能为空，并确保两次输入的密码一致。
- (2) 如果输入的密码合法，则通过 SqlCommand 对象 cmd 实现密码修改。

文件 Pwd.cs 的具体实现代码如下所示：

```

public partial class Pwd : Form
{
    public Pwd()
    {
        InitializeComponent();
    }
    public string names;
    private void button2_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        if (txtPwd1.Text == "")
        {
            MessageBox.Show("请输入密码");
            txtPwd1.Focus();
        }
        else
        {
            if (txtPwd2.Text != txtPwd1.Text)
            {
                MessageBox.Show("两次密码不一致");
                txtPwd2.Focus();
            }
        }
    }
}

```

```

    }
    else
    {
        SqlConnection conn = BaseClass.DBConn.CyCon();
        conn.Open();
        SqlCommand cmd = new SqlCommand("update tb_User set UserPwd='"
            + txtPwd1.Text + "' where UserName='" + names + "'", conn);
        cmd.ExecuteNonQuery();
        if (MessageBox.Show("密码修改成功", "提示", MessageBoxButtons.OK,
            MessageBoxIcon.Asterisk) == DialogResult.OK)
        {
            this.Close();
        }
    }
}

private void frmPwd_Load(object sender, EventArgs e)
{
}
}

```

作为一个窗体项目，可能涉及到多个窗体的操作问题。此时，窗体间的数据传递就变得很重要了。假设我们需要单击主窗体 FMMain 中的某一个按钮时会打开子窗体 FMChild 并将某一个值传给子窗体 FMChild。

在一般情况下，单击按钮显示子窗体 FMChild 的代码为：

```

FMChild fmChild = new FMChild();
fmChild.ShowDialog();
fmChild.Dispose();

```

如果需要将主窗体 FMMain 中的 string strValueA 的值传给 FMChild，那么首先对 strValueA 进行如下处理，使其成为主窗体 FMMain 的一个属性：

```

private string strValueA;
public string StrValueA {
    get {
        return strValueA;
    }
    set {
        strValueA = value;
    }
}

```

接着，修改显示子窗体的代码为以下两种中的一种。

方法一：

```

FMChild fmChild = new FMChild();
fmChild.ShowDialog(this);
fmChild.Dispose();

```

方法二：

```

FMChild fmChild = new FMChild();
FMChild.Owner = this;
fmChild.ShowDialog();
fmChild.Dispose();

```

然后在修改子窗体 FMChild 中声明一个主窗体 FMMain 对象：



```
FMMain fmMain;
```

在需要使用主窗体 FMMain 的 string strValueA 的地方加上如下代码:

```
fmMain = (FMMain)this.Owner;
```

这样, 就可以获得主窗体 FMMain 中的 strValueA 的值了。这时, 如果需要将子窗体 FMChild 中的 string strValueB 传给主窗体 FMMain, 同样处理 string strValueB:

```
private string strValueB;
public string StrValueB{
    get { return strValueB; }
    set { strValueB = value; }
}
```

那么, 在关闭子窗体代码 fmChild.Dispose(); 后, 可以写一些代码来保存或者处理 FMChild 的 strValueB, 例如:

```
string strTmp = fmChild.StrValueB;
```

在此处需要注意的是, Form1 中的 label1 要设为 public。另外, 也可以用 C# 实现在两个 WinForm 间传值, 具体实现的思路如下所示。

- 从 Form1 传递到 Form2: 两个窗体即两个类, 两个窗体间的数据传送可以采用构造函数来实现。
- 从 Form2 返回到 Form1, 并传递数据: 实例化 Form2 后, 打开 f2 用 ShowDialog() 方法, 然后等 f2 关闭时, 再回传数据到 Form1。

具体的实现步骤如下。

(1) 新建两个窗口: Form1、Form2。

(2) 打开 Form2, 添加一个 textBox(textBox1); 添加一个 Button(button1); 然后添加一个构造函数:

```
//定义一个变量, 用来传值
public string returnValue;
public Form2(string txtValue)
{
    InitializeComponent();
    this.textBox1.Text = txtValue;
}
```

然后在 button1 的单击事件中添加如下代码:

```
private void button1_Click(object sender, EventArgs e)
{
    returnValue = this.textBox1.Text;
    this.Close();
}
```

(3) 在 Form1 中添加一个 textBox(textBox1); 添加一个 Button(button1); 然后在 button1 的单击事件中添加如下代码:

```
private void button1_Click(object sender, EventArgs e)
{
    string txtValue = this.textBox1.Text;
    Form2 f2 = new Form2(txtValue);
    f2.ShowDialog();
    this.textBox1.Text = f2.returnValue;
}
```

}

Form1 中(父窗口)的代码如下:

```
public class Form1 : System.Windows.Forms.Form
{
    private System.Windows.Forms.Button btnOpen;
    public System.Windows.Forms.TextBox txtContent; //注意是public
    ...
    [STAThread]
    static void Main()
    {
        Application.Run(new Form1());
    }
    private void btnOpen_Click(object sender, System.EventArgs e)
    {
        Form2 frm = new Form2(this);
        frm.ShowDialog();
    }
}
```

Form2 中(子窗口)的代码如下:

```
public class Form2 : System.Windows.Forms.Form
{
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.TextBox txtValue;
    private Form _parentForm = null;
    public Form2()
    {
        InitializeComponent();
    }
    public Form2(Form parentForm)
    {
        InitializeComponent();
        this._parentForm = parentForm;
    }
    ...
    //更新父窗口中文本框中的值
    private void button1_Click(object sender, System.EventArgs e)
    {
        ((Form1)_parentForm).txtContent.Text = this.txtValue.Text;
    }
}
```

### 9.6.9 桌台信息模块

此模块的功能是显示某个桌台的具体信息,本模块的实现文件是 Desk.cs,其实现流程如下。

- (1) 通过 button1\_Click 实现“充填”按钮的处理。
- (2) 通过 button2\_Click 实现“修改”按钮的处理。
- (3) 通过 button3\_Click 实现“保存”按钮的处理。
- (4) 通过 button4\_Click 实现“取消”按钮的处理。
- (5) 通过 button6\_Click 实现“删除”按钮的处理。

文件 Desk.cs 的具体实现代码如下所示:

```

public partial class Desk : Form
{
    public Desk()
    {
        InitializeComponent();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        BindData();
    }
    private void BindData()
    {
        SqlConnection conn = BaseClass.DBConn.CyCon();
        SqlDataAdapter sda = new SqlDataAdapter("select RoomName,RoomJC,RoomBJF,
            RoomWZ,RoomType,RoomBZ,ID from tb_Room order by ID desc", conn);
        DataSet ds = new DataSet();
        sda.Fill(ds);
        dataGridView1.DataSource = ds.Tables[0];
    }
    private void frmDesk_Load(object sender, EventArgs e)
    {
    }

    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        txtname.Text = dataGridView1.SelectedCells[0].Value.ToString();
        txtjc.Text = dataGridView1.SelectedCells[1].Value.ToString();
        txtbjf.Text = dataGridView1.SelectedCells[2].Value.ToString();
        txtwz.Text = dataGridView1.SelectedCells[3].Value.ToString();
        txtlx.Text = dataGridView1.SelectedCells[4].Value.ToString();
        txtbz.Text = dataGridView1.SelectedCells[5].Value.ToString();
        button2.Enabled = true;
        button6.Enabled = true;
    }

    private void button7_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        SqlConnection conn = BaseClass.DBConn.CyCon();
        conn.Open();
        SqlCommand cmd = new SqlCommand("delete from tb_Room where RoomName='"
            + dataGridView1.SelectedCells[0].Value.ToString() + "'", conn);
        cmd.ExecuteNonQuery();
        conn.Close();
        BindData();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        button1.Enabled = false;
        button3.Enabled = true;
        button4.Enabled = true;
        txtjc.Enabled = true;
        txtbjf.Enabled = true;
        txtwz.Enabled = true;
        txtlx.Enabled = true;
    }
}

```



```

        txtbz.Enabled = true;
    }

    private void button4_Click(object sender, EventArgs e)
    {
        button1.Enabled = true;
        button2.Enabled = false;
        button3.Enabled = false;
        button4.Enabled = false;
        button6.Enabled = false;
        txtname.Enabled = false;
        txtjc.Enabled = false;
        txtbjf.Enabled = false;
        txtwz.Enabled = false;
        txtlx.Enabled = false;
        txtbz.Enabled = false;
    }

    private void button3_Click(object sender, EventArgs e)
    {
        SqlConnection conn = BaseClass.DBConn.CyCon();
        conn.Open();
        SqlCommand cmd = new SqlCommand("select count(*) from tb_Room where RoomName='"
            + txtname.Text + "'", conn);
        int i = Convert.ToInt32(cmd.ExecuteScalar());
        if (i > 0)
        {
            cmd = new SqlCommand("update tb_Room set RoomName='" + txtname.Text
                + "',RoomJC='" + txtjc.Text + "',RoomBJF='" + txtbjf.Text + "',RoomWZ='"
                + txtwz.Text + "',RoomType='" + txtlx.Text + "',RoomBZ='" + txtbz.Text
                + "' where ID='" + dataGridView1.SelectedCells[6].Value.ToString()
                + "'", conn);
            cmd.ExecuteNonQuery();
            conn.Close();
            BindData();
            button1.Enabled = true;
            button2.Enabled = false;
            button3.Enabled = false;
            button4.Enabled = false;
            button5.Enabled = true;
            button6.Enabled = false;
            button9.Enabled = true;
            txtname.Enabled = false;
        }
        else
        {
            cmd = new SqlCommand("insert into tb_Room(RoomName,RoomJC,RoomBJF,
                RoomWZ,RoomType,RoomBZ) values('" + txtname.Text + "','" + txtjc.Text + "','"
                + txtbjf.Text + "','" + txtwz.Text + "','" + txtlx.Text + "','"
                + txtbz.Text + "')", conn);
            cmd.ExecuteNonQuery();
            conn.Close();
            BindData();
            button1.Enabled = true;
            button2.Enabled = false;
            button3.Enabled = false;
            button4.Enabled = false;
            button5.Enabled = true;
            button6.Enabled = false;
            button9.Enabled = true;
            txtname.Enabled = false;
        }
    }

```

```

    }
}

private void button1_Click(object sender, EventArgs e)
{
    txtname.Text = "";
    txtlx.Text = "";
    txtjc.Text = "";
    txtbz.Text = "";
    txtbjf.Text = "";
    txtname.Enabled = true;
    txtjc.Enabled = true;
    txtbjf.Enabled = true;
    txtwz.Enabled = true;
    txtlx.Enabled = true;
    txtbz.Enabled = true;
    button3.Enabled = true;
    button4.Enabled = true;
    button2.Enabled = false;
}
}

```

到此为止，整个项目全部介绍完毕。当将项目开发完毕后，开发公司需要对项目进行封装并发布处理。代码封装是一个陈旧的话题，在 C# 项目中，为了系统程序的安全，有时需要对重要的代码进行封装处理。下面以验证码文件为例，向读者介绍将类文件转换为程序集的方法。验证码文件转换为程序集的流程如下。

(1) 在 Visual Studio 中新建项目，选择模板为“类库”，命名为“ValidateCode”，如图 9-16 所示。



图 9-16 新建类库

(2) 修改文件 Class1.cs 为 ValidateCode.cs，然后将文件 Yanzhengma.cs 的代码复制进来，如图 9-17 所示。

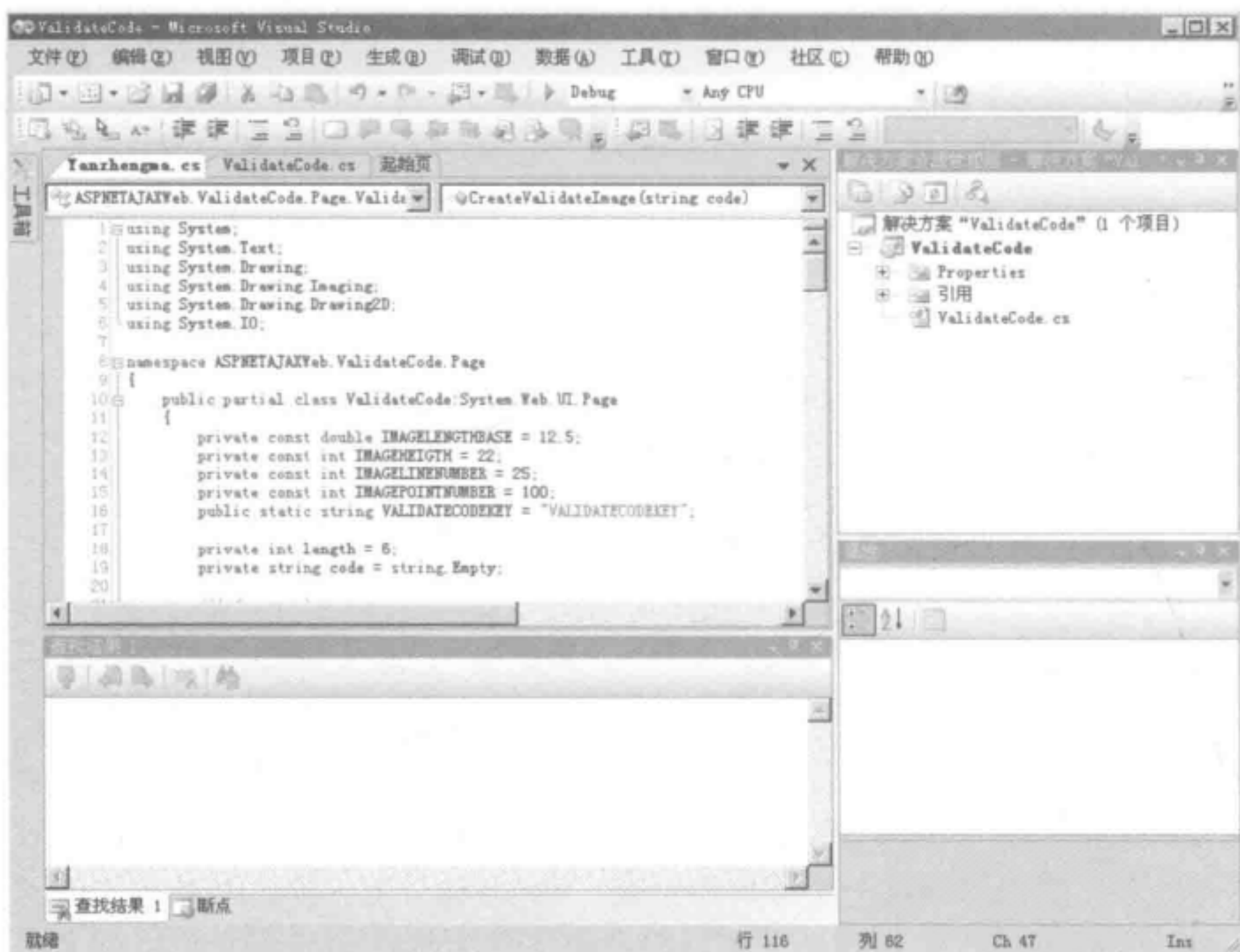


图 9-17 设置代码

- (3) 右键单击“解决方案管理器”中的 ValidateCode 项目，然后选择“属性”命令。
- (4) 在弹出对话框中，设置程序集名为“ASPNETAJAXWeb.ValidateCode”，默认命名空间为 ASPNETAJAXWeb.ValidateCode.Page，如图 9-18 所示。

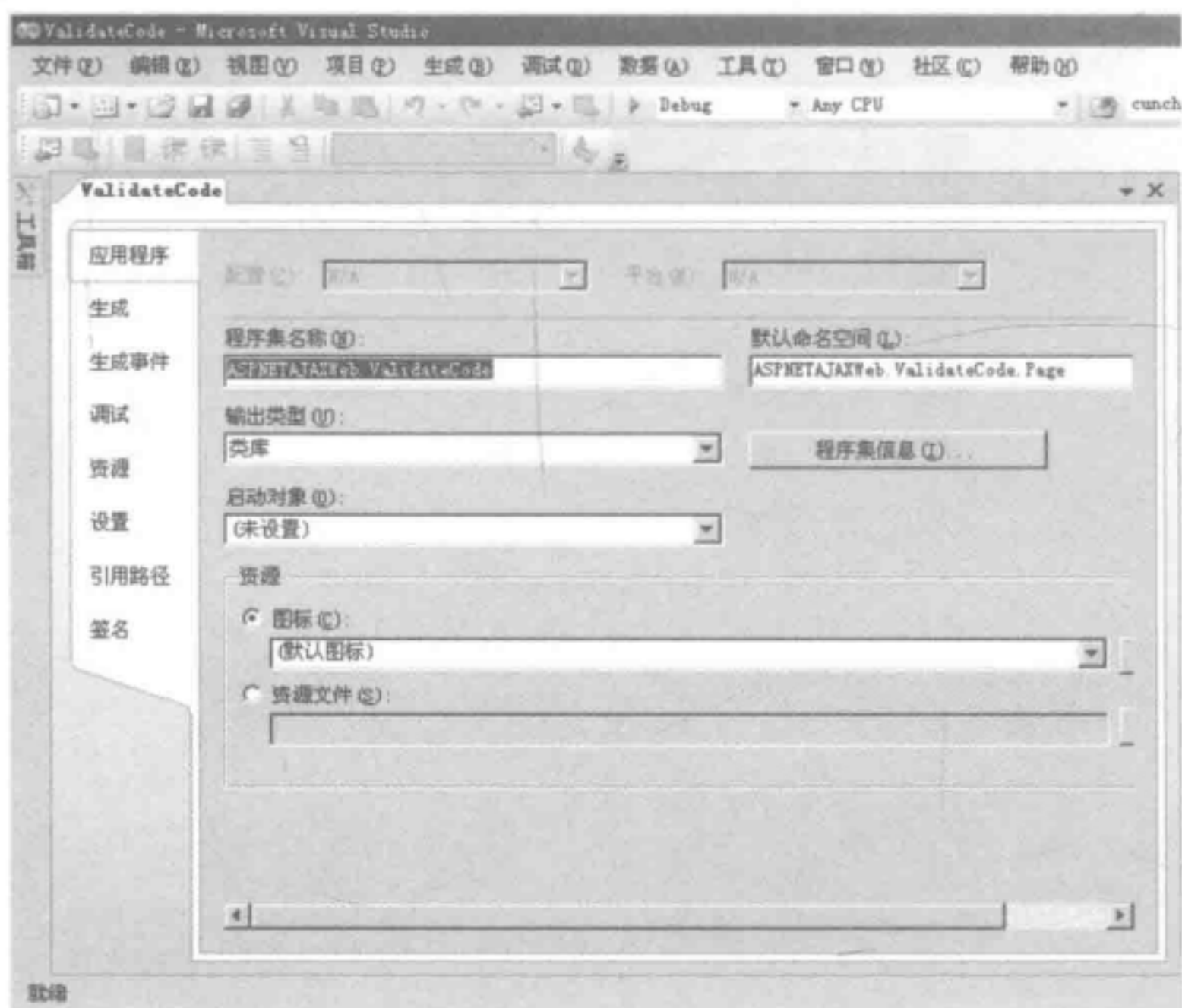


图 9-18 设置类库



经过上述步骤操作处理后，将在“ValidateCode\bin\Debug”文件夹内自动生成一个验证码程序集文件 ValidateCode.dll。读者可以将其拷贝在自己项目的 Bin 文件夹内，然后进行引用。具体操作的流程如下所示。

- (1) 将 ValidateCode.dll 拷贝在自己项目的 Bin 文件夹内。
- (2) 将需要调用 ValidateCode.dll 的文件放在项目的根目录下，即与 Bin 文件夹同级的目录。
- (3) 右键单击“解决方案管理器”中的 Bin 节点，然后从弹出的快捷菜单中选择“添加引用”命令，如图 9-19 所示。
- (4) 在弹出的“添加引用”对话框中单击“浏览”，然后，找到 Bin 文件夹内的 ValidateCode.dll，将其引用到项目中，如图 9-20 所示。



图 9-19 添加引用

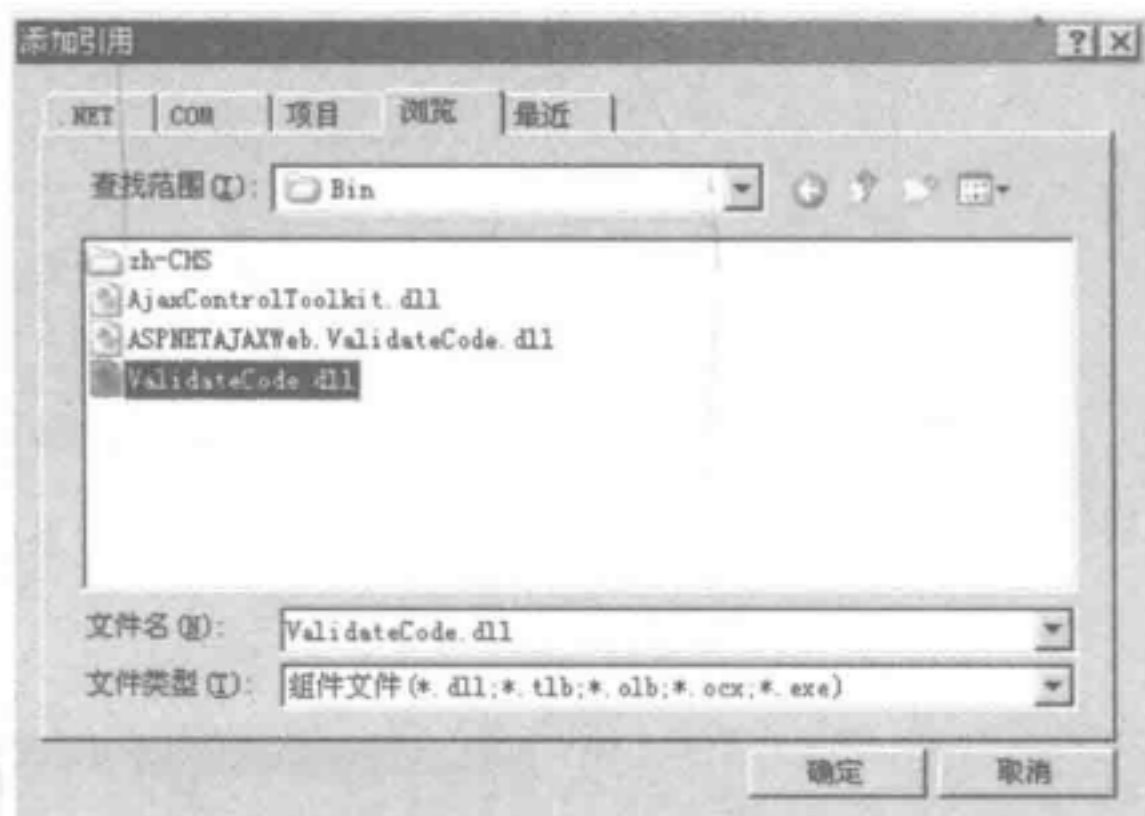


图 9-20 引用文件

## 9.7 项目调试

视频讲解 光盘：视频\第9章\项目调试.avi

将本项目命名为“CanYin”，用户登录界面如图 9-21 所示。



图 9-21 用户登录界面的效果

系统主界面的效果如图 9-22 所示。



图 9-22 系统的主界面

开台单界面的效果如图 9-23 所示。

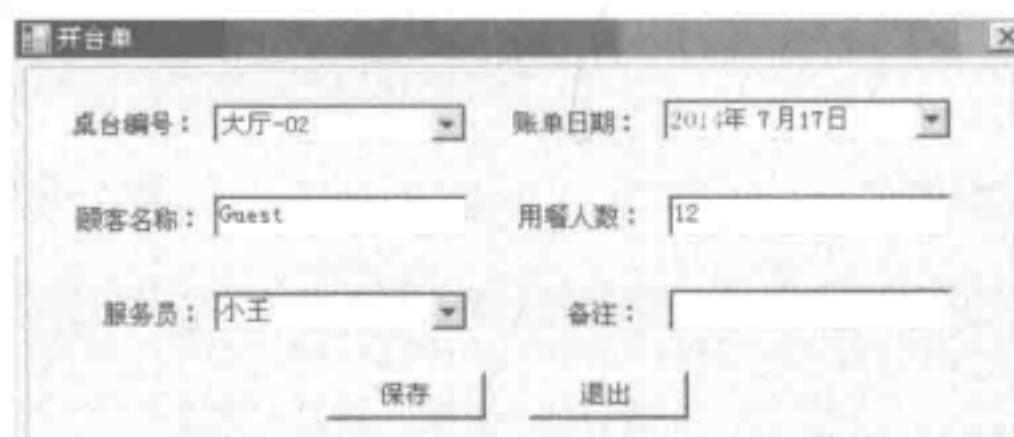


图 9-23 开台单界面的效果

点菜、加菜界面的效果如图 9-24 所示。

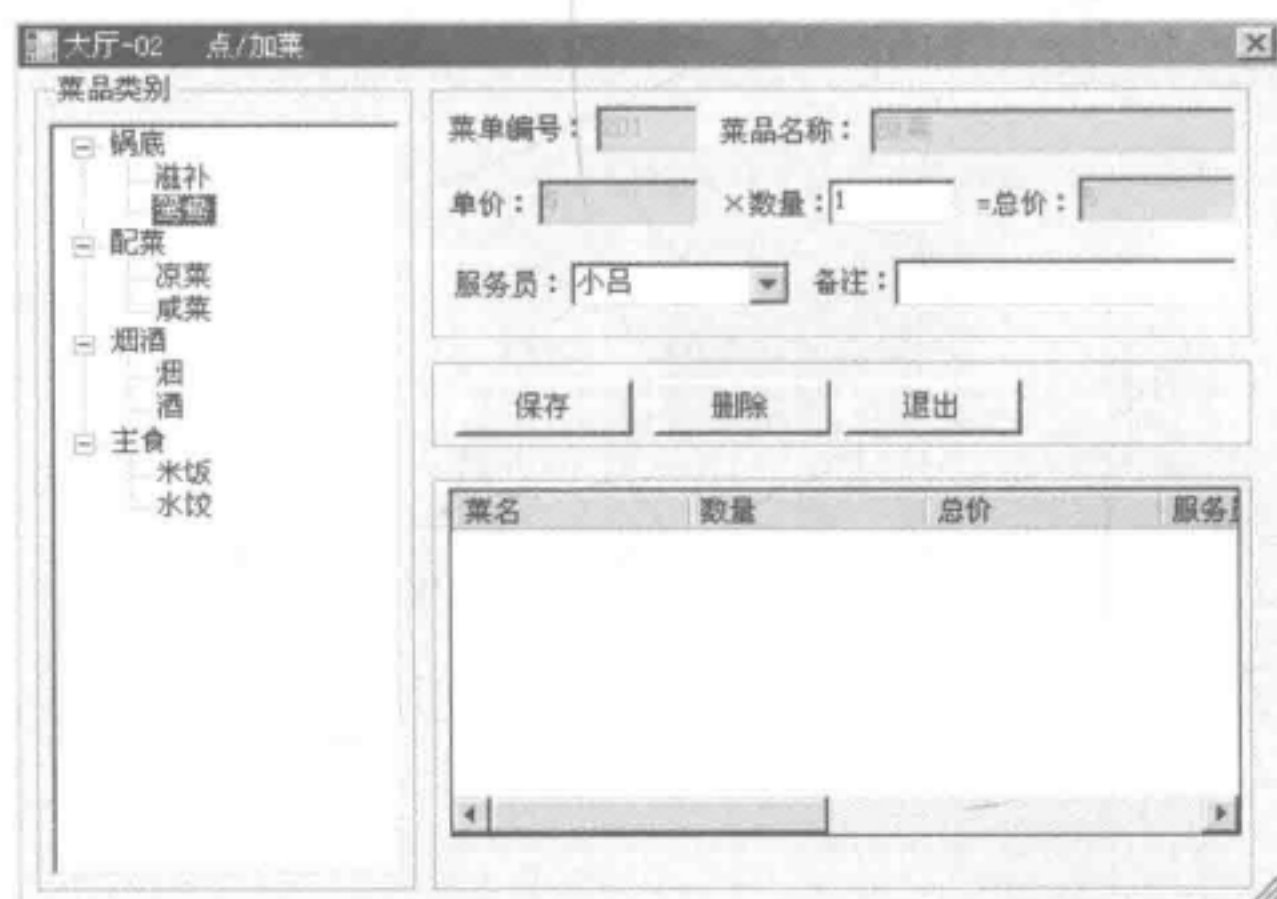


图 9-24 点菜、加菜界面的效果

职员界面的效果如图 9-25 所示。

职员信息

重填

修改

保存

取消

查询

删除

退出

员工姓名：

身份证号：

员工编号：

员工性别：

员工年龄：

联系电话：

| 员工姓名 | 身份证号码 | 员工编号 | 性别 | 年龄 | 联系电话   |
|------|-------|------|----|----|--------|
| 小张   | m006  | 006  | 男  | 26 | 130*** |
| 小房   | m005  | 005  | 男  | 27 | 130*** |
| 小贵   | m004  | 004  | 女  | 23 | 130*** |
| 小王   | m003  | 003  | 男  | 23 | 130*** |
| 小郭   | m002  | 002  | 女  | 23 | 130*** |
| 小吕   | m001  | 001  | 男  | 27 | 130*** |

图 9-25 职员界面的效果





## 第 10 章 短信群发系统

赢项目开发

企业短信群发系统是企业节约开支、提高效益而服务的，它将“打折信息”、“促销活动”、“新品发布”等相关信息发布到目标客户的手机上，为企业占有市场创造了无限商机，也能为企业大幅降低广告开支。

企业短信群发系统的另一层含义是指手机终端拥有的短信群发功能，用户编辑一条短信或转发相关的短信时，可以编辑 10 条甚至更多的手机号码进行群发。

在本章的内容中，将通过一个具体的实例，来讲解使用 C# 开发短信群发系统的具体实现流程。



### 赠送的超值电子书

- 091 C#集合概述
- 092 使用集合
- 093 数组的方法和属性
- 094 CreateInstance 方法
- 095 ArrayList 集合类
- 096 Hashtable 集合类
- 097 Queue 集合类
- 098 SortedList 集合类
- 099 Stack 集合类
- 100 foreach 概述

## 10.1 做好项目管理者

视频讲解 光盘：视频\第10章\做好项目管理者.avi

无论程序员个体之间差异有多大，都会不断地在实战中实现技术上的反思和提高。当开发技术和经验积累到一定程度后，必然会经历带队做项目的阶段。身处这一阶段的人，开发技术上的差异已经不是那么重要了，最关键的是管理能力和协调能力。在本节的内容中，将与读者一起，探讨做好项目管理者秘诀。

### 10.1.1 软件工程师到项目管理者之路

很多程序员可能会干一辈子的软件工程师，但是，也有一些程序员工作机遇比较好，不但负责了项目的设计工作，而且负责了项目管理工作，这表明已经对项目经理这个工作有了一定的尝试。因此，这类程序员可以审视一下自己是否适合这项工作。如果自己对这种工作比较满意，而且具备项目经理的能力要求，则可以向项目经理这个方向发展。

项目经理承担着项目管理的职责，对项目负主要责任。项目经理和程序员的作用也不相同，项目经理的重点已经从编程转移到对人、对技术、对进度、对项目的管理。由于软件的项目经理与软件项目的相关性非常大，因此，必须了解软件开发的各个环节、了解开发的各种技术和应用，了解开发队伍人员的水平和特点。所以，项目经理依旧与程序员脱不了干系。从程序员成长到项目经理，可以更好地理解程序员在项目中的地位和作用，了解软件开发的各种规律性的东西，从而保证项目的正常完成。而且，项目经理的收入在公司中也是比较高的。因此，做项目经理是程序员的另一个发展空间。在此需要特别提醒一下，程序员在担任项目经理之前，最好要把软件设计工作做好，这样，在做项目经理的时候，就会有很好的基础。

当然，也有很多程序员希望自己能成为公司中的主管、经理、老总、老板；这样，无论是收入和在公司中的地位，都相对比较高一些。另外，一方面，程序员可能厌倦了年复一年的编程工作，对工作产生了抵触情绪，希望能摆脱这种步步紧逼的工作状态。走向管理其实要求程序员要比一般人做更多的准备，做更多的转型工作，并不是想当管理者就能成为管理者的。但是，如果程序员有这个志向和爱好，又积累了这方面的工作经验，自己也感觉在这个方面能够发展，更重要的是有这样的机会，走向管理也是可能的。软件公司的管理者不同于一般公司的管理者；专业能力越强，管理起来就越得心应手。没有专业能力的管理者，遇到的问题会很多，也很难解决。因此，建议程序员最好要把编程、项目设计、项目管理等基础打好，这样，转型到管理者的成功率就会高一些。

### 10.1.2 赢在管理——运转一个高效的开发团队

从组织心理学的角度来看，软件开发是一种典型的聚合型作业——成员虽然形式上独立完成分配的作业，但最后将聚合为完整的作业。最终作业对各部分作业的技术标准和完成时限是有要求的，形成了成员活动的相互制约。项目经理作为一个项目的领导者，是整



个团队的灵魂。

在项目开发过程中，之所以项目经理的角色非常重要，是因为他需要负责项目组开发人员的日常管理，控制项目的进度，负责与设计部门、市场部门及客户之间必要的沟通。这个阶段通常是多个部门的人员共同组成一个项目组，因此，项目一定要保证统一管理，理想状态是项目经理全权负责项目组人员的工作安排、业绩考核、工资奖金等，因为项目经理最了解项目组成员的工作态度和工作业绩。要想在项目经理的位置上做到游刃有余，就需要让你的团队运转得健步如飞。要做到这一点是需要秘诀的。

### (1) 自身修养

#### ① 亲和力

自身除了开发技术过关外，还要具备完美的亲和力。亲和力是指你与团队相互依赖、相互信任能力的大小。亲和力是你领导团队走向成功的基础，如果一个团队的向心力不够，各自为政，那么失败就会在身边陪伴你。要团队的每个成员都信任你，就必须做到关心下属，主动与下属沟通，为下属争取合法权益等。关心下属就是在日常工作中对下属的工作状况、发展方向进行指导，避免其走弯路；在生活中，也对其身体状况进行关心，促进身体和心理健康的恢复。

多找下属沟通是消除误会的润滑剂，同时，也是了解下属内心真实想法的唯一捷径。做软件项目经理的人，在某些事情上处理问题的方式的确会与人不同，也难以令人理解。这个时候，就必须多与下属沟通，逐步达成共识，争取大家的理解和支持。记住，没有下属的理解和支持，你永远无法实现项目管理的规范化。

另外，为下属争取合法权益是软件项目经理的一项重要职责。敢负责任是软件项目经理的基本素质，如果不经常研究工作数据以保障下属的合法权益，就很难让你的团队保持高效率。

#### ② 敢负责任

一个人因为有责任感，才有生存的意义，当经理就是要负责任。软件项目经理关系到一个项目的成败；他必须承担及时汇报项目进度、做成本核算和计算质量系数的责任，同时，也必须担负项目组成员绩效考核、政策落实、预留人才储备等责任。项目经理是整个项目中责任最大的人，如果没有良好的心理素质和应对能力，是无法担负责任的。在实际工作中，软件项目经理主要要负责项目组的人员安排调度、工作分配、工作审核、工作跟踪、项目计划、项目汇报总结、成本核算、利润分配等职责。

### (2) 慧眼识金

在软件项目外包过程中，如何在一大批开发人员中进行甄别和筛选，找到最适合的开发人员，是项目经理必须解决的一大难题。如果草率地选择开发人员，往往会造成项目开发周期延长、质量无法达到要求、成本增加甚至项目彻底失败的严重后果。

要想在公司内部选择出最合适的开发人员，通常可以采用下面的方法来实现：

- 查看开发人员的档案、作品展示信息。详细了解每一名备选者的作品展示信息，通过查看这些信息，可以了解其技术能力和经验。
- 查看开发人员的工作历史记录以及客户的评价信息，如果开发人员完成了较多的项目并且客户给出的评分和评价都很好，这样的开发人员通常是值得信赖的。当然，有时也会有例外。有的开发人员可能因为一些客观原因出现了个别失败的项

目, 这种情况下, 可以通过与同事进一步沟通, 来了解和判断。

- 对于初步通过筛选的程序员, 接下来, 要进一步了解其完成的项目或提供的作品的类型、规模、使用的技术等信息。有的开发人员完成了很多项目或者提供了较多的展示作品, 但如果这些项目或作品与当前的项目差异很大, 甚至完全不同, 就表示专业领域不同, 不一定能够胜任该项目。
- 对比开发人员的外语能力和沟通能力, 如果项目要使用英语, 就需要考察开发人员的外语能力。如果确信开发过程中需要与开发人员进行频繁的沟通, 就需要了解对方是否有相关的沟通条件。

### (3) 团队意识高于一切

对于国内中小型企业来说, 通常, 开发团队就是固定的那么几个人, 没有太多的人才可供选择。在这种情况下, 团队中的成员都已经彼此十分了解了。此时一定要确保这些仅有的团队成员都具有很强的团队意识。因为现在的软件开发已经不再是以前的个人英雄主义时代, 现在更多的情况都是以团队的形式进行系统的设计和开发的, 团队精神变得越来越重要。所以在组织自己的团队时, 特别是挑选人员组建一个长期固定的团队时, 其成员必须具备团队精神。那么, 到底什么是团队精神呢? 包括如下 4 个要点。

#### ① 荣辱与共

作为一个团队中的成员, 一定要牢记这 4 个字, 把整个团队的荣辱放在第一位, 这似乎是集体主义精神的体现, 与当前更为流行的以个人为中心的思想有些不一样。但是, 只有把整个团队的利益放在首位, 团队才能够发展和进步。而团队的发展和进步必定会给其中的每个成员带来好处。

#### ② 交流分享

交流不但在团队中重要, 而且在任何工作中都是非常重要的。人与人之间需要充分交流后, 才能够更好地工作。团队之中, 每个成员之间都应该充分交流, 否则, 会在信息的传达过程中出现理解上的偏差。例如, 如果第一阶段工程(需求分析、概要设计)的负责人不与实施阶段工程(详细设计、编码、测试)的人员充分交流, 那么, 很可能会做出一个客户不满意的产品。

#### ③ 精诚协作

作为一个团队中的成员, 也一定要牢记“精诚协作”这 4 个字。想要实现精诚协作, 首先就要远离“事不关己, 高高挂起”。尽管有些工作不是我们份内的工作, 但是, 既然都属于团队的事情, 那我们就都有责任尽自己所能去做。有人会说: 做得多, 错就多, 帮别人修改了程序, 当这个程序出问题的时候, 就会怪罪到自己的头上。这种情况的确存在, 作者也遇到过多次, 但是, 更应珍惜的是在这个过程中与其他团队成员的交流以及所学习到的知识。任何事情都不可能是完美的, 都具有两面性。而且这样做非常有利于形成真正意义上的团队, 当出现问题的时候, 我们帮助过别人, 当我们自己出现问题的时候, 也会有人帮助我们。

#### ④ 尊重理解

每个人都有自己的长处, 也都有自己的短处, 我们每个人只能尽量做到完美。生活中有很多其他的事情会对工作造成影响, 当发现别人犯错的时候, 我们应该理解, 并且需要以对事不对人的态度去解决问题。



例如，假如测试人员发现开发人员的程序中出现了很多缺陷，就不应该去指责，而是应该记录下来，然后与开发人员一起去分析，提醒他以后不要出现类似的错误。

## 10.2 做好需求分析

视频讲解 光盘：视频\第 10 章\需求分析.avi

本项目是曾经的客户 Authorization 介绍的，他的朋友开了一个网络公司，主要接一些网络硬件方面的业务，现在有一个外包项目：为某科技服务公司开发一个短信群发系统。该公司是一家以后期服务为主的高科技公司。公司为了扩大规模，增强企业的竞争力，提高办公效率，决定聚集部分资金投入软件开发，为企业和用户提供综合信息服务，以向企业提供有偿信息服务为盈利方式。

### 10.2.1 开发背景

企业短信群发系统是基于中国移动、联通、电信、网通直接提供的短信接口，实现与客户指定号码进行短信批量发送和自定义发送的目的。

企业短信群发系统的宗旨是为企业发展、节约开支、提高效益而产生的，它将“打折信息”、“促销活动”、“新品发布”等相关信息发布到目标客户的手机上，可以为企业树立品牌形象或占有市场创造无限的商机，也能为企业大幅降低广告开支。

企业短信群发系统的另一层含义，是指手机终端拥有的短信群发功能，用户编辑一条短信或转发相关短信时，可以编辑 10 条甚至更多的手机号码进行群组发送。

手机短信作为“第五媒体”的地位，已经得到了广泛的认同，与传统大众媒体具有相通、相似、相近的共同点，拥有庞大的受众群体。对于广告主而言，手机短信息广告媒体具有以下不可替代的信息传播优势。

#### (1) 速度快

短信广告的传播不受时间和地域的限制，全国无论在哪个省市，发送到数百万手机用户的广告信息，均可在发送完毕后马上接收到。发布的广告内容可以随时更改，以保证最新的信息可以在最短的时间内传播给广大消费者。

#### (2) 分众性、回报高

短信广告可以直接影响到最有消费力的一族，且同一产品可根据不同的接收对象轻松传递不同的广告信息，以求最大限度地提高客户的购买欲。

#### (3) 投资省

短信广告打破了传统广告媒体定价的行规，广告主可以定好自己的支出预算，定向、定条地发送给目标客户。传播形式时尚、新颖。

#### (4) 精确性

短信广告最大的特性就是直达接收者手机，“一对一”地传递信息，强制性阅读，时效性强，有 100% 的阅读率！可以提高人与广告的接触频率。

#### (5) 蔓延性

短信广告具有很强的散播性，速度快，一分钟即时发送，一瞬间万人传播！接收者可



将信息随身保存，随时咨询广告主，需要时可反复阅读，并可随时发送给感兴趣的朋友。

#### (6) 灵活性

短信广告发布时间极具灵活性，广告主可以根据产品的特点，弹性地选择广告投放时间，甚至具体到在某个特定的时间段内发布。

#### (7) 互动性

短信广告可以让机主与销售终端互动，与大众媒体互动，通过这些，能够使短信用户参与到商业互动中，即短信广告使人们参与互动的机会大增。

#### (8) 低成本

短信广告的发布费用非常低廉，与传统媒体动辄数十万元甚至上百万元的广告费用相比，短信广告的成本几乎可以忽略不计。而通过短信平台提交短信广告，比直接用手机发短信息更便宜，大大降低了广告主的广告发布成本。

#### (9) 瞬时轰动效应强

短信广告具有其他任何一个广告媒体无法比拟的瞬时轰动效果。

### 10.2.2 需求分析

手机短信群发系统的设计有 5 大方面的功能：输入功能、修改功能、删除功能、业务处理功能、快速查询和统计功能。这些功能的设计是基于以下几个原因。

#### (1) 输入功能

输入功能即对日常联系、通信录资料情况、客户企业情况进行添加输入的功能。一个企业的联系客户不会是一成不变的，在日常联系客户的同时，要用电脑进行信息内容处理和更新，首先要把原始数据录入电脑，这是整个系统正常运行的前提。因此，添加输入服务的功能是必须具有的功能。

#### (2) 修改功能和删除功能

市场经济风云变幻，企业联系的人群也相应地跟着变动，所以，必须把企业联系人群及时更新。而企业联系人群的修改功能、删除功能能够及时地改变联系信息，让这些信息与实际情况相符，为企业的用户提供最新的信息。所以，修改功能、删除功能也是不可缺少的。

#### (3) 浏览和快速添加功能

快速浏览数据和快速添加要发送的人的企业列表，将能够给企业用户带来很多方便快捷的服务，能够提高本系统的发送速度。这对企业来说是很有必要的。

## 10.3 项目规划

视频讲解  光盘：视频\第 10 章\项目规划.avi

软件项目规划的目的是预测未来，确定要达到的目标，估计会遇到的问题，并提出实现目标、解决问题的有效方案、方针、措施和手段的过程。在本节的内容中，将详细讲解本章项目规划分析的具体过程。

### 10.3.1 系统目标

根据需求分析的描述以及与用户的沟通，现制定网站实现的目标如下：

- 系统采用人机对话方式，界面美观友好，信息查询灵活、方便，数据存储安全可靠。
- 实施强大的后台审核功能。
- 实现各种查询，如定位查询、模糊查询等。
- 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。
- 最大限度地实现了易维护性和易操作性。
- 界面简洁、框架清晰、美观大方。
- 为充分展现网站的交互性，供求信息网采用动态网页技术实现用户信息在线发布。
- 充分体现用户对网站信息进行检举的权利。

### 10.3.2 划分功能模块

根据系统需求，划分系统功能模块，结构如图 10-1 所示。

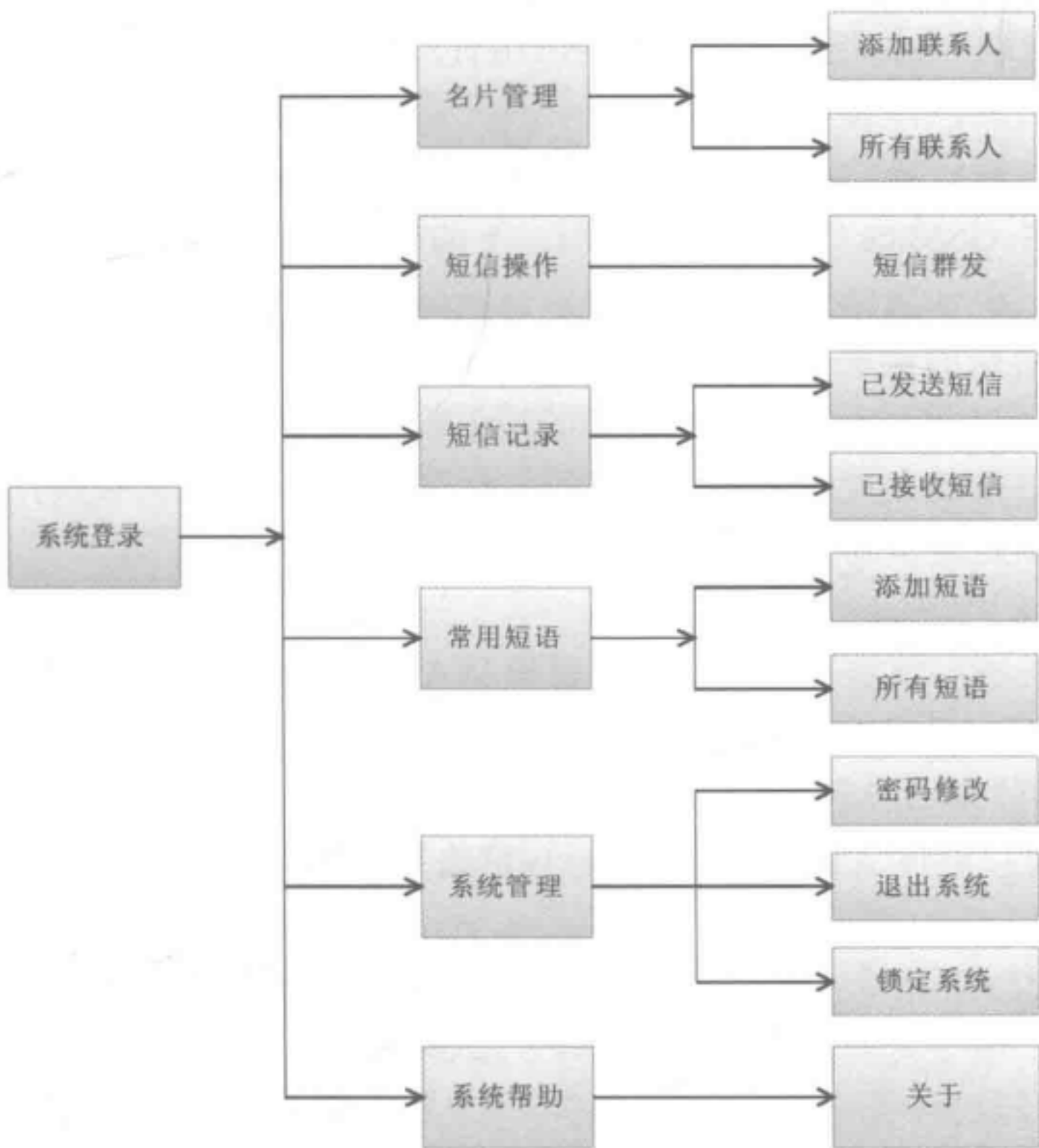


图 10-1 系统功能模块的结构

### 10.3.3 规划运作流程

规划整个项目的运作流程，如图 10-2 所示。

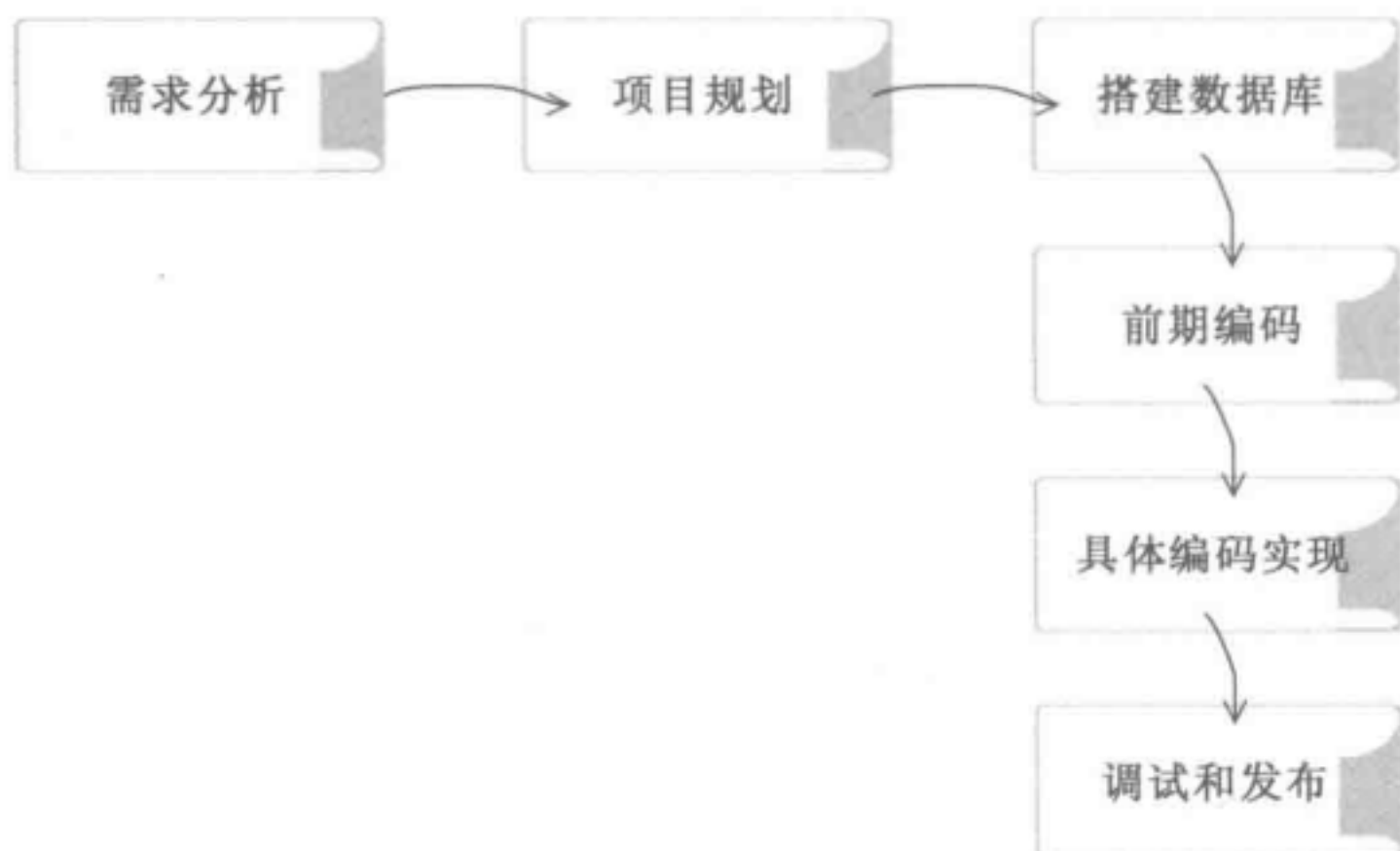


图 10-2 项目的运作流程

另外，因为本项目是一个短信群发平台，所以需要硬件——短信猫。所谓短信猫，其实是一种用来收发短信的设备，它与我们用的手机一样，需要手机 SIM 卡的支持，在需要收发短信的时候，在短信猫里面插入一张我们平时用的手机卡，插上电源，通过(USB 或者串口、网口)数据线与电脑相连，在电脑的应用管理软件中就可以实现短信收发的功能。

在开发本项目时，作者专门购买了一个短信猫，里面附带了一个完整的 SDK 开发包，在里面提供了操作短信猫的函数，这些函数都封装在 `dllforvc.dll` 动态链接库中。常用函数的具体说明在文件 `GSM.cs` 中有简单的介绍。下面简要回顾短信猫的基本知识。

#### (1) 原理

短信猫收发短信的原理、资费与我们平常所用的手机是一样的，但因为短信猫专注于短信收发应用，所以相对于手机，短信猫的短信收发速度更快，可靠性更高，具有可实时发送等优点，在目前的企业短信中应用广泛。标准短信猫 = 短信猫硬件 + 短信猫二次开发包。标准短信猫是短信猫硬件和软件的有机结合体。

#### (2) 开发及应用

短信猫常用的核心模块有西门子短信猫和 WAVECOM。其中，西门子短信猫又分为手机版和工业模块版，手机板主要是 3508，工业短信猫主要是 TC35、TC37 MC 等类型；WAVECOM 主要分为 OEM 和原装两种主要有 1206.2403 等类型。短信猫通过串口 RS-232 与计算机连接，可以通过 AT 指令控制进行短信收发的设备。基于短信猫的开发应用，有以下几种方式。

- 直接使用 AT 指令：通过串口用 AT 指令驱动短信模块收发短信，这是最底层的开发模式，需要对短信模块的 AT 指令相当熟悉。
- 短信猫开发包：短信猫厂商基于串口 AT 指令集成的二次开发包，开发商只需直接调用短信收发 API 即可。
- 短信猫通信中间件：短信猫厂商提供的基于数据库接口的短信收发后台服务软件，是一种更高级的短信开发解决方案。

短信猫的二次开发流程如下。

① 短信相关应用需要发送短信时，将短信接收者与内容提交到短信发送队列；同时从短信接收队列中读取收到的短信。



② 软件开发商需要开发独立的短信后台服务，从短信发送队列中读取短信，调用短信猫开发包发送短信；同时通过调用短信猫开发包读取设备已收到的短信，放入短信接收队列中。

③ 短信猫开发包内部实际上是通过串口通信与短信猫连接，通过 AT 指令驱动短信模块收发短信。

因为短信猫是串行通信设备，必须串行提交短信发送，而且提交后，必须等到其有回应后才能提交下一条，否则会造成短信猫死机。特别是，现在大部分应用都是多用户应用，如果存在多线程同时并发操作短信模块，也会造成短信猫死机。即使是针对同一短信模块的收发，也必须为一前一后串行，而不能通过收发两个并发线程来操作。因此，建议使用短信队列，常用的方式就是使用数据库表。

## 10.4 搭建数据库

视频讲解 光盘：视频\第 10 章\搭建数据库.avi

因为企业的客户有限，所以本项目使用 Access 数据库，命名为 ddd，其中包含了 4 张数据表。下面分别给出数据表概要说明、数据库 E-R 图分析及主要数据表的结构。

### 10.4.1 数据库 E-R 图分析

根据前面的需求分析，规划出项目中的数据库实体有用户信息实体、常用短语实体、已接受短信信息实体、所有联系人信息实体和已发送短信实体。

(1) 用户信息实体 E-R 图如图 10-3 所示。

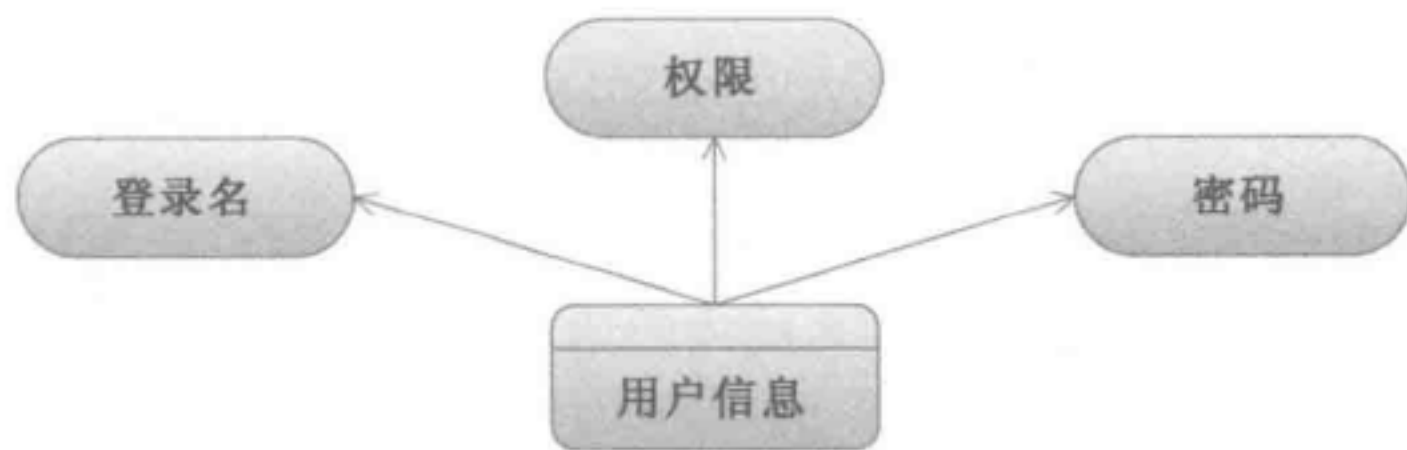


图 10-3 用户信息实体 E-R 图

(2) 已发短信实体 E-R 图如图 10-4 所示。



图 10-4 已发短信实体 E-R 图

(3) 电话簿实体 E-R 图如图 10-5 所示。



图 10-5 电话簿实体 E-R 图

(4) 常用短语实体 E-R 图如图 10-6 所示。



图 10-6 常用短语实体 E-R 图

(5) 已接收短信实体 E-R 图如图 10-7 所示。



图 10-7 已接收短信实体 E-R 图

## 10.4.2 数据表的结构

在设计完数据库实体 E-R 图之后，下面将根据实体 E-R 图设计数据表的结构。

- (1) tb\_Admin: 用户信息表，存储系统内的用户信息。具体结构如表 10-1 所示。
- (2) tb\_note: 常用短语信息表，存储收费供求信息和推荐供求信息。具体数据表结构如表 10-2 所示。
- (3) tb\_Resvice: 已接收短信信息表，存储已接收的短信信息。具体结构如表 10-3 所示。
- (4) tb\_TelSend: 已发信息表，存储系统已发送的信息。具体结构如表 10-4 所示。

表 10-1 用户信息表

| 字段名称          | 数据类型 | 是否主键 | 功能描述 |
|---------------|------|------|------|
| ID            | 自动编号 | 是    | 编号   |
| AdminUserName | 文本   | 否    | 登录名  |
| AdminUserPwd  | 文本   | 否    | 密码   |
| power         | 文本   | 否    | 权限   |

表 10-2 常用短语信息表

| 字段名称 | 数据类型 | 是否主键 | 功能描述 |
|------|------|------|------|
| ID   | 自动编号 | 是    | 编号   |
| type | 文本   | 否    | 文本   |
| note | 备注   | 否    | 备注   |

表 10-3 已接收短信信息表

| 字段名称       | 数据类型 | 是否主键 | 功能描述     |
|------------|------|------|----------|
| ID         | 自动编号 | 是    | 编号       |
| smsnum     | 文本   | 否    | 发送短信的号码  |
| smscontent | 备注   | 否    | 已接收短信的内容 |

表 10-4 已发信息表

| 字段名称       | 数据类型 | 是否主键 | 功能描述  |
|------------|------|------|-------|
| ID         | 自动编号 | 是    | 编号    |
| TelNum     | 文本   | 否    | 接收者号码 |
| TelContent | 备注   | 否    | 短信内容  |
| TelTime    | 文本   | 否    | 发送时间  |

(5) tb\_tel: 常用联系人信息表, 用于存储系统的常用联系人信息。具体数据表结构如图 10-5 所示。

表 10-5 常用联系人信息表

| 字段名称     | 数据类型 | 是否主键 | 功能描述 |
|----------|------|------|------|
| ID       | 自动编号 | 是    | 编号   |
| UserName | 文本   | 否    | 姓名   |
| UserSex  | 文本   | 否    | 性别   |
| UserTel  | 文本   | 否    | 手机号码 |

(6) 数据结构看似很简单, 但实际上具体划分时有很大的学问。通常有垂直划分和水平划分两种方式。

- ① 垂直划分: 按照功能划分, 把数据分别放到不同的数据库和服务器。
- 当一个网站刚刚开始创建时, 可能只是考虑一天只有几十或者几百个人访问, 数据库可能就是个 DB, 所有表都放一起, 一台普通的服务器可能就够了, 而且开发人员也非常高兴, 信心十足, 因为所有的表都在一个库中, 这样查询语句就可以随便关联了, 多美的一件事情啊。但是, 随着访问压力的增加, 读写操作不断增加, 数据库的压力绝对会越来越大, 可能接近极限, 这时, 可能会想到增加从服务器, 做什么集群之类的, 可是问题又来了: 数据量也会快速地增长。
- 这时, 可以考虑对读写操作进行分离, 根据业务, 把不同的数据放到不同的库中。其实, 在一个大型而且臃肿的数据库中, 表和表之间的数据很多是没有关系的, 或者更加不



需要(join)操作,理论上,就应该把它们分别放到不同的服务器上。例如,用户的收藏夹的数据和博客的数据库就可以放到两个独立的服务器中。这就叫垂直划分(其实叫什么不重要)。具体如图 10-8 所示。

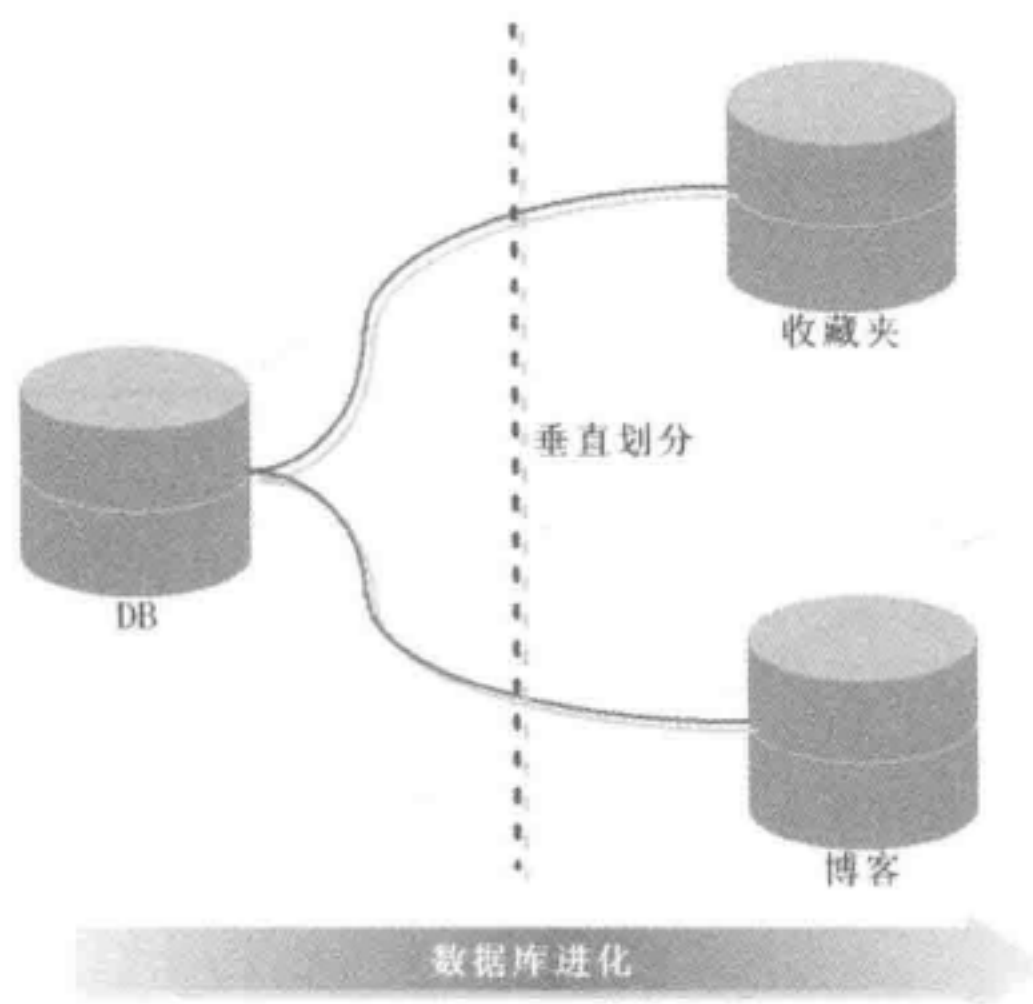


图 10-8 垂直划分演示

② 水平划分：把一个表的数据划分到不同的数据库，两个数据库的表结构一样。应该根据一定的规则划分，可以根据数据的产生者来做引导，上面的数据是由人产生的，可以根据人的 id 来划分数据库。然后再根据一定的规则，先获知数据在哪个数据库中，具体如图 10-9 所示。

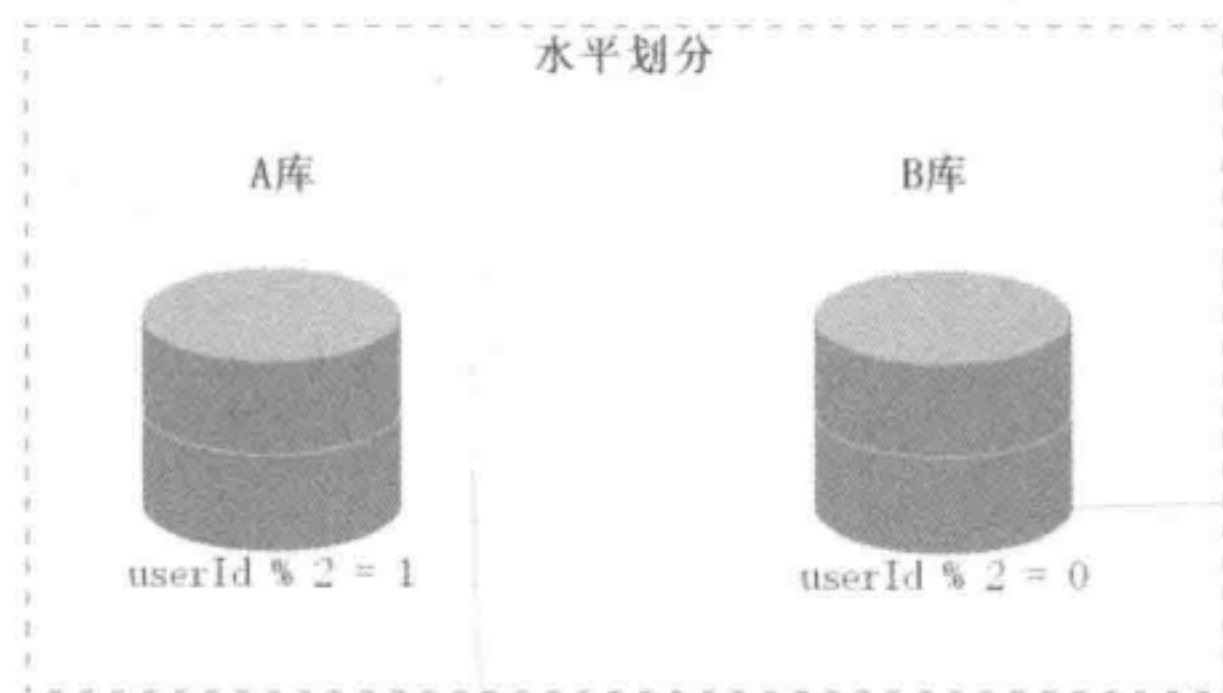


图 10-9 水平划分演示

其实很多大型网站都经历了数据库垂直划分和水平划分的阶段。这可以根据经验来确定，不一定有某些硬性的规则。

## 10.5 设计公共类

视频讲解 光盘：视频\第 10 章\设计公共类.avi

设计公共类的目的，是以类的形式来组织、封装一些常用的方法和事件，这样可以提

高代码的重用率，并方便了对代码的管理。本系统中使用了两个公共类，分别是 ConnClass 和 GSM。

### 10.5.1 ConnClass 类

类 ConnClass 的功能是建立与 Access 数据库的连接，具体实现代码如下所示：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.OleDb;
using System.Windows.Forms;
namespace SMS.BaseClass
{
    class ConnClass
    {
        public static OleDbConnection DataConn()
        {
            string strg = Application.StartupPath.ToString();
            strg = strg.Substring(0, strg.LastIndexOf(@"\"));
            strg = strg.Substring(0, strg.LastIndexOf(@"\"));
            strg += @"\DataBase";
            strg += @"\db_SMS.mdb";
            return new OleDbConnection(
                "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + strg);
        }
    }
}
```

### 10.5.2 GSM 类

GSM 类的功能是封装各个操作短信猫的方法，具体实现代码如下所示：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;
namespace SMS.BaseClass
{
    class GSM
    {
        //初始化 gsm modem, 并连接 gsm modem
        [DllImport("dllforvc.dll",
            EntryPoint = "GSModemInitNew",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
        public static extern bool GSModemInitNew(
            string device,
            string baudrate,
            string initstring,
            string charset,
            bool swHandshake,
            string sn);

        //获取短信猫新的标识号码
        [DllImport("dllforvc.dll",
            EntryPoint = "GSModemGetSnInfoNew",
```

```
        CharSet = CharSet.Ansi,  
        CallingConvention = CallingConvention.StdCall)]  
public static extern string GSMModemGetSnInfoNew(string device, string baudrate);  
  
//获取当前通信端口  
[DllImport("dllforvc.dll",  
    EntryPoint = "GSMModemGetDevice",  
    CharSet = CharSet.Ansi,  
    CallingConvention = CallingConvention.StdCall)]  
public static extern string GSMModemGetDevice();  
  
//获取当前通信波特率  
[DllImport("dllforvc.dll",  
    EntryPoint = "GSMModemGetBaudrate",  
    CharSet = CharSet.Ansi,  
    CallingConvention = CallingConvention.StdCall)]  
public static extern string GSMModemGetBaudrate();  
  
//断开连接并释放内存空间  
[DllImport("dllforvc.dll",  
    EntryPoint = "GSMModemRelease",  
    CharSet = CharSet.Ansi,  
    CallingConvention = CallingConvention.StdCall)]  
public static extern void GSMModemRelease();  
  
//取得错误信息  
[DllImport("dllforvc.dll",  
    EntryPoint = "GSMModemGetErrorMsg",  
    CharSet = CharSet.Ansi,  
    CallingConvention = CallingConvention.StdCall)]  
public static extern string GSMModemGetErrorMsg();  
  
//发送短信息  
[DllImport("dllforvc.dll",  
    EntryPoint = "GSMModemSMSsend",  
    CharSet = CharSet.Ansi,  
    CallingConvention = CallingConvention.StdCall)]  
public static extern bool GSMModemSMSsend(  
    string serviceCenterAddress,  
    int encodeval,  
    string text,  
    int textlen,  
    string phonenummer,  
    bool requestStatusReport);  
  
//接收短信息, 返回字符串格式为: 手机号码|短信内容||手机号码|短信内容||  
//RD_opt 为 1, 接收短信息后不做任何处理; 为 0, 接收后删除信息  
[DllImport("dllforvc.dll",  
    EntryPoint = "GSMModemSMSReadAll",  
    CharSet = CharSet.Ansi,  
    CallingConvention = CallingConvention.StdCall)]  
public static extern string GSMModemSMSReadAll(int RD_opt);  
}  
}
```

上述代码中, 因为要使用 `dllforvc.dll`, 所以须引用 `System.Runtime.InteropServices` 命名空间。

到此为止, 整个项目的前期编码工作结束。可以发现, 在使用 C# 开发项目时, 公共类必不可少, 既可以节省代码编写量, 也可以实现面向对象, 何乐而不为呢? 但是究竟哪些



信息常作为公共类呢，在此做了一个简单的总结，并全部写在一个代码文件里。具体如下：

```

/*****
** 1.获取数据库的连接，返回值需判断是否为 null-----GetSqlConnection
* 2.根据 Select 查询语句，返回 DataSet-----GetDataSet
* 3.使用数据库内容填充 DataGridView-----FillDataGridViewFromSQLString
* 4.使用数据库内容填充 DataGridView-----FillDataGridViewFromSQLString(重载)
* 5.返回 SQL 语句所查询出来的行数-----GetRowCount
* 6.填充下拉列表-----FillComboBox*
* 7.由一条 SQL 语句生成一个 DataReader；返回值需要判断是否为空-----GetDataReader
* 10.返回单个查询数据：第一列，第一行的值-----GetFirstData
* 9.对数据库中的一条记录操作：增、删、更新-----ExecuteCommand
* 10.对数据库进行增删改操作-----ExecuteCommand2
* 11.判断 str 是不是全由数字构成-----IsNumeric
* 12.检测含有中文字符串的实际长度-----len
*****/
using System;
using System.Data.SqlClient;
using System.Data;
using System.Windows.Forms;
namespace Tayside.Common
{
    /// <summary>
    /// DataBase 的摘要说明
    /// </summary>
    public class DataBase
    {
        public DataBase()
        {
        }
        /// <summary>
        /// 1.获取数据库的连接，返回值需判断是否为 null
        /// </summary>
        /// <returns></returns>
        public static SqlConnection GetSqlConnection()
        {
            string strCnn = "Server=192.1610.12.136;database=Tayside;user id=sa;password=";
            try
            {
                SqlConnection sqlCnn = new SqlConnection(strCnn);
                sqlCnn.Open();
                return sqlCnn;
            }
            catch (Exception ee)
            {
                string temp = ee.Message;
                return null;
            }
        }

        /// <summary>
        /// 获取 SqlCommand 对象
        /// </summary>
        /// <returns></returns>
        public static SqlCommand GetSqlCommand()
        {
            SqlConnection sqlCnn = GetSqlConnection();
            if (sqlCnn == null)
                return null;
            else

```

```

    {
        SqlCommand sqlCmm = new SqlCommand();
        sqlCmm.Connection = sqlCnn;
        return sqlCmm;
    }
}

/// <summary>
/// 2.根据 Select 查询语句, 返回 DataSet
/// </summary>
/// <param name="strSql">Select SQL 语句</param>
/// <returns>返回值需判断是否为空</returns>
public static DataSet GetDataSet(string strSql)
{
    try
    {
        using(SqlConnection sqlCnn = GetSqlConnection())
        {
            SqlDataAdapter dataAdapter = new SqlDataAdapter(strSql, sqlCnn);
            DataSet dataSet = new DataSet();
            dataAdapter.Fill(dataSet);
            return dataSet;
        }
    }
    catch
    {
        return null;
    }
}

/// <summary>
/// 3.使用数据库内容填充 DataGridView
/// </summary>
/// <param name="dataGridView">要填充的 DataGridView</param>
/// <param name="strSql">要获取数据库内容的 SQL 字符串</param>
/// <returns></returns>
public static bool FillDataGridViewFromSQLString(DataGridView dataGridView, string strSql)
{
    try
    {
        DataSet ds = GetDataSet(strSql);
        dataGridView.SetDataSource(ds);
        return true;
    }
    catch(Exception ee)
    {
        string t = ee.Message;
        return false;
    }
}

/// <summary>
/// 4.使用数据库内容填充 DataGridView
/// </summary>
/// <param name="dataGridView">要填充的 DataGridView</param>
/// <param name="strSql">要获取数据库内容的 SQL 字符串</param>
/// <param name="table">要填充 DataGridView 的表名</param>
/// <returns></returns>
public static bool FillDataGridViewFromSQLString(
    DataGridView dataGridView, string strSql, string table)
{

```

```

try
{
    DataSet ds = GetDataSet(strSql);
    dataGrid.SetDataBinding(ds, table);

    return true;
}
catch(Exception ee)
{
    string t = ee.Message;
    return false;
}
}

/// <summary>
/// 5、返回 SQL 语句所查询出来的行数
/// </summary>
/// <param name="strSql"></param>
/// <returns></returns>
public static int GetRowCount(string strSql)
{
    DataSet ds = GetDataSet(strSql);
    int count = ds.Tables[0].Rows.Count;
    return count;
}

/// <summary>
/// 6.填充下拉列表
/// </summary>
/// <param name="cmBox">要填充的 ComboBox</param>
/// <param name="strSql">查询语句</param>
/// <returns>是否成功</returns>
public static bool FillComboBox(ComboBox cmBox, string strSql)
{
    try
    {
        using(SqlConnection sqlCnn = GetSqlConnection())
        {
            SqlDataReader dr = GetDataReader(strSql);
            while(dr.Read())
            {
                cmBox.Items.Add(dr.GetValue(0));
            }
            return true;
        }
    }
    catch
    {
        return false;
    }
}

/// <summary>
/// 7.由一条 SQL 语句生成一个 DataReader: 返回值需要判断是否为空
/// </summary>
/// <param name="strSql">要使用的 SQL 语句</param>
/// <returns></returns>
public static SqlDataReader GetDataReader(string strSql)
{
    try
    {

```



```
        SqlConnection sqlCnn = GetSqlConnection();
        SqlCommand sqlCmm = new SqlCommand(strSql, sqlCnn);
        return sqlCmm.ExecuteReader(CommandBehavior.CloseConnection);
    }
    catch
    {
        return null;
    }
}

/// <summary>
/// 10.返回单个查询数据：第一列，第一行的值
/// </summary>
/// <param name="strSql">Select SQL 语句</param>
/// <returns></returns>
public static string GetFirstData(string strSql)
{
    try
    {
        using( SqlConnection sqlCnn = GetSqlConnection() )
        {
            SqlCommand sqlCmm = new SqlCommand(strSql, sqlCnn);
            return sqlCmm.ExecuteScalar().ToString();
        }
    }
    catch
    {
        return "";
    }
}

/// <summary>
/// 9.对数据库中的一条记录操作：增、删、更新
/// </summary>
/// <param name="strSql">要执行的 SQL 语句</param>
/// <returns>返回执行是否成功</returns>
public static bool ExecuteCommand(string strSql)
{
    try
    {
        using( SqlConnection sqlCnn = GetSqlConnection() )
        {
            SqlCommand sqlCmm = new SqlCommand(strSql, sqlCnn);
            int temp = sqlCmm.ExecuteNonQuery();
            return temp == 1;
        }
    }
    catch
    {
        return false;
    }
}

/// <summary>
/// 10.对数据库进行增删改操作
/// </summary>
/// <param name="strSql"></param>
/// <returns></returns>
public static bool ExecuteCommand2(string strSql)
{

```

```

try
{
    using( SqlConnection sqlCnn = GetSqlConnection())
    {
        SqlCommand sqlCmm = new SqlCommand(strSql, sqlCnn);
        int temp = sqlCmm.ExecuteNonQuery();
        return true;
    }
}
catch
{
    return false;
}
}

/// <summary>
/// 11.判断str是不是全由数字构成
/// </summary>
/// <param name="str"></param>
/// <returns></returns>
public static bool IsNumeric(string str)
{
    if (str==null || str.Length==0)
        return false;
    foreach(char c in str)
    {
        if (!Char.IsNumber(c))
        {
            return false;
        }
    }
    return true;
}

/// <summary>
/// 12.检测含有中文字符串的实际长度
/// </summary>
/// <param name="str">字符串</param>
public static int len(string str)
{
    System.Text.ASCIIEncoding n = new System.Text.ASCIIEncoding();
    byte[] b = n.GetBytes(str);
    int l = 0; // l 为字符串的实际长度
    for (int i=0;i <= b.Length-1;i++)
    {
        if (b[i] == 63) //判断是否为汉字或全角符号
        {
            l++;
        }
        l++;
    }
    return l;
}
}
}

```

在 C# 程序项目中，连接到数据库服务器通常由几个需要较长时间的步骤组成。必须建立物理通道(例如套接字或命名管道)，必须与服务器进行初次连接，必须分析连接字符串信息，必须由服务器对连接进行身份验证等。实际上，大部分的应用程序都是使用一个或几

个不同的连接配置。当应用程序的数据量和访问量大的时候，这意味着在运行应用程序的过程中，许多相同的连接将反复地被打开和关闭，从而会引起数据库服务器效率低下甚至引发程序崩溃。为了确保应用程序的稳定和降低成本，可以在 ADO.NET 中使用称为连接池的优化方法来管理和维护连接。

连接池可以减少创建连接的次数，定义最小连接数(固定连接数)，当用户在连接上调用 Open 时，连接池就会检查池中是否有可用的连接。如果发现有连接可用，会将该连接返回给调用者，而不是创建新连接。应用程序在该连接上调用 Close 时，连接池会判断该连接是否在最小连接数之内，如果“是”，会将连接回收至活动连接池中，而不是真正关闭连接，否则将烧毁连接。连接返回到池中后，即可在下一个 Open 调用中重复使用。

在使用数据库连接时需要注意什么事项呢？必须及时关闭不用的连接池。因为每次打开连接，就会建立一条到服务器数据库的通道，每台服务器的总通道数量是有限的，大概就 2000 个左右，而且内存占用也会比较大，虽然打开和关闭有点麻烦，但是双方面考虑之后，还是用完即关闭好一点，这样可以节约内存。在本项目的公共类中，就及时关闭了不用的连接。如果不写关闭代码，一个终端退出后，数据库通道还是占用的，那么很多人连接之后，就把通道占满了，这样其他用户就连接不上了，除非重启数据库服务或重启服务器，但这样做是得不偿失的。

## 10.6 具体编码

视频讲解 光盘：视频\第 10 章\具体编码.avi

经过几天的加班加点，前期公共类的编码工作就全部完成了。现在已经有了需求分析和公共类代码，后面的具体编码工作就变得十分简单了。

### 10.6.1 登录验证模块

此模块的功能是验证用户的身份，确保只有系统的合法用户才能登录系统。本模块的实现文件是 Login.cs，下面讲解其实现流程。

(1) 单击“登录”按钮后，首先判断是否输入了用户名和密码，然后判断输入的用户名和密码是否合法，确保只有系统的合法用户才能登录系统。对应的实现代码如下所示：

```
private void btnLogin_Click(object sender, EventArgs e)
{
    try
    {
        if (txtLoginName.Text == "")
        {
            MessageBox.Show("用户名不能为空");
        }
        else
        {
            if (txtLoginPwd.Text == "")
            {
                MessageBox.Show("密码不能为空");
            }
            else
            {
                // ... (rest of the code)
            }
        }
    }
    catch { }
}
```



```

{
    OleDbConnection conn = BaseClass.ConnClass.DataConn();
    conn.Open();
    OleDbCommand cmd = new OleDbCommand(
        "select count(*) from tb_Admin where AdminUserName='"
        + txtLoginName.Text + "' and AdminUserPwd='" + txtLoginPwd.Text + "'",
        conn);
    int i = Convert.ToInt32(cmd.ExecuteScalar());
    if (i > 0)
    {
        Main main = new Main();
        main.adminname = txtLoginName.Text;
        main.admintime = DateTime.Now.ToShortDateString();
        main.Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("用户名或者密码错误");
    }
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString());
}
}

```

(2) 当用户输入合法数据并按下 Enter 键后，就可以登录系统，单击“取消”按钮后，将关闭窗体。对应的实现代码如下所示：

```

private void btnCancel_Click(object sender, EventArgs e)
{
    this.Close();
}
private void txtLoginPwd_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        btnLogin_Click(sender, e);
    }
}

```

至此，整个登录验证模块设计完毕，执行后的效果如图 10-10 所示。

图 10-10 登录验证表单界面

## 10.6.2 主窗体模块

主窗体是用户登录后显示的主界面，分为如下 3 部分。

- 顶部菜单：实现一些对应的操作处理。
- 中间提示：显示对应的提示信息。
- 底部提示：显示系统的当前状态信息。

主窗体界面的效果如图 10-11 所示。

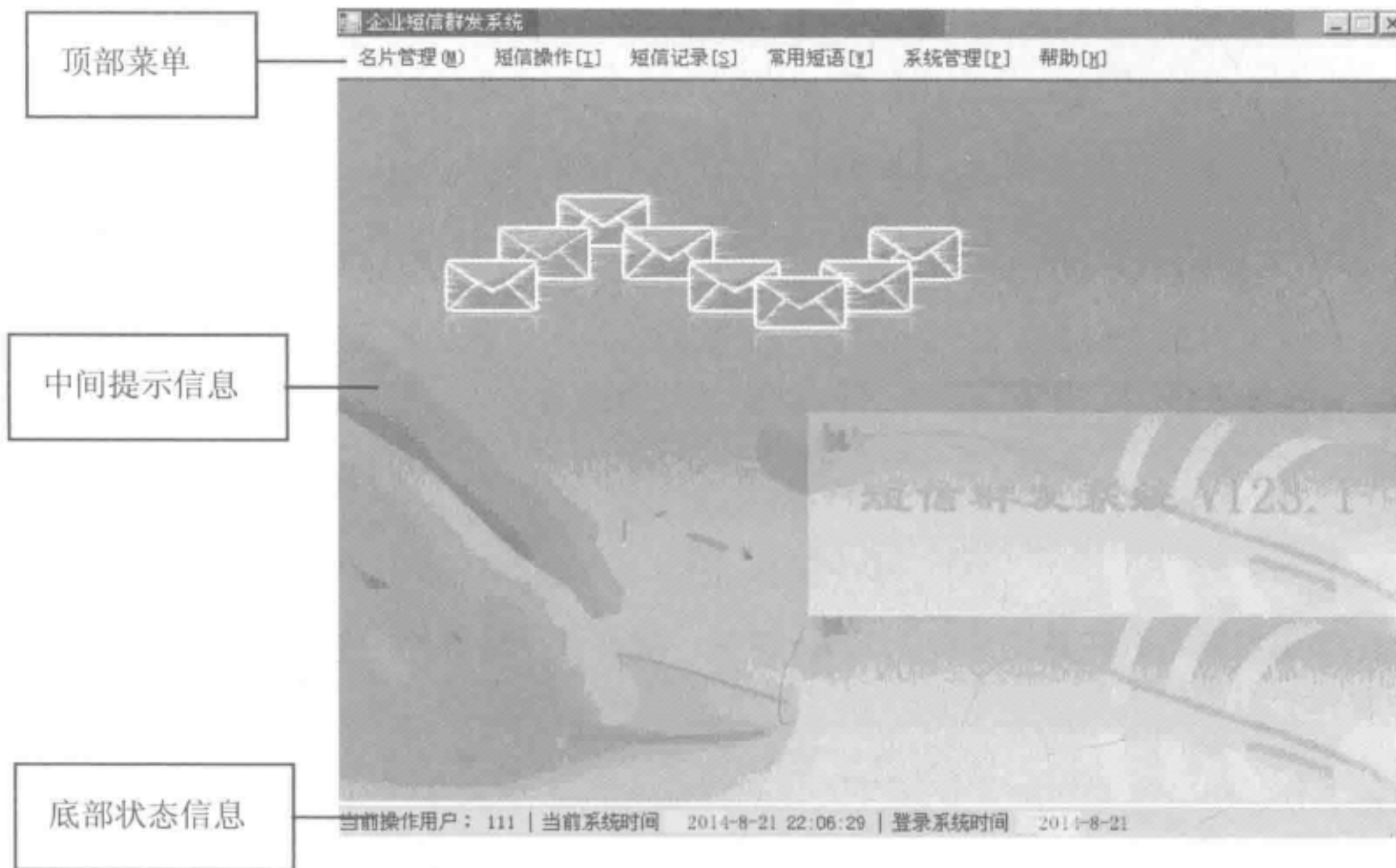


图 10-11 系统主界面

首先需要进行窗体界面设计，在 Visual Studio 中新建一个窗体，并分别插入如下菜单和对应的子菜单。

- (1) 名片管理。
  - 增加联系人：打开添加新的联系人窗体。
  - 所有联系人：打开显示所有的联系人窗体。
- (2) 短信操作。短信群发：实现短信群发处理。
- (3) 短信记录。
  - 已发送短信：打开显示已发送的短信窗体。
  - 已接收短信：打开显示已接收的短信窗体。
- (4) 常用短语。
  - 添加短语：打开显示添加短语的窗体。
  - 所有短语：打开显示所有短语的窗体。
- (5) 系统管理。
  - 密码修改：打开显示修改密码的窗体。
  - 退出系统：退出系统。

- 锁定系统：锁定整个系统。

(6) 帮助。关于：打开“关于”窗体对话框。

主界面的实现文件是 Main.cs，具体实现代码如下所示：

```
namespace SMS
{
    public partial class Main : Form
    {
        public Main()
        {
            InitializeComponent();
        }
        public string adminname;
        public string admintime;
        private void Form1_Load(object sender, EventArgs e)
        {
            toolStripStatusLabel5.Text = DateTime.Now.ToString();
            toolStripStatusLabel2.Text = adminname;
            toolStripStatusLabel10.Text = admintime;
            OleDbConnection conn = BaseClass.ConnClass.DataConn();
            OleDbCommand cmd = new OleDbCommand(
                "select power from tb_Admin where AdminUserName='"+adminname+"'", conn);
            conn.Open();
            string userPower = Convert.ToString(cmd.ExecuteScalar());
            if (userPower == "0")
            {
                名片管理 MToolStripMenuItem.Enabled = false;
                短信记录 SToolStripMenuItem.Enabled = false;
                常用短语 MToolStripMenuItem.Enabled = false;
            }
            conn.Close();
        }

        private void toolStripButton1_Click(object sender, EventArgs e)
        {
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            toolStripStatusLabel5.Text = DateTime.Now.ToString();
        }
        private void 关于 OToolStripMenuItem_Click(object sender, EventArgs e)
        {
            AboutBox1 ab = new AboutBox1();
            ab.MdiParent = this;
            ab.Show();
        }
        private void 所有短语 GToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Cydy fydy = new Cydy();
            fydy.ShowDialog();
        }
        private void 添加短语 KToolStripMenuItem_Click(object sender, EventArgs e)
        {
            AddDy adddy = new AddDy();
            adddy.MdiParent = this;
            adddy.Show();
        }
        private void 所有联系人 CToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }
    }
}
```



```
Tel tel = new Tel();
tel.MdiParent = this;
tel.Show();
}
private void 已发送短信FToolStripMenuItem_Click(object sender, EventArgs e)
{
    Yfxx yfxx = new Yfxx();
    yfxx.MdiParent = this;
    yfxx.Show();
}
private void 已接收短信GToolStripMenuItem_Click(object sender, EventArgs e)
{
    Resvice res = new Resvice();
    res.MdiParent = this;
    res.Show();
}
private void 退出系统ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("确定要退出本系统吗?", "警告", MessageBoxButtons.OKCancel,
        MessageBoxIcon.Warning) == DialogResult.OK)
    {
        Application.Exit();
    }
}
private void 密码修改ToolStripMenuItem_Click(object sender, EventArgs e)
{
    ChangePwd pwd = new ChangePwd();
    pwd.MdiParent = this;
    pwd.Show();
}
private void 增加联系人SToolStripMenuItem_Click(object sender, EventArgs e)
{
    AddUser users = new AddUser();
    users.MdiParent = this;
    users.Show();
}
private void 锁定系统ToolStripMenuItem_Click(object sender, EventArgs e)
{
    Lock formlock = new Lock();
    formlock.Owner = this;
    formlock.ShowDialog();
}
private void frmMain_FormClosed(object sender, FormClosedEventArgs e)
{
    if (MessageBox.Show("确定要退出本系统吗?", "警告", MessageBoxButtons.OKCancel,
        MessageBoxIcon.Warning) == DialogResult.OK)
    {
        Application.Exit();
    }
    else
    {
        Main main = new Main();
        main.Show();
    }
}
private void 短信群发RToolStripMenuItem_Click(object sender, EventArgs e)
{
    SendSMS sendsms = new SendSMS();
    sendsms.MdiParent = this;
    sendsms.Show();
}
```

```
}  
}
```

至此，整个主界面模块设计完毕，执行后的效果如图 10-12 所示。

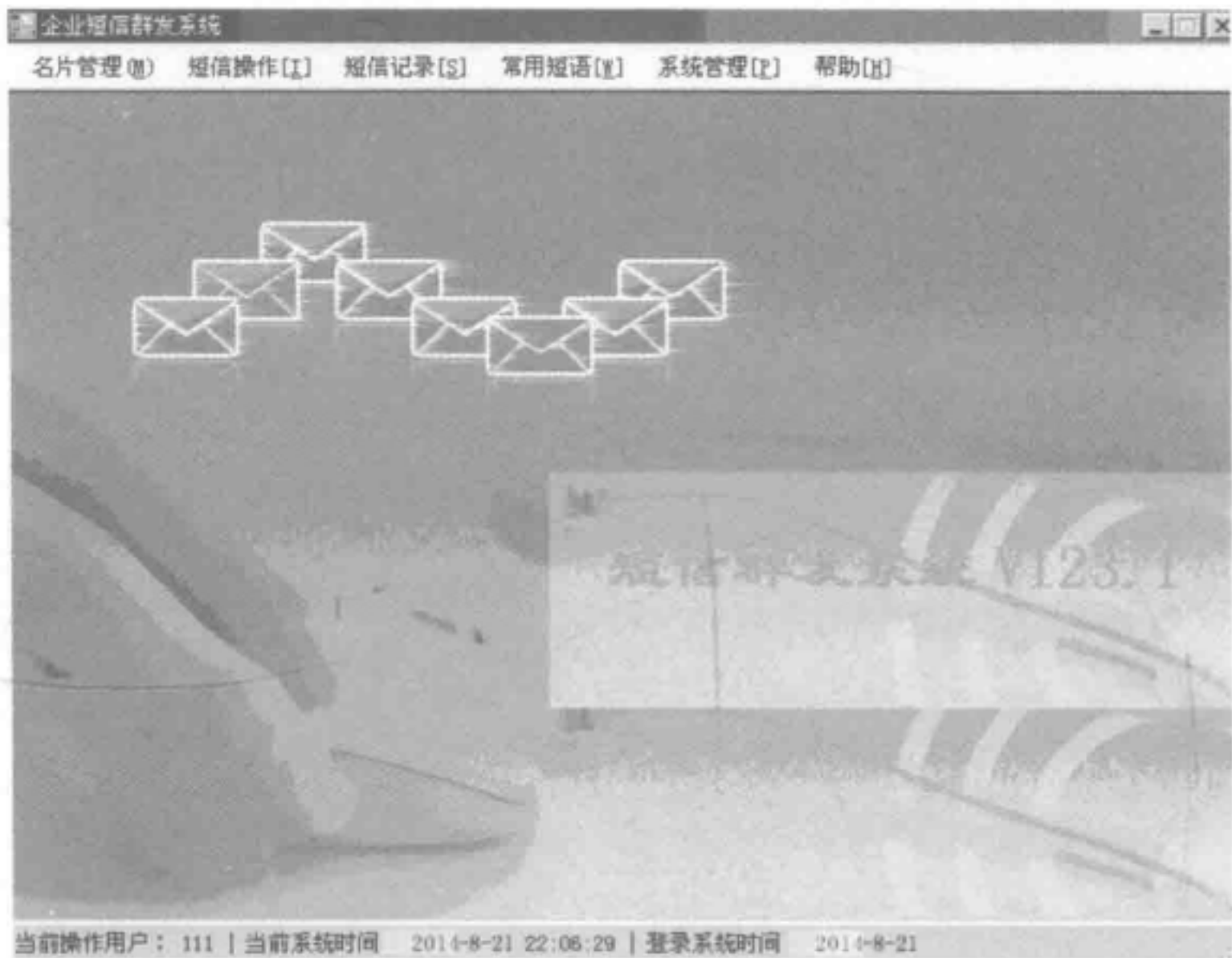


图 10-12 主界面模块

### 10.6.3 短信群发模块

此模块是整个系统的核心，功能是实现系统群发。设计后的窗体效果如图 10-13 所示。

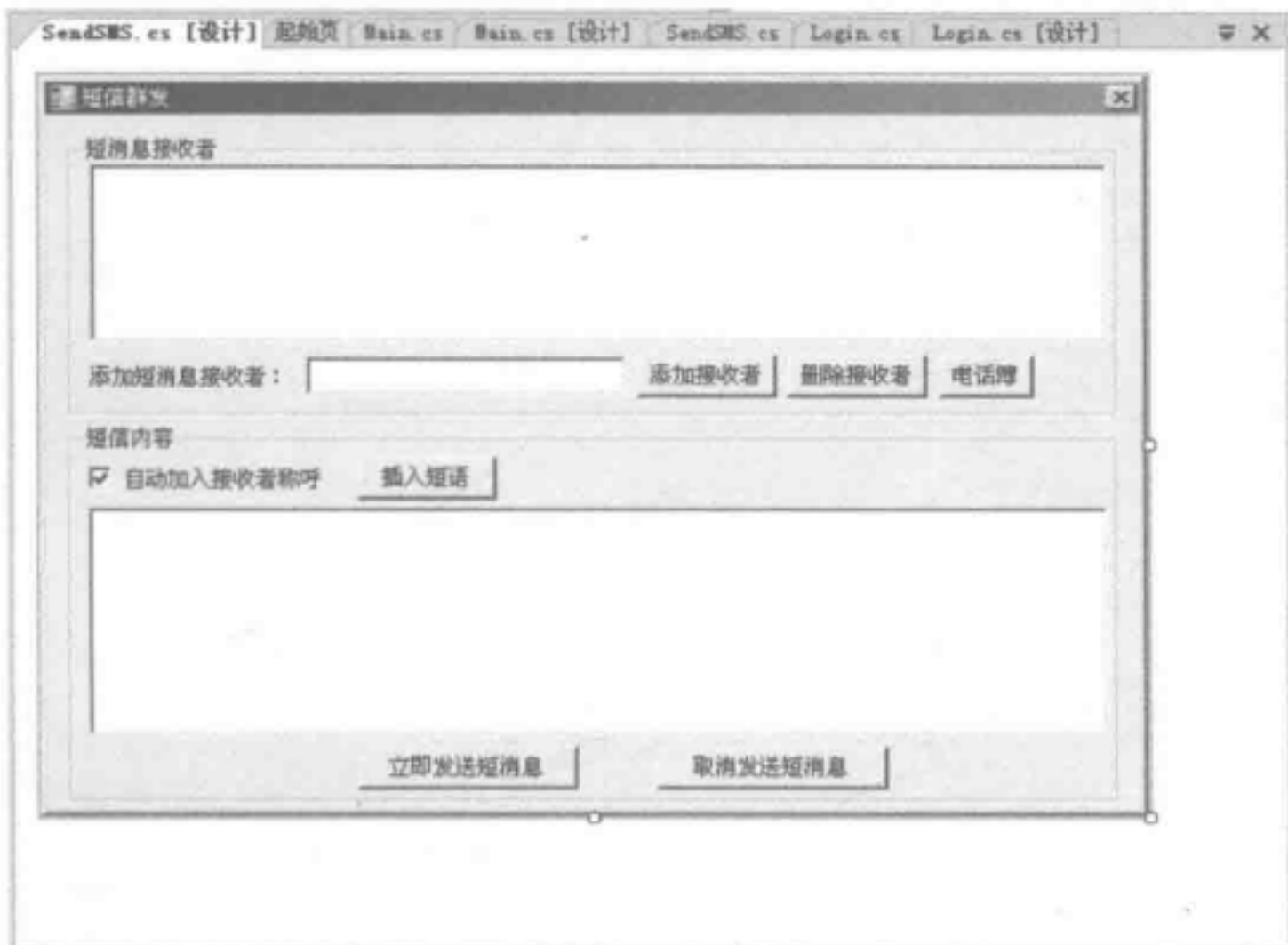


图 10-13 “短信群发”窗体

本模块的实现文件是 SendSMS.cs，下面讲解具体的实现流程。

(1) 定义全局变量，调用方法 SendSMS()实现群发处理。对应的代码如下所示：

```
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Data.OleDb;
namespace SMS
{
    public partial class SendSMS : Form
    {
        public SendSMS()
        {
            InitializeComponent();
        }
        public string content;
        public string ResviceNum;
        public static string NUM;
        public static string JQCOM;
        public static string BTL;
        public string Power = "YIWU-IJDD-CDQW-JDWG";
        public static int k = 0;
        public static int j = 0;
        private void frmSendSMS_Load(object sender, EventArgs e)
        {
            txtaddnum.Focus();
            NUM = BaseClass.GSM.GSMModemGetSnInfoNew(JQCOM, BTL); //机器号码
            JQCOM = BaseClass.GSM.GSMModemGetDevice(); //COM1
            BTL = BaseClass.GSM.GSMModemGetBaudrate(); //波特率
        }
    }
}
```

(2) 定义方法 AddMessage, 添加短信信息。对应的代码如下所示:

```
private void AddMessage(string num, string strcontent, string sendtime)
{
    OleDbConnection conn = BaseClass.ConnClass.DataConn();
    conn.Open();
    OleDbCommand cmd = new OleDbCommand(
        "insert into tb_TelSend([TelNum],[TelContent],[TelTime]) values('"
        + num + "','" + strcontent + "','" + sendtime + "')", conn);
    cmd.ExecuteNonQuery();
    conn.Close();
}
private void SendSms(string strcom, string strbtl)
{
    try
    {
        ////连接设备
        if (BaseClass.GSM.GSMModemInitNew(strcom, strbtl, null, null, false, Power)
            == false)
        {
            MessageBox.Show("设备连接失败!" + BaseClass.GSM.GSMModemGetErrorMsg(),
                "提示", MessageBoxButtons.OK);
            return;
        }
        // 发送短信
        j = lvInceptNum.Items.Count;
        for (int i=0; i<lvInceptNum.Items.Count; i++)
        {
            AddMessage(lvInceptNum.Items[i].Text, txtSmsContent.Text,
```



```

        DateTime.Now.ToString());
    if (BaseClass.GSM.GSMModemSMSsend(null, 8, content,
        Encoding.Default.GetByteCount(content), lvInceptNum.Items[i].Text,
        false) == true)
        k++;
    }
    if (k == j)
    {
        MessageBox.Show("短信发送成功!", "提示", MessageBoxButtons.OK);
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
private void txtaddnum_KeyPress(object sender, KeyPressEventArgs e)
{
    if ((e.KeyChar != 8 && !char.IsDigit(e.KeyChar)) && e.KeyChar != 13)
    {
        MessageBox.Show("请输入数字");
        e.Handled = true;
    }
}
}

```

(3) 当单击“立即发送短消息”按钮后，将短信内容发送出去。对应的代码如下所示：

```

private void btnSend_Click(object sender, EventArgs e)
{
    if (lvInceptNum.Items.Count > 0)
    {
        if (ckbselect.Checked)
        {
            content = "老朋友: " + txtSmsContent.Text;
        }
        else
        {
            content = txtSmsContent.Text;
        }
        if (this.lvInceptNum.Items.Count <= 0)
        {
            MessageBox.Show("手机号码不能为空!", "提示", MessageBoxButtons.OK);
            this.txtaddnum.Focus();
            return;
        }
        if (txtSmsContent.Text == "")
        {
            MessageBox.Show("短信内容不能为空!", "提示", MessageBoxButtons.OK);
            txtSmsContent.Focus();
            return;
        }
        SendSms(JQCOM, BTL);
    }
    else
    {
        MessageBox.Show("请输入接收的电话号码");
    }
}
}

```

(4) 分别定义单击“取消发送短消息”按钮、“插入短语”按钮、“添加接收者”按

钮、“删除接收者”按钮、“电话簿”按钮的处理事件。对应的代码如下所示：

```
private void btnCancel_Click(object sender, EventArgs e)
{
    this.Close();
}

private void btnInsert_Click(object sender, EventArgs e)
{
    Dy frmd = new Dy();
    frmd.Owner = this;
    frmd.ShowDialog();
}

private void btnIncept_Click(object sender, EventArgs e)
{
    if (txtaddnum.Text == "")
    {
        MessageBox.Show("电话号码不能为空!");
    }
    else
    {
        if (txtaddnum.Text.Length < 11)
        {
            MessageBox.Show("手机号码为11位");
            return;
        }
        if (lvInceptNum.Items.Count > 0)
        {
            for (int i=0; i<lvInceptNum.Items.Count; i++)
            {
                if (lvInceptNum.Items[i].Text == txtaddnum.Text)
                {
                    MessageBox.Show("号码已经添加过了!");
                    return;
                }
            }
            lvInceptNum.Items.Add(txtaddnum.Text);
            txtaddnum.Text = "";
        }
        else
        {
            lvInceptNum.Items.Add(txtaddnum.Text);
            txtaddnum.Text = "";
        }
    }
}

private void btnDelIncept_Click(object sender, EventArgs e)
{
    if (lvInceptNum.Items.Count > 0)
    {
        foreach (ListViewItem it in lvInceptNum.SelectedItems)
        {
            lvInceptNum.Items.Remove(it);
        }
    }
    else
    {
        MessageBox.Show("没有电话号码");
    }
}
```

```

    }
    private void btnSelTel_Click(object sender, EventArgs e)
    {
        TelNote telnote = new TelNote();
        telnote.Owner = this;
        telnote.ShowDialog();
    }
    private void lvInceptNum_SelectedIndexChanged(object sender, EventArgs e)
    {
    }
    private void txtSmsContent_TextChanged(object sender, EventArgs e)
    {
    }
}

```

### 10.6.4 短信接收模块

短信接收模块的功能是接收短信信息，要实现此功能，首先应当具备短信的接收和发送功能。当发送短信后，对方可能回复短信，所以要具备短信接收功能。

此短信接收模块是通过动态链接库 `dllforvc.dll` 中的 `GSMModemInitNew()` 方法读取 SIM 卡中的短信信息的，然后将信息插入到数据库中，并通过数据控件显示出来。设计后的窗体效果如图 10-14 所示。

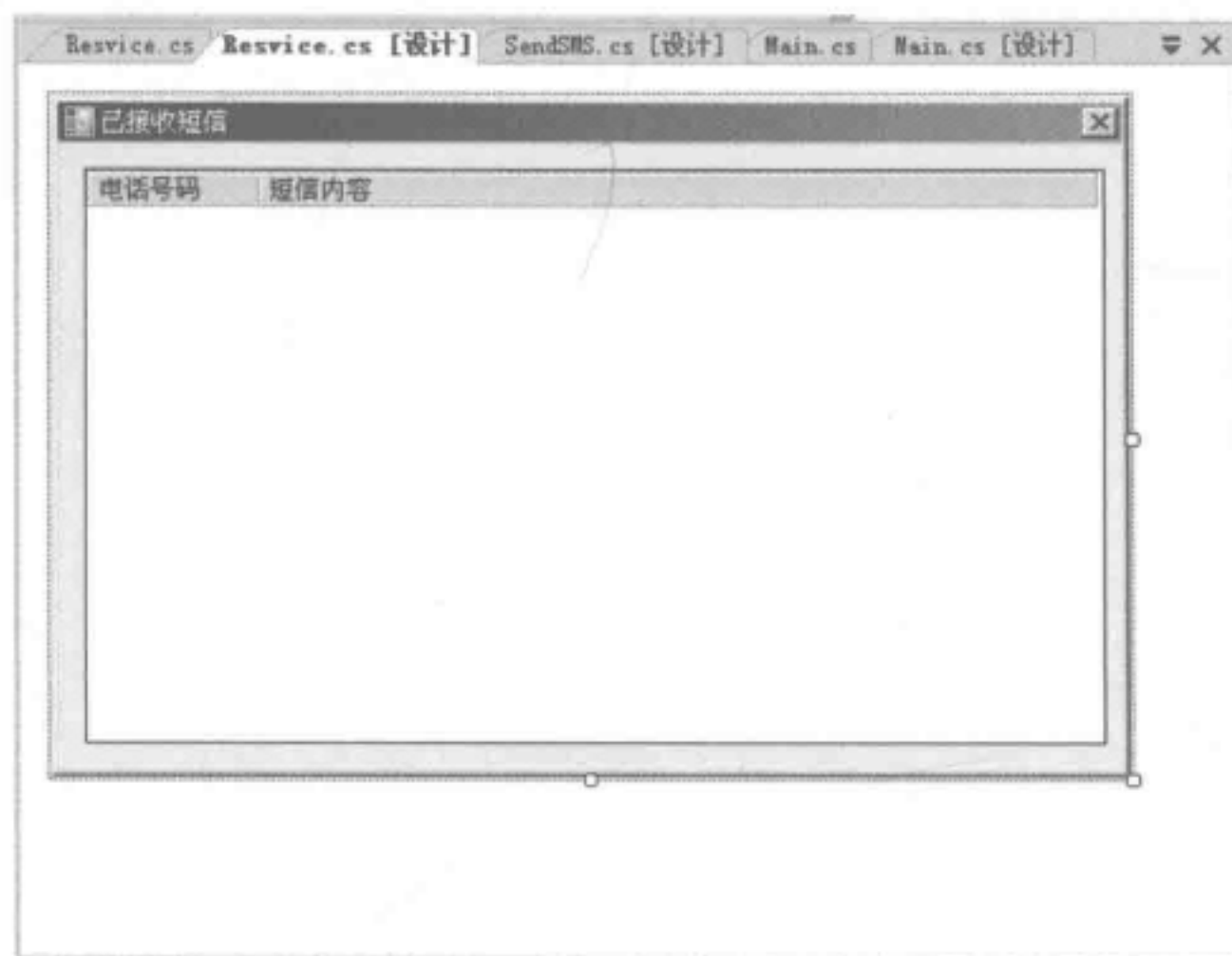


图 10-14 短信接收窗体

#### 1. 文件 Resvice.cs

本模块的实现文件是 `Resvice.cs`，下面讲解其具体的实现流程。

(1) 引入命名空间 `System.Data.OleDb`，定义公共变量 `jcom`、`btl` 和 `power`。对应的代码如下所示：

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;

```



```
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
namespace SMS
{
    public partial class Resvice : Form
    {
        public Resvice()
        {
            InitializeComponent();
        }
        public static string jcom;
        public static string btl;
        public string power = "YIWU-IJDD-CDQW-JDWG";
    }
}
```

(2) 开始加载窗体，首先连接短信猫，并检查 SIM 卡中是否有短信信息，如果有，则清空数据库并将短信添加到数据库中，最后将数据库中的短信显示出来。对应的代码如下所示：

```
private void frmResvice_Load(object sender, EventArgs e)
{
    jcom=BaseClass.GSM.GSMModemGetDevice(); //COM1
    btl = BaseClass.GSM.GSMModemGetBaudrate(); //波特率
    //连接设备
    if (BaseClass.GSM.GSMModemInitNew(jcom, btl, null, null, false, power) == false)
    {
        MessageBox.Show("设备连接失败!" + BaseClass.GSM.GSMModemGetErrorMsg(),
            "提示", MessageBoxButtons.OK);
        return;
    }
    string smscontent = BaseClass.GSM.GSMModemSMSReadAll(1);
    if (smscontent==null)
    {
        MessageBox.Show("暂时没有新的信息!", "系统提示", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        return;
    }
    else
    {
        DelData();
        string newstr = smscontent.Replace("||", "#");
        string[] scontent = newstr.Split('#');
        string smscon = "";
        for (int i=0; i<scontent.Length; i++)
        {
            smscon = scontent[i];
            string[] a = smscon.Split('|');
            string b = "";
            b = a[0].ToString();
            if (b.Length<11 && smscon!="")
            {
                string smsstr = b;
                string smscot = scontent[i].Substring(
                    b.Length, scontent[i].Length - b.Length).Replace("|", "");
                AddData(smsstr, smscot);
            }
            else
            {
                if (smscon != "")
                {

```

```

        if (scontent[i].Substring(0, 1) == "|")
        {
            string smsstr = scontent[i].Substring(3,
                scontent[i].Length - 3).Substring(0, 11);
            string smscot = scontent[i].Substring(14,
                scontent[i].Length - 14).Replace("|", "");
            AddData(smsstr, smscot);
        }
        else
        {
            string smsstr = scontent[i].Substring(2,
                scontent[i].Length - 2).Substring(0, 11);
            string smscot = scontent[i].Substring(13,
                scontent[i].Length - 13).Replace("|", "");
            AddData(smsstr, smscot);
        }
    }
    //MessageBox.Show(smscon);
}
GetData();
}
}

```

(3) 定义如下 3 个方法。

- AddData(): 将短信添加到数据库。
- DelData(): 清空数据库。
- GetData(): 获取数据库中的短信信息, 并将信息绑定到 dataGridView 控件。

对应的代码如下所示:

```

private void AddData(string num, string content)
{
    OleDbConnection conn = BaseClass.ConnClass.DataConn();
    conn.Open();
    OleDbCommand cmd = new OleDbCommand(
        "insert into tb_Resvice([smsnum],[smscontent]) values('"
        + num + "', '" + content + "')" , conn);
    cmd.ExecuteNonQuery();
    conn.Close();
}
private void DelData()
{
    OleDbConnection conn = BaseClass.ConnClass.DataConn();
    conn.Open();
    OleDbCommand cmd = new OleDbCommand("delete from tb_Resvice", conn);
    cmd.ExecuteNonQuery();
    conn.Close();
}
private void GetData()
{
    OleDbConnection conn = BaseClass.ConnClass.DataConn();
    OleDbDataAdapter sda =
        new OleDbDataAdapter("select * from tb_Resvice order by ID DESC", conn);
    DataSet ds = new DataSet();
    sda.Fill(ds);
    dgvResvice.DataSource = ds.Tables[0];
    conn.Close();
}

```

(4) 双击某条短信后，将弹出窗体，显示此短信的详细信息。

对应的实现代码如下所示：

```
private void dataGridView1_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    string id = dgvResvice.SelectedCells[0].Value.ToString();
    string snum = dgvResvice.SelectedCells[1].Value.ToString();
    string stxt = dgvResvice.SelectedCells[2].Value.ToString();

    SmsXX smsxx = new SmsXX();
    smsxx.Did = id;
    smsxx.Dnum = snum;
    smsxx.Dcontent = stxt;
    smsxx.Show();
}
}
```

## 2. 文件 SmsXX.cs

当弹出新窗体后，即可以浏览此短信的信息，也可以对此短信进行回复，回复成功后，将回复内容添加到数据库中去。上述功能是通过文件 SmsXX.cs 实现的，下面来讲解其实现流程。

(1) 引入命名空间 System.Data.OleDb，并定义全局变量。对应的代码如下所示：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
namespace SMS
{
    public partial class SmsXX : Form
    {
        public SmsXX()
        {
            InitializeComponent();
        }
        public string content;
        public string ResviceNum;
        public static string NUM;
        public static string JQCOM;
        public static string BTL;
        public string Power = "YIWU-IJDD-CDQW-JDWG";
        public static int k = 0;
        public static int j = 0;
        public string Did;
        public string Dnum;
        public string Dcontent;
        private void frmSmsXX_Load(object sender, EventArgs e)
        {
            txtFrom.Text = Dnum;
            txtContent.Text = Dcontent;
            NUM = BaseClass.GSM.GSMModemGetSnInfoNew(JQCOM, BTL); //机器号码
        }
    }
}
```



```
JQCOM = BaseClass.GSM.GSMModemGetDevice(); //COM1
BTL = BaseClass.GSM.GSMModemGetBaudrate(); //波特率
}
```

(2) 定义方法 AddMessage 将短信的内容添加到数据库中，对应的代码如下所示：

```
private void AddMessage(string num, string strcontent, string sendtime)
{
    OleDbConnection conn = BaseClass.ConnClass.DataConn();
    conn.Open();
    OleDbCommand cmd = new OleDbCommand(
        "insert into tb_TelSend([TelNum],[TelContent],[TelTime]) values('"
        + num + "','" + strcontent + "','" + sendtime + "')", conn);
    cmd.ExecuteNonQuery();
    conn.Close();
}
```

(3) 定义方法 SendSms，实现短信回复，回复成功后，会显示成功提示，对应的代码如下所示：

```
private void SendSms(string strcom, string strbtl)
{
    ////连接设备
    if (BaseClass.GSM.GSMModemInitNew(strcom, strbtl, null, null, false, Power)
        == false)
    {
        MessageBox.Show("设备连接失败!" + BaseClass.GSM.GSMModemGetErrorMsg(),
            "提示", MessageBoxButtons.OK);
        return;
    }
    // 发送短信
    AddMessage(txtFrom.Text, txtSmsContent.Text, DateTime.Now.ToString());
    if (BaseClass.GSM.GSMModemSMSsend(null, 8, content,
        Encoding.Default.GetByteCount(content), txtFrom.Text, false) == true)
        k++;
    MessageBox.Show("回复短信成功!", "提示", MessageBoxButtons.OK);
}
```

(4) 浏览短信内容后，如果要对此短信进行回复，可以填好短信内容，并单击“回复”按钮进行回复，对应的代码如下所示：

```
private void btnReply_Click(object sender, EventArgs e)
{
    if (txtSmsContent.Text == "")
    {
        MessageBox.Show("请输入短信内容");
    }
    else
    {
        content = txtSmsContent.Text;
        SendSms(JQCOM, BTL);
        this.Close();
    }
}
private void btnClose_Click(object sender, EventArgs e)
{
    this.Close();
}
}
```

## 10.6.5 电话簿管理模块

此模块的功能是管理系统内的联系人信息，主要包括对联系人号码的添加和删除。

### 1. 添加联系人

设计后的“添加联系人”窗体效果如图 10-15 所示。

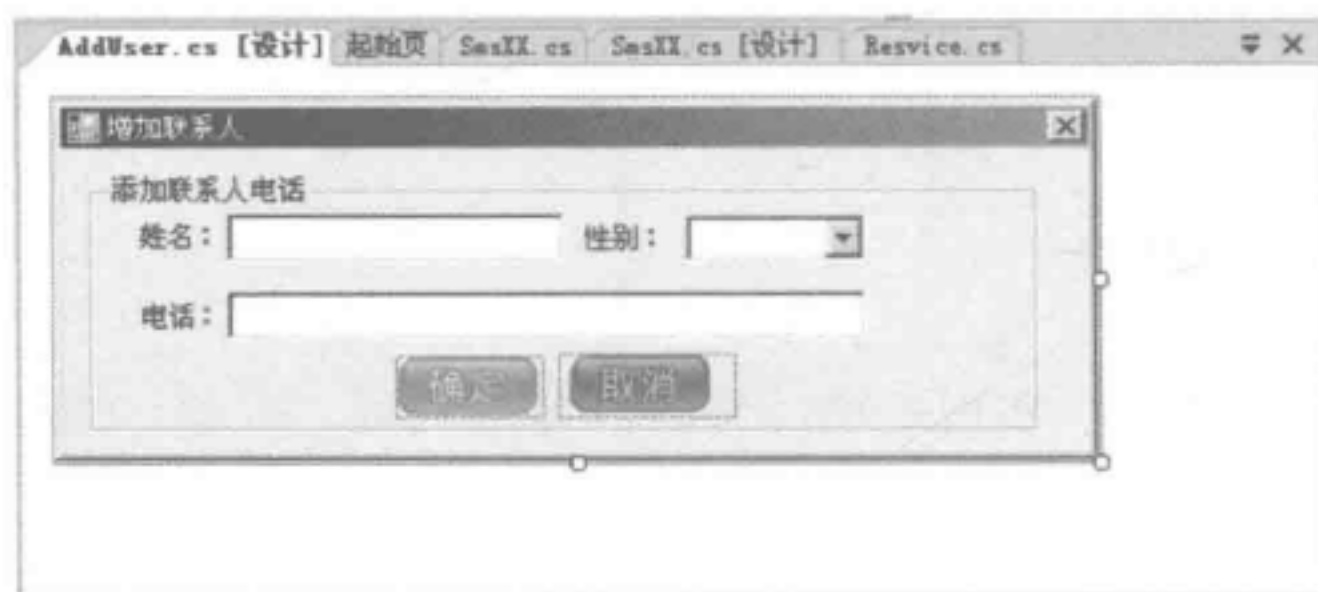


图 10-15 “添加联系人”窗体

添加联系人功能的实现文件是 AddUser.cs，用来向系统内添加新的联系人。通过定义的方法确保输入的号码是数字，并保证输入联系人信息。具体的实现代码如下所示：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
namespace SMS
{
    public partial class AddUser : Form
    {
        public AddUser()
        {
            InitializeComponent();
        }

        private void frmAddUser_Load(object sender, EventArgs e)
        {
            cbbSex.SelectedIndex = 0;
        }

        private void button2_Click(object sender, EventArgs e)
        {
        }

        private void txtNum_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((e.KeyChar!=8 && !char.IsDigit(e.KeyChar)) && e.KeyChar!=13)
            {
                MessageBox.Show("请输入数字");
                e.Handled = true;
            }
        }
    }
}
```

```

}
private void pbSubmit_Click(object sender, EventArgs e)
{
    if (txtName.Text == "")
    {
        MessageBox.Show("请输入姓名");
    }
    else
    {
        if (txtNum.Text.Length < 11)
        {
            MessageBox.Show("电话号码为 11 位");
        }
        else
        {
            OleDbConnection conn = BaseClass.ConnClass.DataConn();
            conn.Open();
            OleDbCommand cmd = new OleDbCommand(
                "insert into tb_tel([UserName],[UserSex],[UserTel]) values('"
                + txtName.Text + "','" + cbbSex.SelectedItem.ToString() + "','"
                + txtNum.Text + "')", conn);
            cmd.ExecuteNonQuery();
            MessageBox.Show("增加联系人成功");
            txtName.Text = "";
            txtNum.Text = "";
            conn.Close();
        }
    }
}

private void pbConcel_Click(object sender, EventArgs e)
{
    txtName.Text = "";
    txtNum.Text = "";
}
}

```

## 2. 显示、删除联系人

新建一个窗体，用于显示系统内的联系人信息，并实现对某联系人的删除操作。设计后的窗体效果如图 10-16 所示。



图 10-16 显示、删除联系人窗体



此功能的实现文件是 Tel.cs，当加载窗体时，显示数据库内的所有联系人信息；然后定义 pbDelete\_Click 事件程序，删除所选的联系人信息。具体实现代码如下所示：

```
using System.Data.OleDb;
namespace SMS
{
    public partial class Tel : Form
    {
        public Tel()
        {
            InitializeComponent();
        }
        //加载窗体时显示数据库内的所有联系人信息
        private void frmTel_Load(object sender, EventArgs e)
        {
            SetTelData();
        }
        private void SetTelData()
        {
            OleDbConnection conn = BaseClass.ConnClass.DataConn();
            OleDbDataAdapter da =
                new OleDbDataAdapter("select * from tb_tel order by ID desc", conn);
            DataSet ds = new DataSet();
            da.Fill(ds);
            dgvAllUser.DataSource = ds.Tables[0];
        }
        private void pbDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("确定删除吗?", "警告", MessageBoxButtons.OKCancel,
                MessageBoxIcon.Warning) == DialogResult.OK)
            {
                string id = dgvAllUser.SelectedCells[0].Value.ToString();
                OleDbConnection conn = BaseClass.ConnClass.DataConn();
                conn.Open();
                OleDbCommand cmd = new OleDbCommand("delete from tb_tel where ID=" + id, conn);
                cmd.ExecuteNonQuery();
                conn.Close();
                MessageBox.Show("删除联系人成功");
                SetTelData();
            }
        }
        private void pbClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

在使用 Visual C# 删除记录时，必须从两个方面彻底删除记录，即从数据库和用 Visual C# 编程时产生的一个 DataSet 对象中彻底删除。在进行程序设计工作的时候，如果只是删除了 DataSet 对象中的记录信息，这种删除是一种伪删除，这是因为当退出程序，又重新运行程序后，会发现，那个要删除的记录依然还存在。这是因为 DataSet 对象只是对数据表的一个镜像，并不是真正的记录本身。而如果只是从数据库中删除记录，因为此时程序用到的数据集是从 DataSet 对象中读取的，子 DataSet 对象中依然保存此条记录的镜像，所以会发现根本没有删除掉记录，但实际上已经删除了；此时，只有退出程序，重新运行后才会发现记录已经删除了。经常使用的方法是删除以上两个方面的记录或记录镜像信息。

当然，也可以使用其他的方法，例如，首先从数据库中删除记录，然后重新建立数据连接，重新创建一个新的 DataSet 对象。这种方法虽然也可以达到相同的目的，但显然相对繁杂些，所以本章项目采用的是第一种方法：直接删除。

## 10.6.6 常用短语管理模块

此模块的功能是管理系统内的常用短语信息。为了便于发送短信息，系统内置了一些常用的短语，在编写短信时，可以直接调用这些短语。

### 1. 常用短语管理

创建一个名为 Cydy.cs 的窗体，用于显示不同类型的常用短语，并对这些短语实现管理操作。设计后的窗体效果如图 10-17 所示。

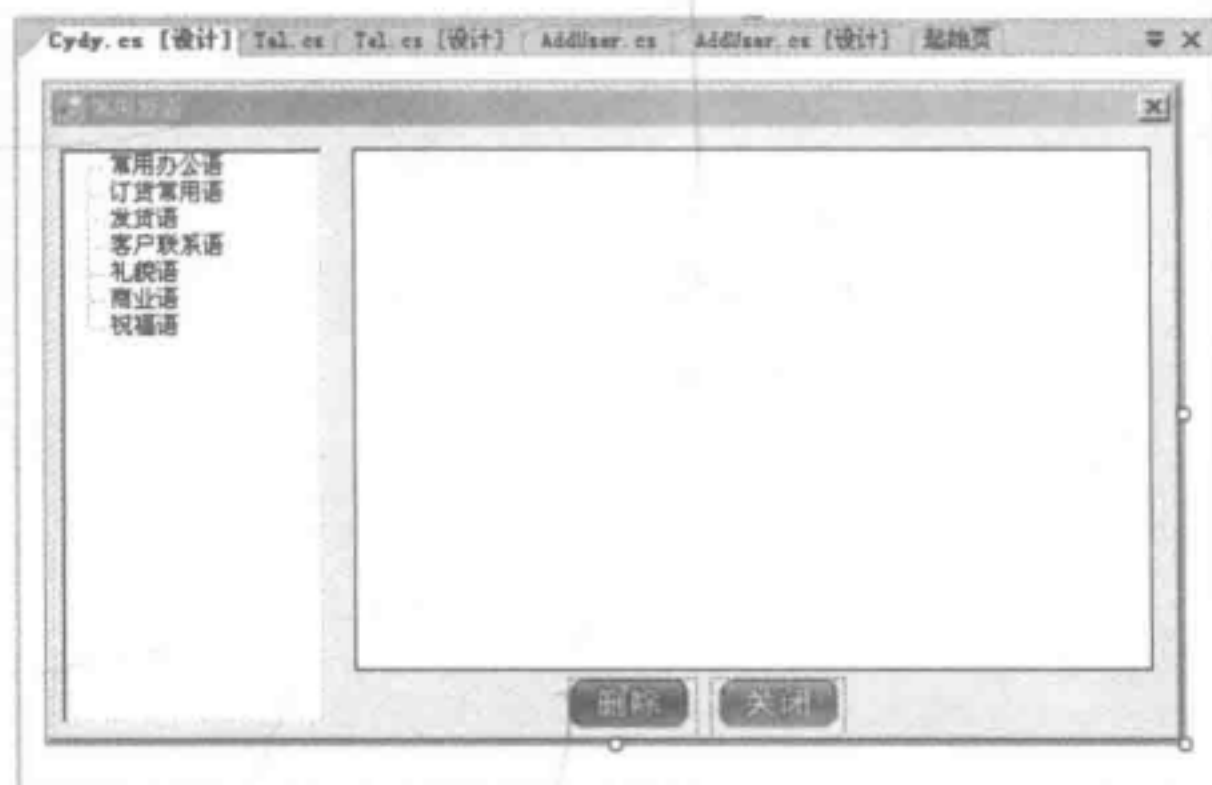


图 10-17 常用短语管理窗体

此功能的实现文件是 Cydy.cs，具体实现代码如下所示：

```
using System.Data.OleDb;
namespace SMS
{
    public partial class Cydy : Form
    {
        public Cydy()
        {
            InitializeComponent();
        }
        private string str;
        private void frmCydy_Load(object sender, EventArgs e) { }
        public void BindData()
        {
            string str1 = "select * from tb_note where type='" + str + "'";
            OleDbConnection conn = BaseClass.ConnClass.DataConn();
            OleDbDataAdapter oda = new OleDbDataAdapter(str1, conn);
            DataSet ds = new DataSet();
            oda.Fill(ds);
            dgvShow.DataSource = ds.Tables[0];
        }
        private void pbClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

```

private void pbDelete_Click(object sender, EventArgs e)
{
    string id = dgvShow.SelectedCells[0].Value.ToString();
    OleDbConnection conn = BaseClass.ConnClass.DataConn();
    conn.Open();
    OleDbCommand cmd = new OleDbCommand("delete from tb_note where ID=" + id, conn);
    cmd.ExecuteNonQuery();
    conn.Close();
    MessageBox.Show("删除成功");
    BindData();
}
private void dgvShow_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    string id = this.dgvShow.SelectedCells[0].Value.ToString();
    string content = this.dgvShow.SelectedCells[2].Value.ToString();
    string type = tvDy.SelectedNode.Text;
    DyChange dychange = new DyChange();
    dychange.Scontent = content;
    dychange.Sid = id;
    dychange.Stype = type;
    dychange.ShowDialog();
}
private void frmCydy_Activated(object sender, EventArgs e)
{
    BindData();
}
private void tvDy_AfterSelect(object sender, TreeViewEventArgs e)
{
    str = tvDy.SelectedNode.Text;
    BindData();
}
}
}

```

## 2. 常用短语详情

创建一个名为 DyChange.cs 的窗体，用于显示某条常用短语的详情信息。设计后的窗体效果如图 10-18 所示。

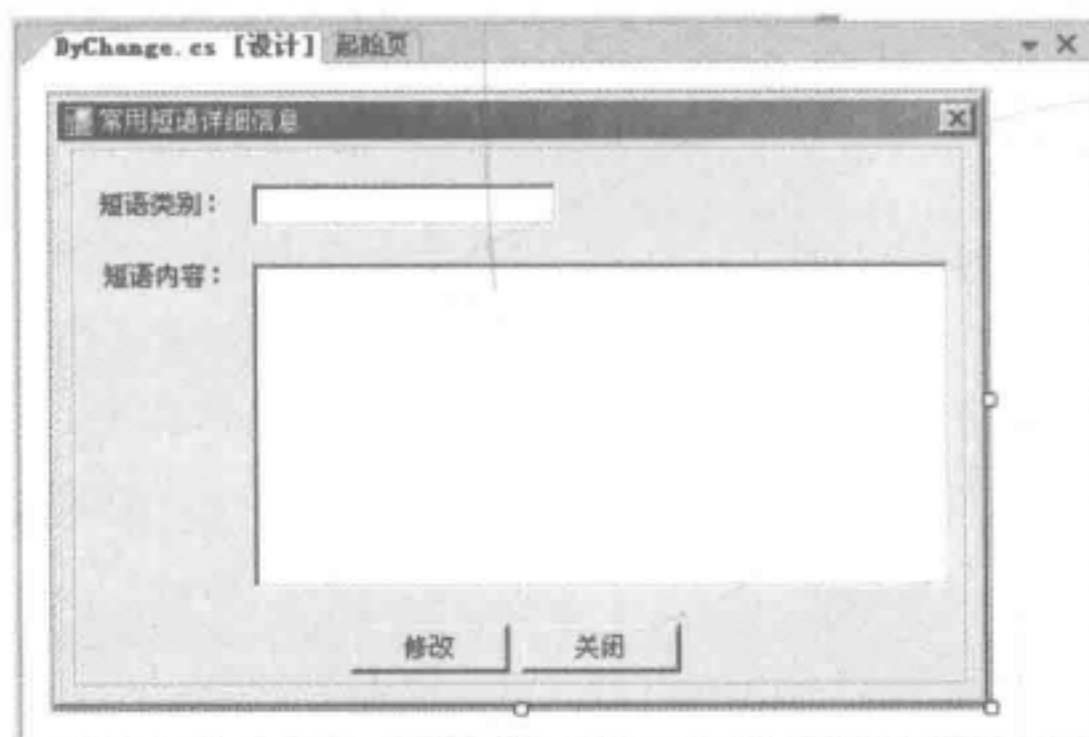


图 10-18 常用短语详情窗体

此功能的实现文件是 DyChange.cs，具体实现代码如下所示：

```

using System.Data.OleDb;
namespace SMS
{

```



```
public partial class DyChange : Form
{
    public DyChange()
    {
        InitializeComponent();
    }
    public string Stype;
    public string Scontent;
    public string Sid;
    private void frmDyChange_Load(object sender, EventArgs e)
    {
        txttype.Text = Stype;
        txtContent.Text = Scontent;
    }
    private void btnSure_Click(object sender, EventArgs e)
    {
        if (txtContent.Text == "")
        {
            MessageBox.Show("请输入常用短语");
        }
        else
        {
            OleDbConnection conn = BaseClass.ConnClass.DataConn();
            conn.Open();
            OleDbCommand cmd = new OleDbCommand(
                "update tb_note set [note]='"+txtContent.Text+" where ID="+Sid, conn);
            cmd.ExecuteNonQuery();
            conn.Close();
            this.Hide();
        }
    }
    private void btnClose_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
}
```

### 3. 添加常用短语

创建一个名为 AddDy.cs 的窗体，用于向系统内添加新的常用短语信息。设计后的窗体效果如图 10-19 所示。

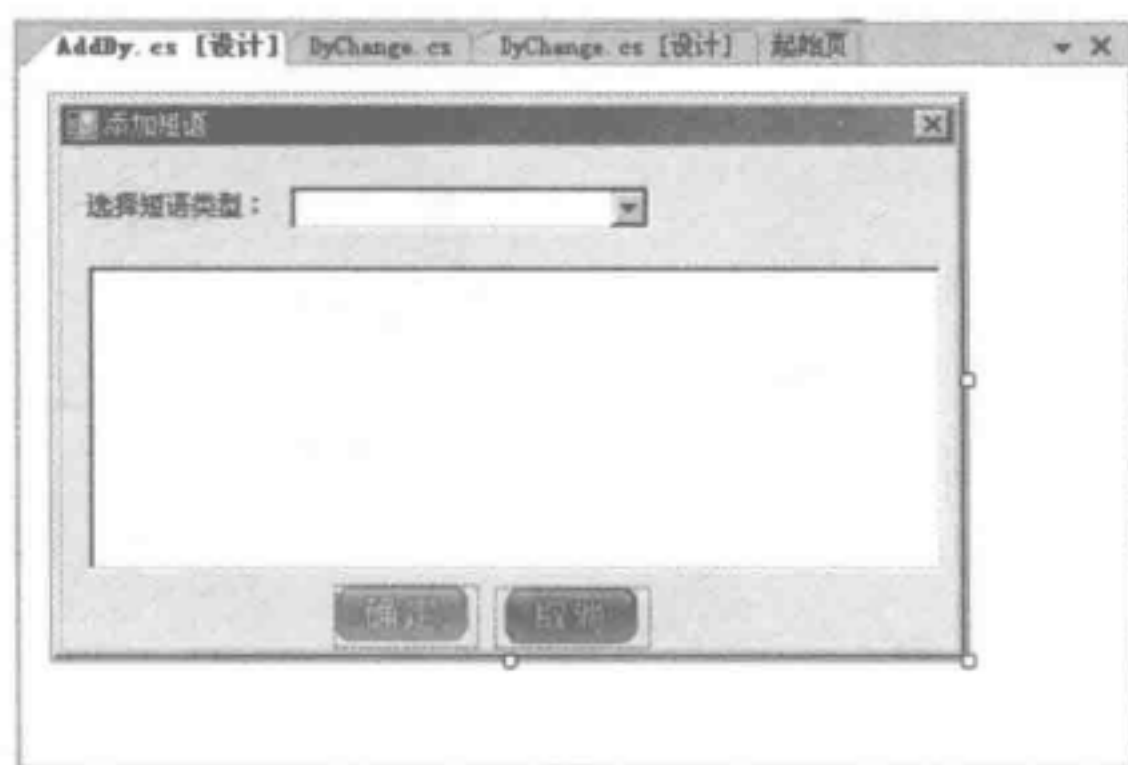


图 10-19 添加常用短语窗体

此功能的实现文件是 AddDy.cs, 具体实现代码如下所示:

```
using System.Data.OleDb;

namespace SMS
{
    public partial class AddDy : Form
    {
        public AddDy()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
        }

        private void frmAddDy_Load(object sender, EventArgs e)
        {
            cbbType.SelectedIndex = 0;
        }

        private void button1_Click(object sender, EventArgs e)
        {
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            string strtype = cbbType.SelectedItem.ToString();
            string strnote = txtDy.Text;
            OleDbConnection conn = BaseClass.ConnClass.DataConn();
            conn.Open();

            OleDbCommand cmd = new OleDbCommand(
                "insert into tb_note([type],[note]) values('" +
                strtype + "','" + strnote + "')" , conn);

            cmd.ExecuteNonQuery();
            txtDy.Text = "";
            MessageBox.Show("添加成功");

            conn.Close();
        }

        private void pictureBox2_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

### 10.6.7 修改密码模块

此模块的功能, 是修改系统内管理员用户的密码, 设计后的窗体界面的效果如图 10-20 所示。

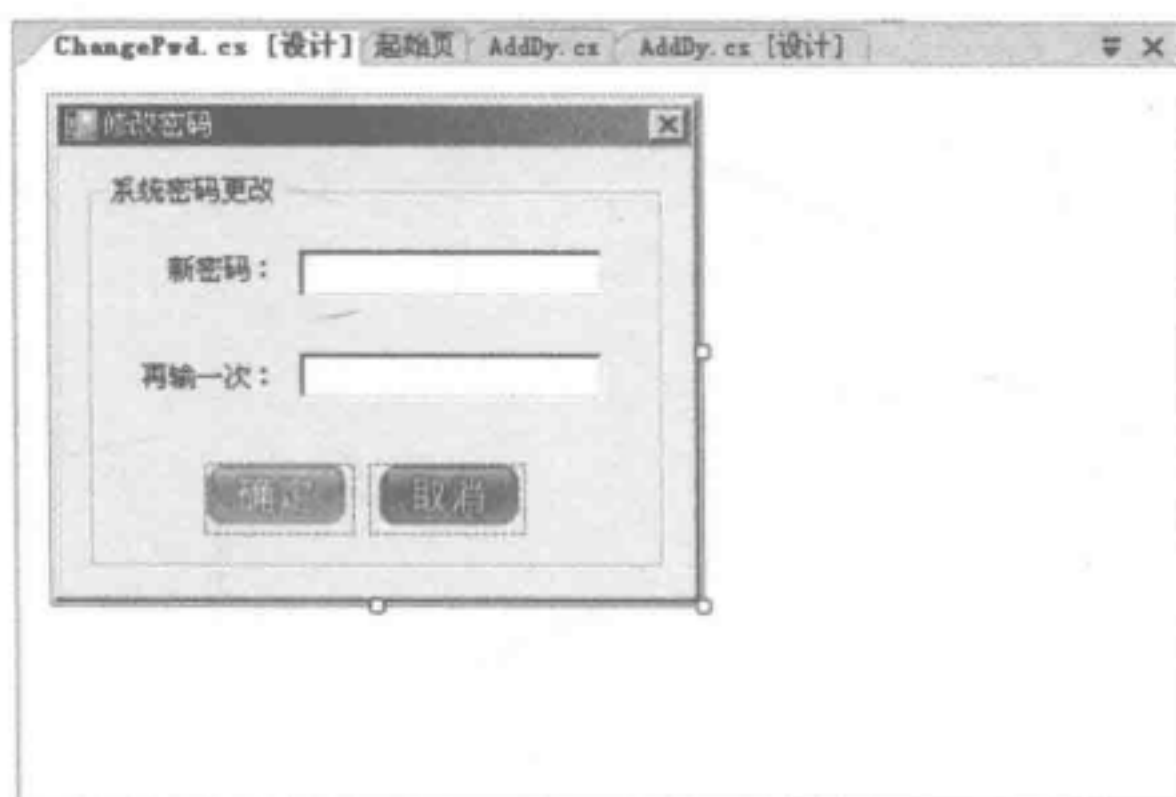


图 10-20 “修改密码”窗体

此模块功能是由文件 ChangePwd.cs 实现的，具体实现代码如下所示：

```
using System.Data.OleDb;
namespace SMS
{
    public partial class ChangePwd : Form
    {
        public ChangePwd()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
        }
        private void button2_Click(object sender, EventArgs e)
        {
        }
        private void pictureBox1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text == "")
            {
                MessageBox.Show("密码不能为空");
            }
            else
            {
                if (textBox2.Text == "" || textBox2.Text != textBox1.Text)
                {
                    MessageBox.Show("两次密码不一致");
                }
                else
                {
                    OleDbConnection conn = BaseClass.ConnClass.DataConn();
                    conn.Open();
                    OleDbCommand cmd = new OleDbCommand(
                        "update tb_Admin set [AdminUserPwd]='" + textBox1.Text + "'", conn);
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    if (MessageBox.Show("密码修改成功", "提示", MessageBoxButtons.OK,
                        MessageBoxIcon.Warning) == DialogResult.OK)
                    {
                        this.Close();
                    }
                }
            }
        }
    }
}
```



```
    }  
    }  
    private void pictureBox2_Click(object sender, EventArgs e)  
    {  
        this.Close();  
    }  
    private void frmChangePwd_Load(object sender, EventArgs e)  
    {  
    }  
    }  
}
```

到此为止，整个项目的编码工作已经结束。作为一个短信群发项目，选购短信猫是一项必然的工作。短信猫产品现在越来越广泛地使用。随着垃圾短信的增多，市场中的负面反映比较多，但短信猫仍然在深入使用。现在的短信猫已经从单纯的短信群发，扩展到短信索函。

如何鉴别短信猫的优劣和正确选购短信猫，成了短信猫购买客户的一个难题。我们推荐简单地从短信猫硬件和软件两方面同时入手，这样就可以找到需要的短信猫了。

在短信猫的开发包方面，一般的小公司或厂家是没有能力提供开发包的，他们会用较低的价格先把自己的劣质短信猫卖给客户，然后把从互联网上收集到的一些简单的 AT 指令交给消费者，让客户根据 AT 指令自己开发。我们都知道，AT 指令基本是低级语言，与短信猫客户要想通过 ASP、Java、Dephi 等高级技术使用短信功能不相适应，因为通过 AT 指令加载短信功能，需要丰富的短信猫硬件内核知识和比较长的开发时间，甚至会造成最后短信开发工程失败的情况。

所以，假如短信猫的生产公司能够提供跨平台(Windows/Linux)的各种语言的短信猫开发包，而且里面有丰富的开发实例，对我们来说是很省心的，这样的公司才是可信的。

## 10.7 项目调试

视频讲解 光盘：视频\第 10 章\项目调试.avi

在 Visual Studio 2013 中将此项目命名为“SMS”，开始调试运行，用户登录界面的效果如图 10-21 所示。

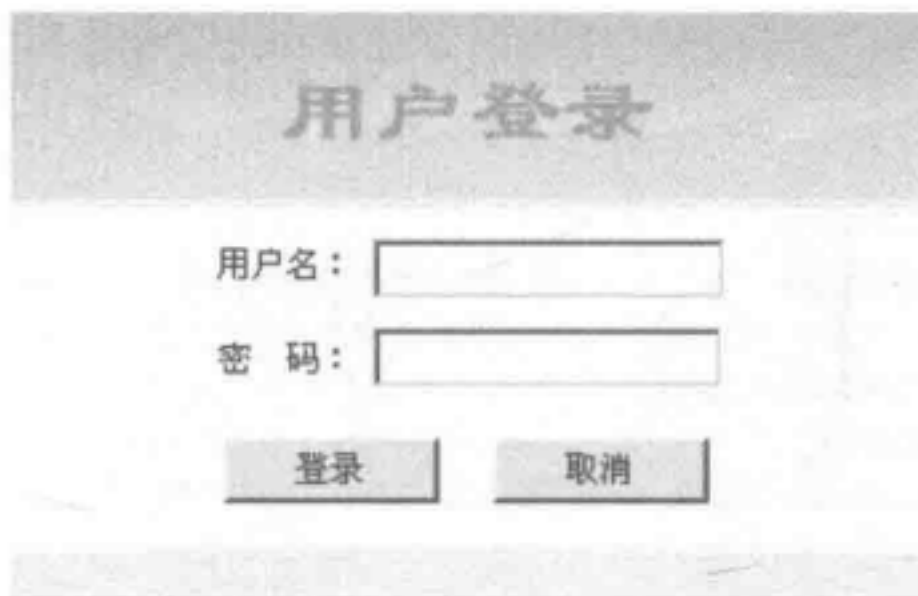


图 10-21 用户登录界面的效果

系统默认主界面的效果如图 10-22 所示。

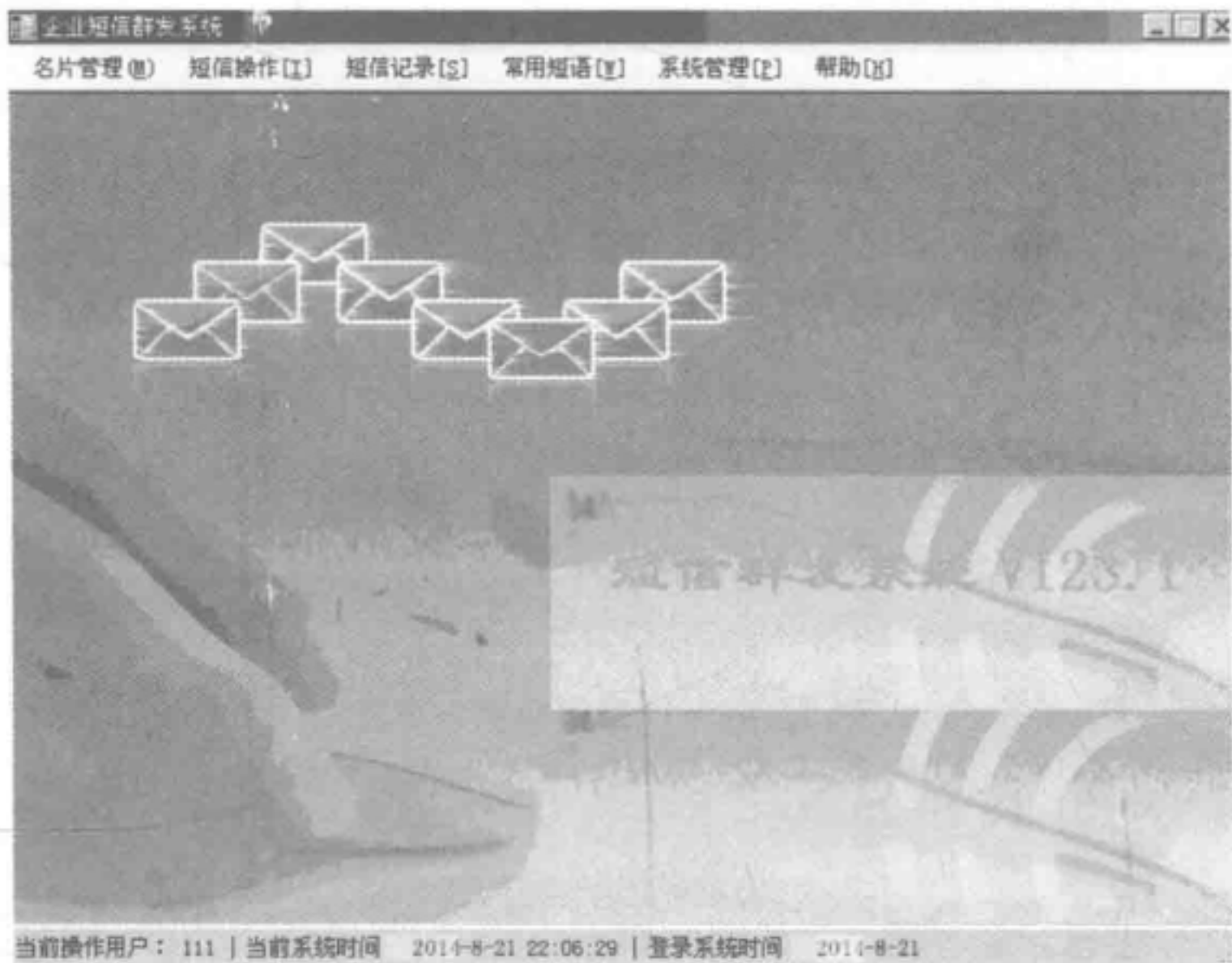


图 10-22 系统默认主界面的效果

“增加联系人”界面如图 10-23 所示。

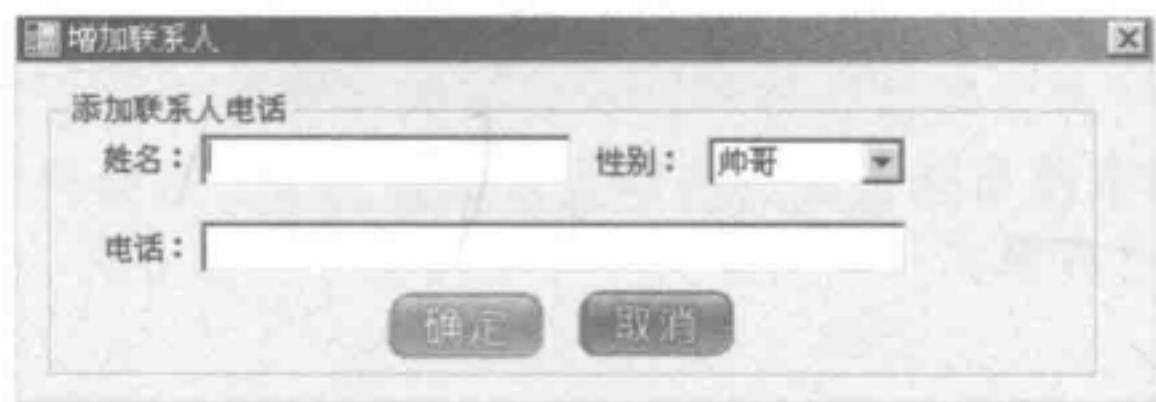


图 10-23 “增加联系人”界面

“已发信息”界面如图 10-24 所示。

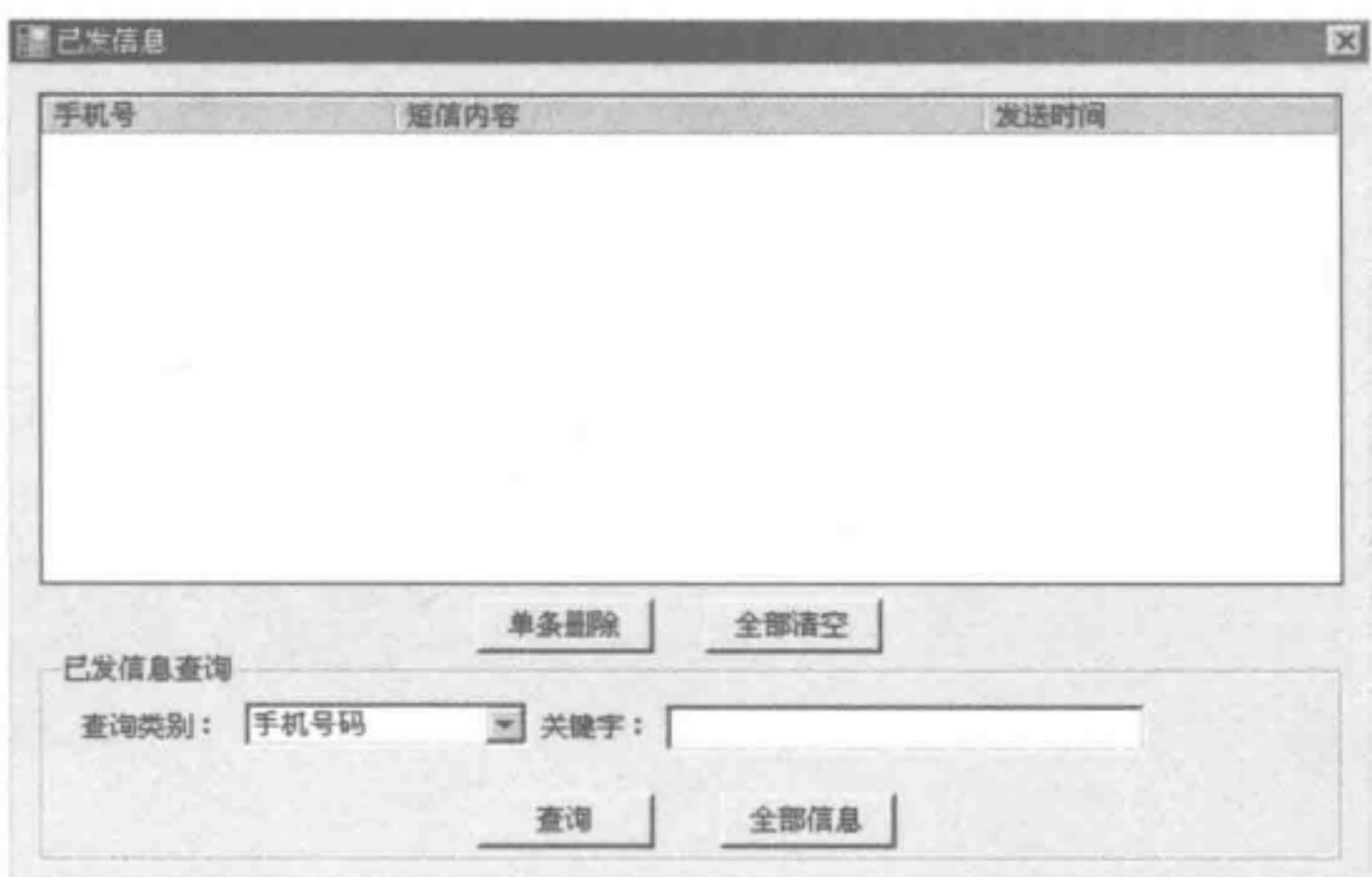


图 10-24 “已发信息”界面

“常用短语”界面如图 10-25 所示。



图 10-25 “常用短语”界面

“修改密码”界面如图 10-26 所示。

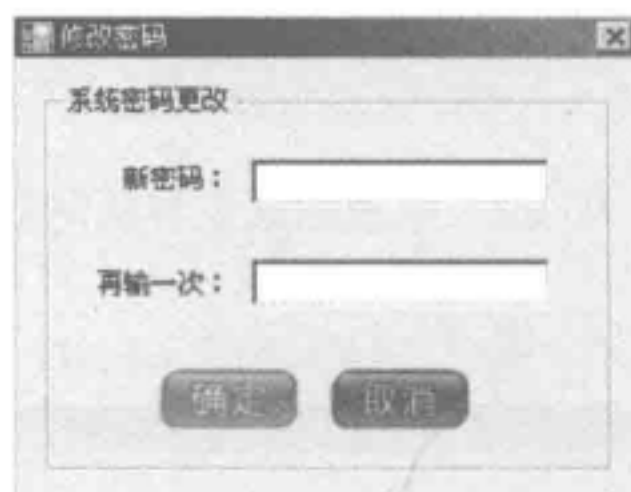


图 10-26 “修改密码”界面



## 第 11 章 超市进销存系统

高帆项目开发

超市用户对进销存系统的需求具有普遍性。超市进销存销售管理系统适用于超市采购、销售和仓库部门，对超市采购、销售及仓库的业务全过程进行有效控制和跟踪。使用超市进销存销售管理系统可有效减少盲目采购、降低采购成本、合理控制库存、减少资金占用并提高市场灵敏度，提升超市的市场竞争力。

在本章的内容中，将介绍 Visual Studio + SQL Server 实现典型进销存系统的具体流程。



### 赠送的超值电子书

- 101 自定义集合类
- 102 迭代器
- 103 C#中的集合可以扩展
- 104 Queue 和 ArrayList 的对比
- 105 CreateInstance 输出某个数组内指定长度的数据
- 106 用形参数组替换重复的实参
- 107 使用 ArrayList 的最佳建议
- 108 类的层次结构
- 109 声明继承
- 110 C#继承规则

## 11.1 算法是程序的灵魂

视频讲解 光盘：视频\第 11 章\算法是程序的灵魂.avi

程序的灵魂是算法，只有掌握了算法，才能轻松地驾驭程序。编程并不是按部就班的事情，正确的做法是选择一种算法去实现功能，这个算法是解决问题的有力武器，也是对一个项目“下手”的第一步。算法能够告诉我们在面对一个应用时，用什么思路去实现，有了这个思路后，编码工作只须遵循这个思路去实现即可。算法是一个程序的编码思路，是程序员解决问题的指路明灯。在本节的内容中，将与读者一起探讨算法的奥秘。

### 11.1.1 什么是算法

算法的英文名称是 Algorithm，算法应该具有如下 5 个要点。

- 有穷性：保证执行有限步骤之后结束。
  - 确切性：每一步骤都有确切的定义。
  - 输入：每个算法有零个或多个输入，以刻画运算对象的初始情况。
  - 输出：每个算法有一个或多个输出，显示对输入数据加工后的结果。没有输出的算法是毫无意义的。
  - 可行性：在原则上，算法能够精确地运行，进行有限次运算后即可完成一种运算。
- 为了理解什么是算法，先看一道有趣的智力题。

“烧水泡茶”有如下几道工序：①烧开水；②洗茶壶、茶杯；③拿茶叶；④泡茶。

烧开水，洗茶壶、茶杯，拿茶叶是泡茶的前提。

其中烧开水需要 15 分钟，洗茶壶需要 2 分钟，洗茶杯需要 1 分钟，拿茶叶需要 1 分钟，泡茶需要 1 分钟。

下面是两种“烧水泡茶”的方法。

**方法一：**

- (1) 烧水。
- (2) 水烧开后，洗刷茶具，拿茶叶。
- (3) 沏茶。

**方法二：**

- (1) 烧水。
- (2) 烧水过程中，洗刷茶具，拿茶叶。
- (3) 水烧开后沏茶。

**问题：**比较这两个方法有何不同，并分析哪个方法更优。

上述两个方法都能最终实现“烧水泡茶”的功能，每种方法的三个步骤就是一种“算法”，其中方法二由于具有并行的操作，可以节省时间，显然是更优的。算法是指在有限步骤内求解某一问题所使用的一组定义明确的规则；通俗地说，就是计算机解题的过程。在这个过程中，无论是形成解题思路还是编写程序，都是在实施某种算法。

### 11.1.2 赢在技术沉淀——计算机中的算法

我们在编写程序实现某个项目功能的时候，需要遵循一定的算法。算法的地位非常重要，号称是程序的“灵魂”。

计算机中，算法可分为如下两大类。

- 数值运算算法：求解数值。
- 非数值运算算法：事务管理领域。

(1) 假设有一个下面的运算：

$$1 \times 2 \times 3 \times 4 \times 5$$

为了计算该运算的结果，最普通的做法是按照如下步骤进行计算。

第1步：先计算  $1 \times 2$ ，得到结果 2。

第2步：将步骤1得到的乘积 2 乘以 3，计算得到结果 6。

第3步：将 6 再乘以 4，计算得到 24。

第4步：将 24 再乘以 5，计算得到 120。

最终计算结果是 240，上述第1步到第4步的计算过程就是一个算法。如果我们想用编程的方式来解决上述运算，通常会使用如下算法来实现。

第1步：假设定义  $t=1$ 。

第2步：使  $i=2$ 。

第3步：使  $t \times i$ ，乘积仍然放在变量  $t$  中，可表示为  $t \times i \rightarrow t$ 。

第4步：使  $i$  的值加 1，即  $i+1 \rightarrow i$ 。

第5步：如果  $i \leq 5$ ，则重新执行步骤3以及其后的步骤4和步骤5；否则，算法结束。

由此可见，上述算法就是数学中的  $n!$  公式。既然有公式了，在具体编码的时候，只须使用这个公式，就可以解决上述运算的问题。

(2) 再看下面的一个数学应用问题。

假设有 80 个学生，要求打印输出成绩在 60 分以上的学生。

在此，设用  $n$  来表示学生学号， $n_i$  表示第  $i$  个学生的学号； $cheng$  表示学生成绩， $cheng_i$  表示第  $i$  个学生的成绩。根据题目要求，我们可以写出如下算法。

第1步： $1 \rightarrow i$ 。

第2步：如果  $cheng_i \geq 60$ ，则打印输出  $n_i$  和  $cheng_i$ ，否则不打印输出。

第3步： $i+1 \rightarrow i$ 。

第4步：如果  $i \leq 80$ ，返回步骤2，否则结束。

由此可见，算法在计算机中的地位十分重要。所以在面对一个项目应用时，一定不要立即埋头苦干地编写代码，而是要仔细思考解决这个问题的算法是什么。想出算法之后，然后以这个算法为指导思想来编码。

算法的表示方法即算法的描述和外在表现，在计算机中，表示算法的方法如下。

#### 1. 用流程图来表示算法

流程图的标识说明如图 11-1 所示。



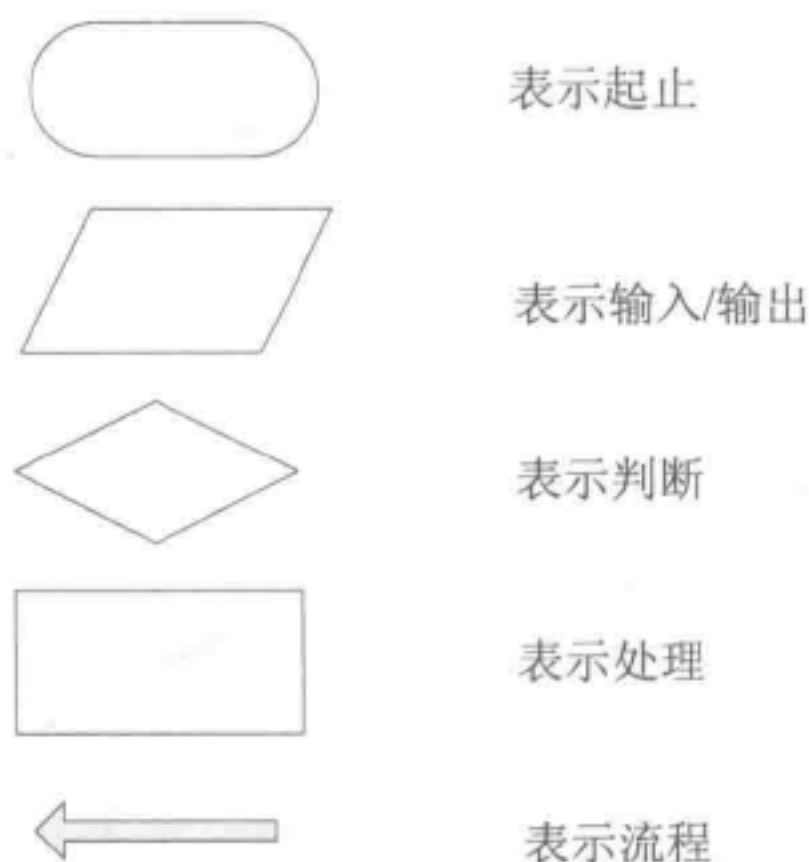


图 11-1 流程图的标识说明

回到前面打印输出成绩在 60 分以上的学生的的问题中。针对该问题，可以使用如图 11-2 所示的算法流程图来表示。

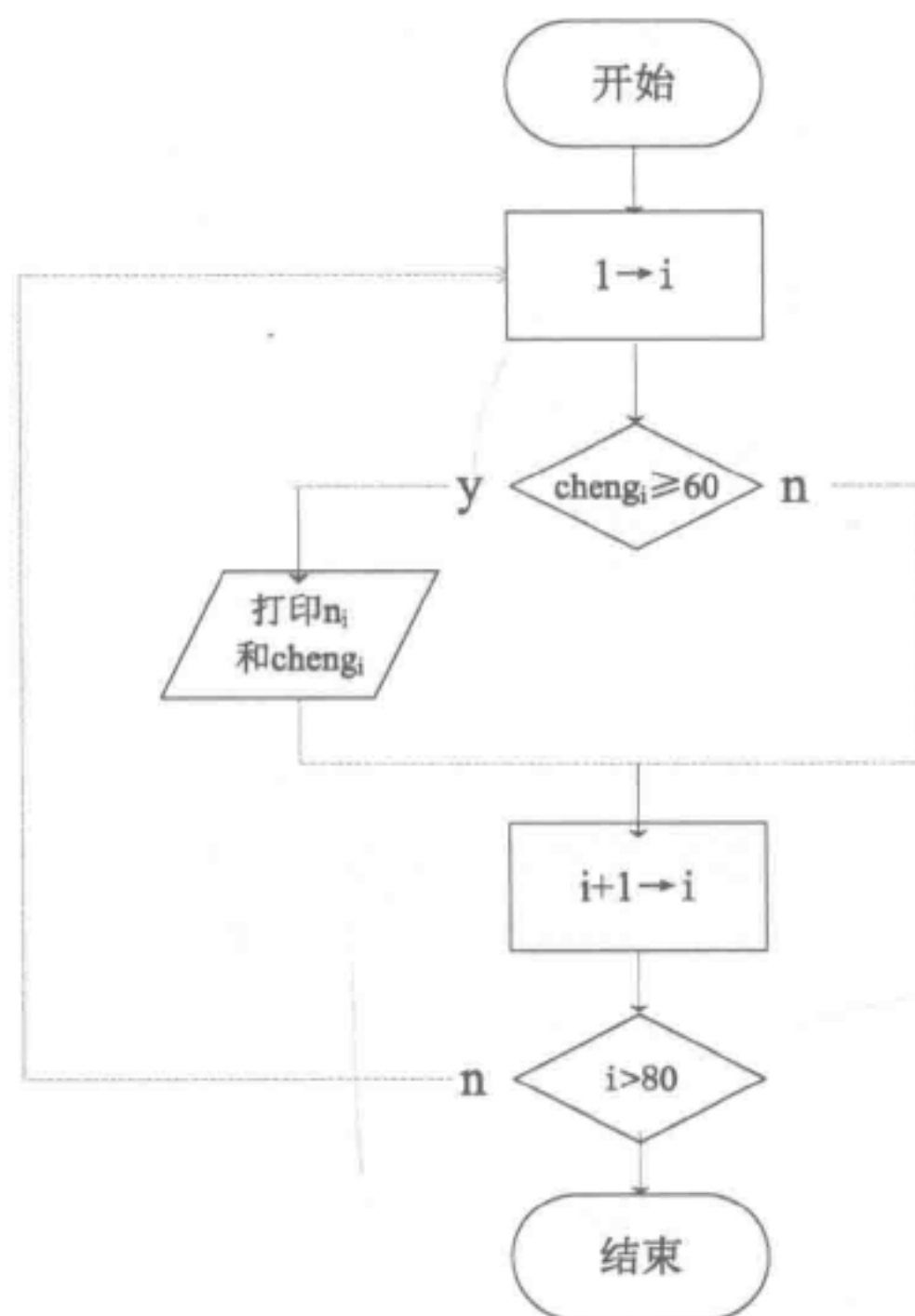


图 11-2 算法流程图

在我们的日常流程设计应用中，通常使用如下所示的 3 种流程图结构。

- 顺序结构：顺序结构如图 11-3 所示，其中 A 和 B 两个框是顺序执行的。即在执行完 A 以后再执行 B 的操作。顺序结构是一种基本结构。
- 选择结构：选择结构也称为分支结构，如图 11-4 所示。此结构中必含一个判断框，根据给定的条件是否成立而选择是执行 A 框还是 B 框。无论条件是否成立，只能执行 A 框或 B 框之一，也就是说，A、B 两框只有一个，也必须有一个被执行。

若两框中有一框为空，程序仍然按两个分支的方向运行。

- 循环结构：循环结构分为两种，一种是当型循环，一种是直到型循环。当型循环是先判断条件  $P$  是否成立，成立才执行  $A$  操作，而直到型循环是先执行  $A$  操作再判断条件  $P$  是否成，成立又进行  $A$  操作。如图 11-5 所示。

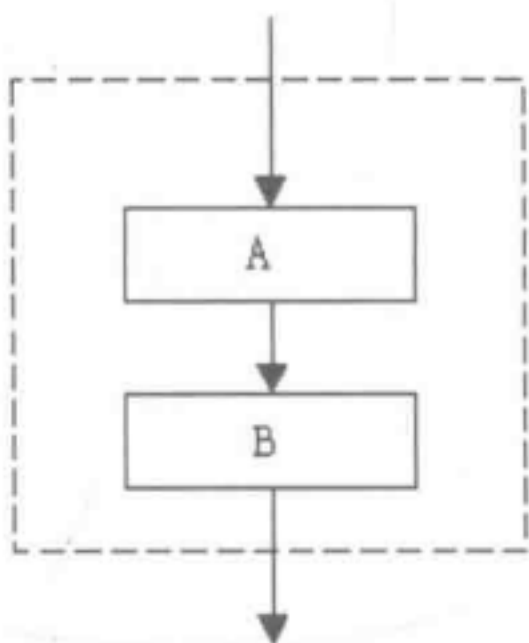


图 11-3 顺序结构

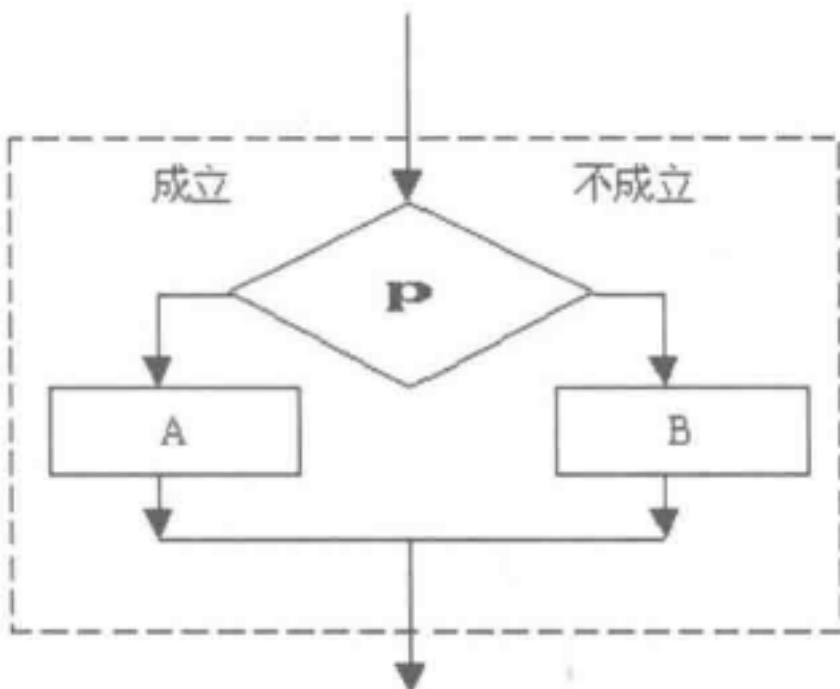


图 11-4 选择结构

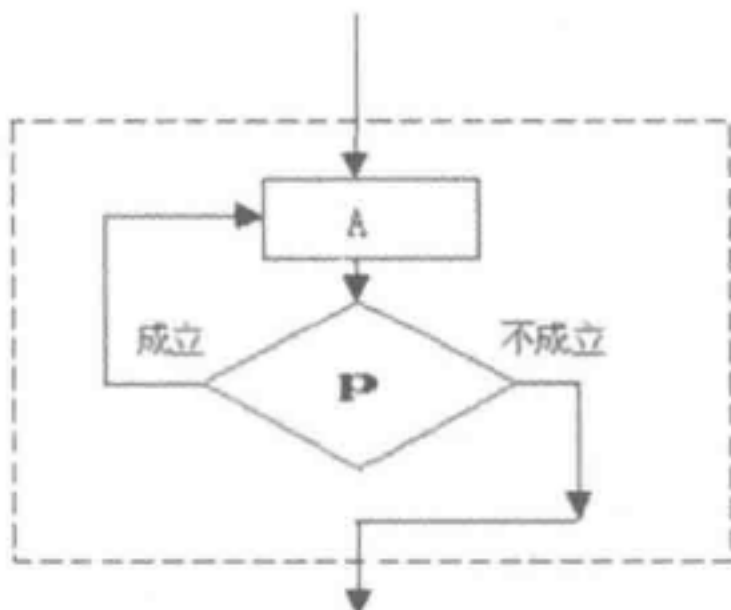
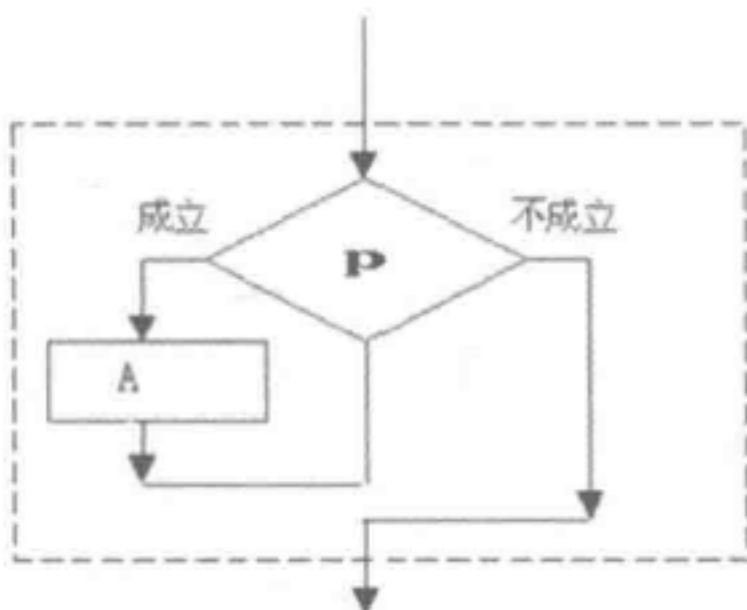


图 11-5 循环结构

细心的读者可以发现，上述 3 种基本结构有如下 4 个很重要的特点，对于我们理解算法是很有帮助的：

- 只有一个入口。
- 只有一个出口。
- 结构内的每一部分都有机会被执行到。
- 结构内不应当存在“死循环”。

## 2. 用 N-S 流程图来表示算法

在 1973 年，聪明的美国学者提出了 N-S 流程图的概念，通过它可以表示计算机的算法。N-S 流程图由一些特定意义的图形、流程线及简要的文字说明构成，能够比较清晰明确地表示程序的运行过程。N-S 图的推出背景很有渊源，人们在使用传统流程图的过程中，发现流程线不一定是必需的，所以设计了一种新的流程图，这种新的方式可以把整个程序写在一个大框图内，这个大框图由若干个小的基本框图构成，这种流程图简称 N-S 图。遵循 N-S 流程图的特点，顺序结构表示为如图 11-6 所示的结构，选择结构表示为如图 11-7 所示的结构，循环结构表示为如图 11-8 所示的结构。

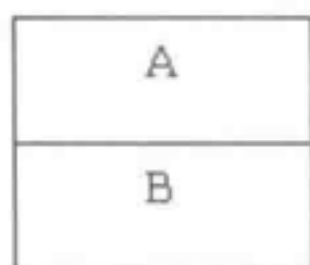


图 11-6 顺序结构

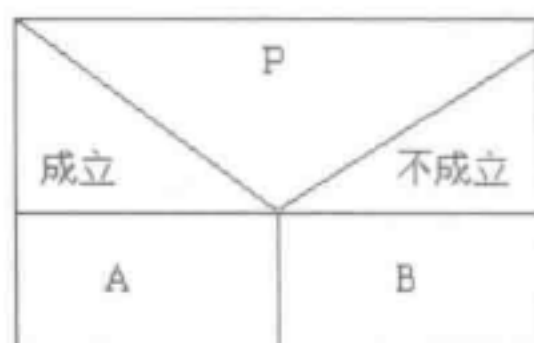


图 11-7 选择结构

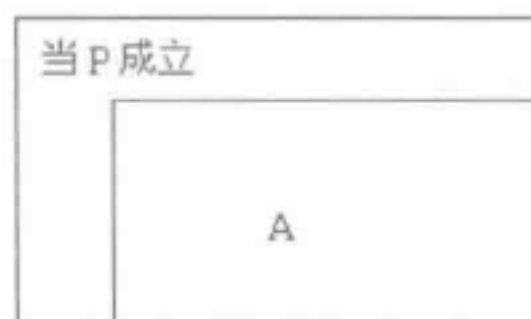


图 11-8 循环结构

## 11.2 新的项目

视频讲解 光盘：视频\第 11 章\新的项目.avi

本项目的客户提出了如下所示的 3 点要求。

- (1) 用 C# 实现。
- (2) 实现对超市内进货、销售和库存的自动化管理。
- (3) 实现智能营销统计。

本项目的具体运作流程如图 11-9 所示。



图 11-9 运作流程

## 11.3 系统需求分析

视频讲解 光盘：视频\第 11 章\系统需求分析.avi

所谓“需求分析”，是指对要解决的问题进行详细的分析，弄清楚问题的要求，包括



需要输入什么数据,要得到什么结果,最后应输出什么。可以说,在软件工程中的“需求分析”,就是确定要计算机“做什么”,要达到什么样的效果。

### 11.3.1 系统背景介绍

目前,无论是公司还是企业,对于货物都实行了信息化管理,以提高管理水平和工作效率,同时也可以最大限度地减少手工操作带来的错误。于是,进销存管理信息系统便应运而生。在工厂中,产品的进销存涉及产品原料的采购、库存、投入生产、报损,甚至有时涉及到销售,同时,对于产品也有相应的生产、库存、销售和报损等环节。在其他非生产性单位,如超市、商店等,则主要涉及到进货、库存、销售和报损 4 个方面。

超市进销存管理的对象是很多的,广而言之,它可以包括:商业、企业或超市的商品,图书馆里的图书,博物馆中的展品等。在这里,仅涉及超市商品。

超市进销存管理系统按分类、分级的模式对仓库进行全面的管理和监控,缩短了超市信息的流转时间,使企业的物资管理层次分明、井然有序,为采购、销售提供依据;智能化的预警功能可自动提示存货的短缺、超储等异常状况;系统还可进行 ABC 分类汇总,减少资金积压。完善的超市管理功能,可对企业的存货进行全面的控制和管理,降低超市成本,增强企业的市场竞争力。

超市进销存管理系统研究的内容涉及超市进销存管理的全过程,包括入库、出库、退货、订货、超市统计查询等。

根据工作流程,超市进销存管理系统将包含以下内容。

(1) 能对企业内的各类货物进行 ABC 分类管理,并提供最低超市量、最高超市量、安全超市量的预警功能。

(2) 可以存储各类信息档案,包括物资、产品基本信息、供货单位信息、使用单位信息等。

(3) 可以方便快捷地进行物资入库管理\物资出库管理等,安全、高效;支持各种类型的出/入库业务:生产入库、委外加工入库、采购入库、其他入库、生产领料出库、委外领料出库、销售出库和其他出库等。

(4) 提供退货管理功能。

(5) 通过查询超市,及时了解超市余额信息,便于订货下单,以免由于缺货影响生产。另外,还提供经济订货量计算功能和打印订货采购单功能。

(6) 支持超市盘点功能,可按仓库、物料进行盘点,自动汇总盘点数据,及时生成盘盈亏调整单。

(7) 可及时打印超市余额,方便领导决策或安排及时订货。

### 11.3.2 功能模块划分

本程序包括 5 个模块,分别是基本档案模块、进货管理模块、销售管理模块、库存管理模块、系统维护模块,具体如图 11-10 所示。

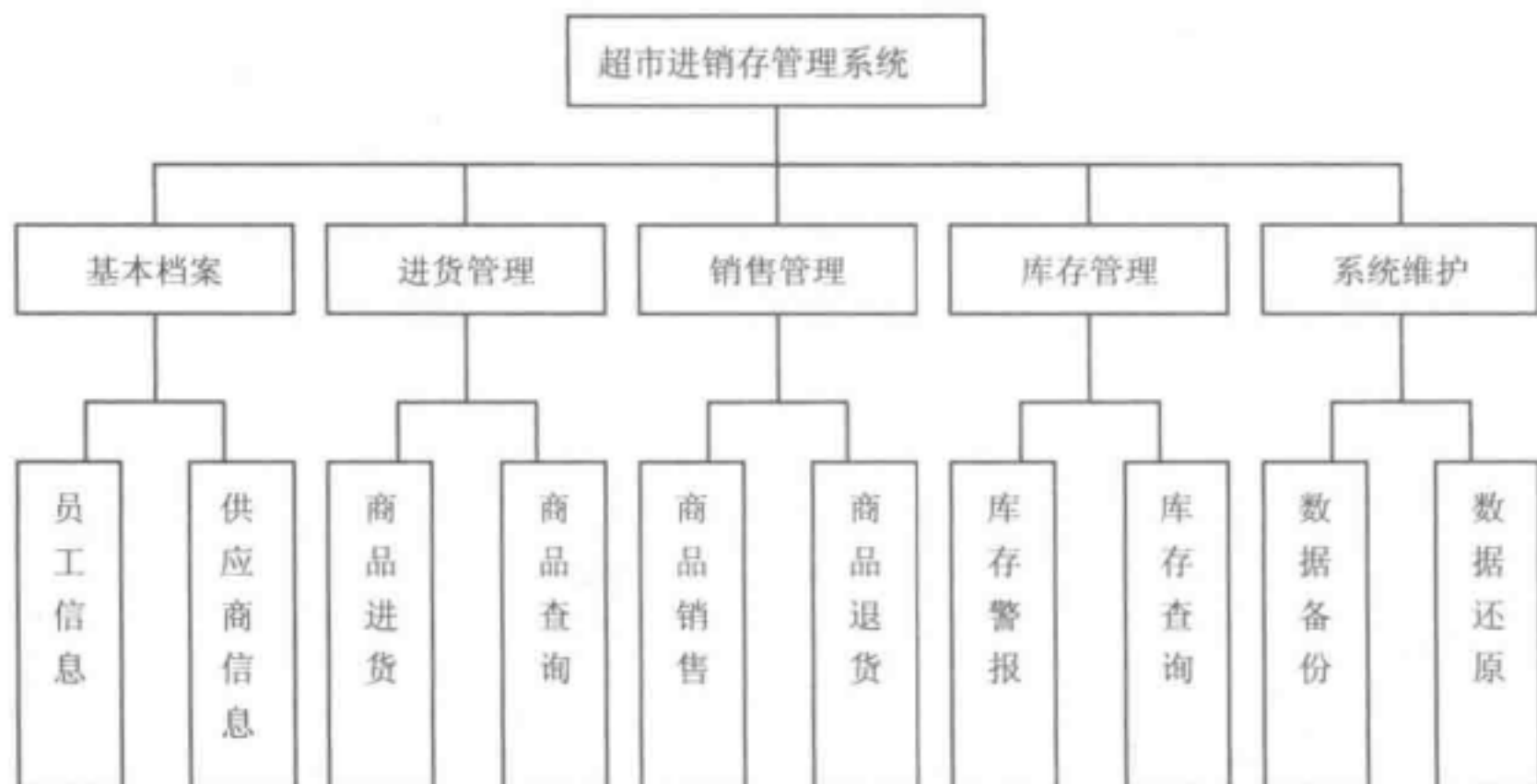


图 11-10 系统的功能模块

## 11.4 规划和运作

视频讲解 光盘：视频\第 11 章\规划和运作.avi

接下来，开始对整个项目的运作进行整体规划，此阶段的工作十分重要，只要规划好具体的实现文件，后面的编码工作将变得更加清晰明了，并具有针对性。

### 11.4.1 规划系统文件

为了在开发过程中更加主动和简洁，决定在 Visual Studio 2013 中规划系统需要的文件，这样，在后期具体编码时，就不需要再考虑如何命名每个模块的实现文件了。所规划的系统文件的具体结构如图 11-11 所示。

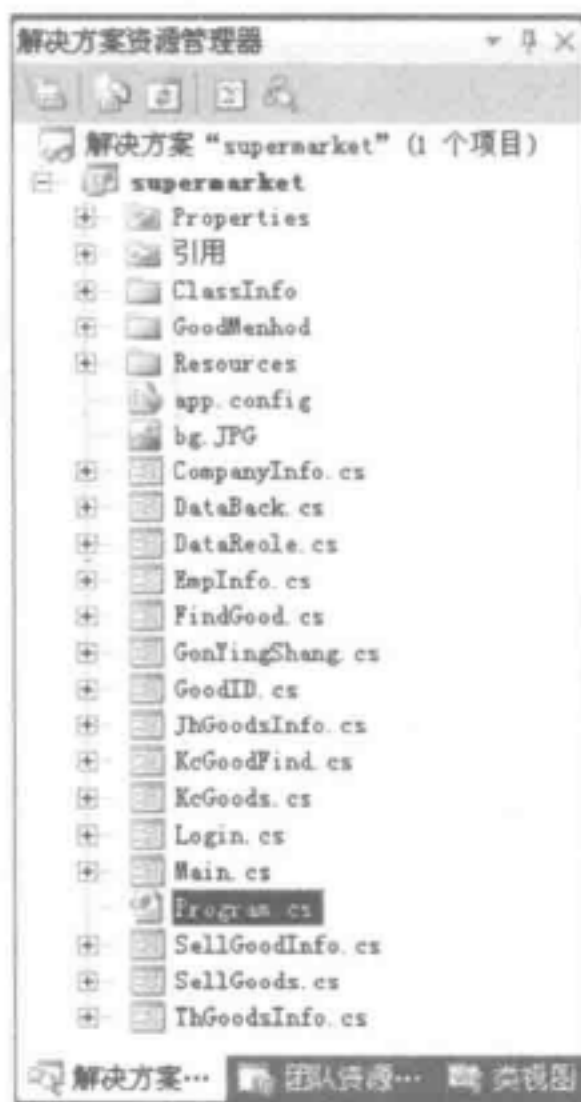


图 11-11 规划的项目文件结构

## 11.4.2 运作流程

规划系统文件工作结束后，接着分析出整个系统的运作流程。具体如图 11-12 所示。

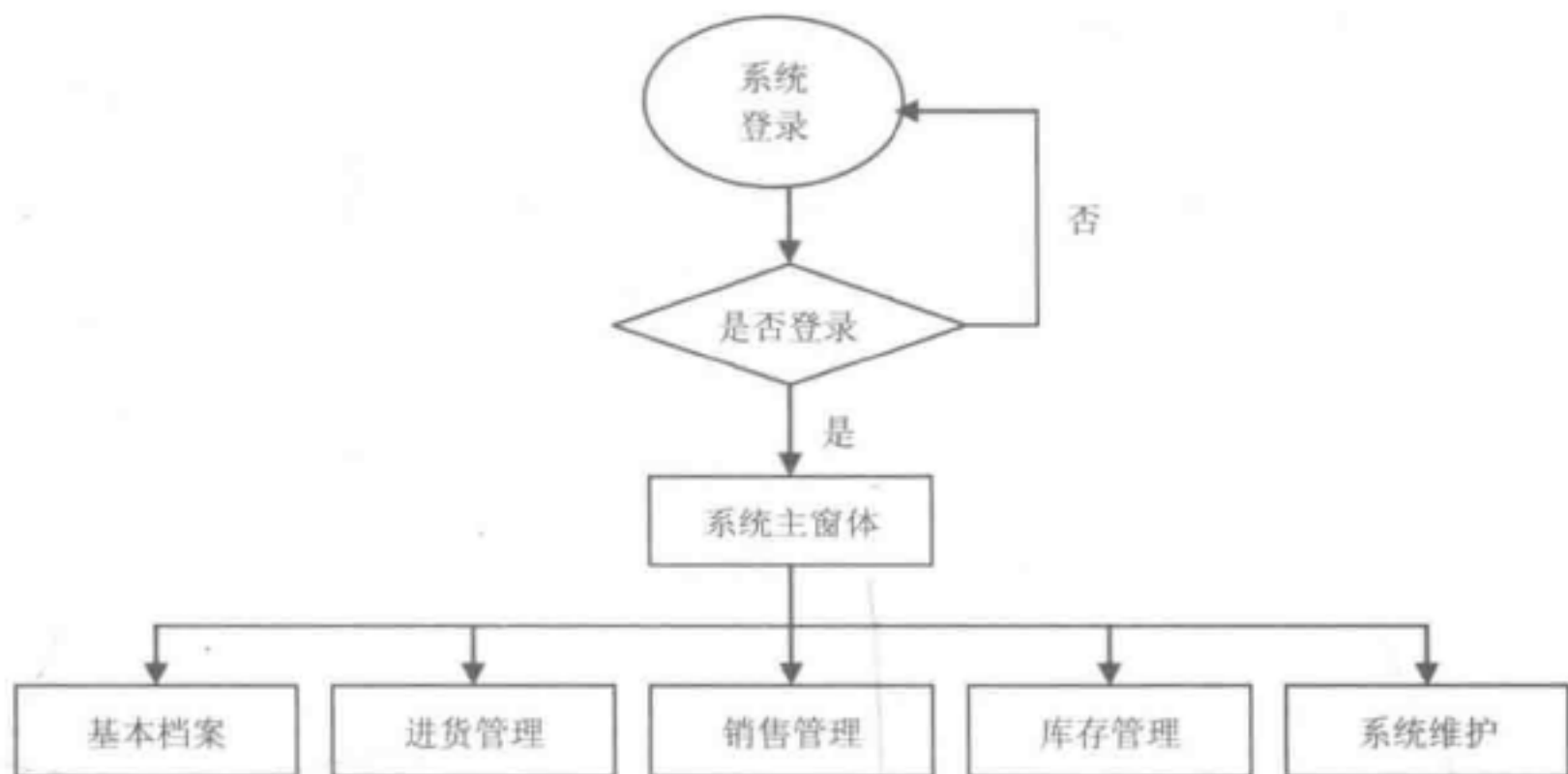


图 11-12 运作流程

## 11.5 设计数据库

视频讲解 光盘：视频\第 11 章\设计数据库.avi

数据库是动态软件技术的基础，本项目将采用 SQL Server 2008 作为数据库工具。

### 11.5.1 数据库的概念设计

超市进销存系统需要提供信息的查询、保存、更新以及删除等功能，这就要求数据库能充分满足各种信息的输入和输出。通过对上述系统功能的分析，针对超市系统的特点，总结出下列需求信息。

(1) 供应商信息实体 E-R 图如图 11-13 所示。

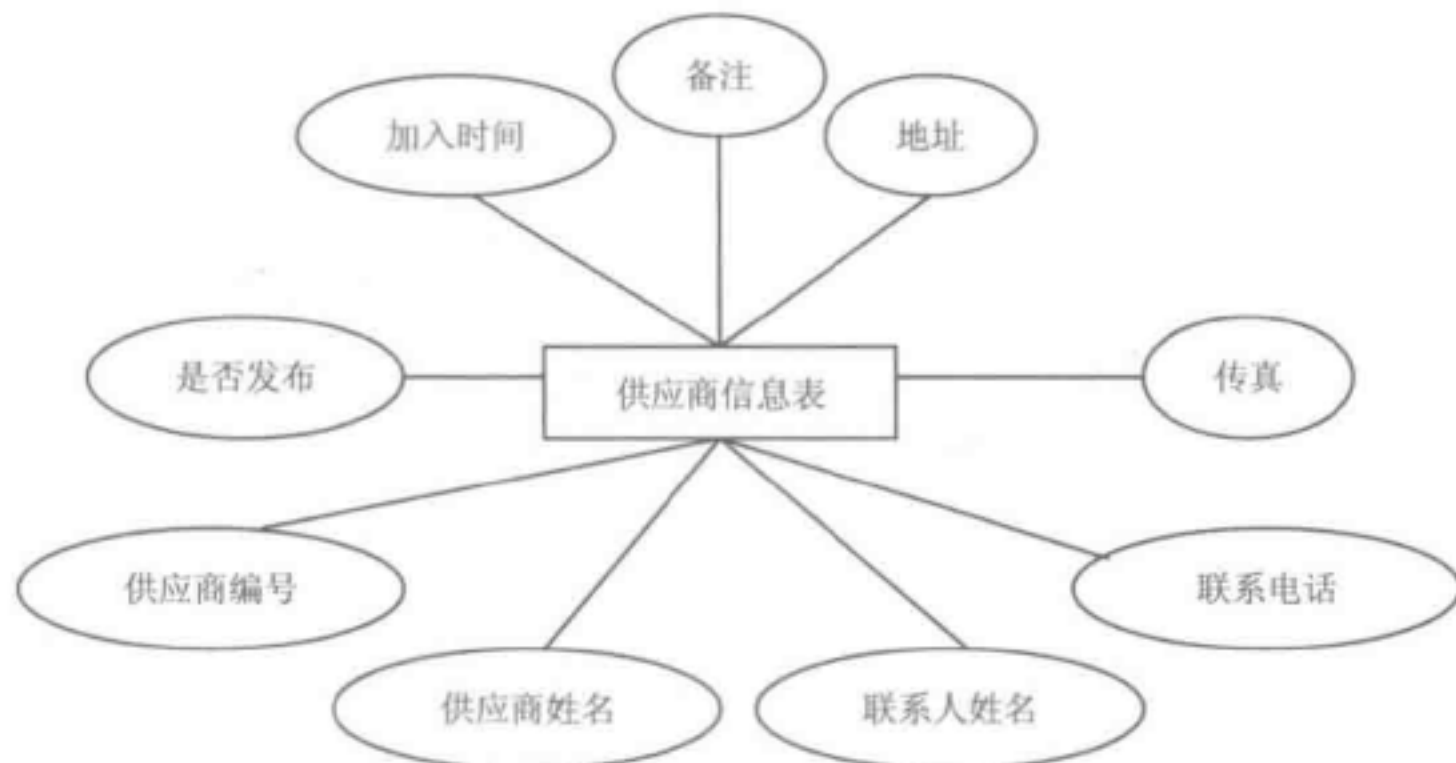


图 11-13 供应商信息实体 E-R 图

(2) 员工信息实体 E-R 图如图 11-14 所示。





图 11-14 员工信息实体 E-R 图

(3) 进货信息实体 E-R 图如图 11-15 所示。



图 11-15 进货信息实体 E-R 图

(4) 库存信息实体 E-R 图如图 11-16 所示。



图 11-16 库存信息实体 E-R 图

(5) 商品销售信息实体 E-R 图如图 11-17 所示。



图 11-17 商品销售信息实体 E-R 图

(6) 商品退货信息实体 E-R 图如图 11-18 所示。

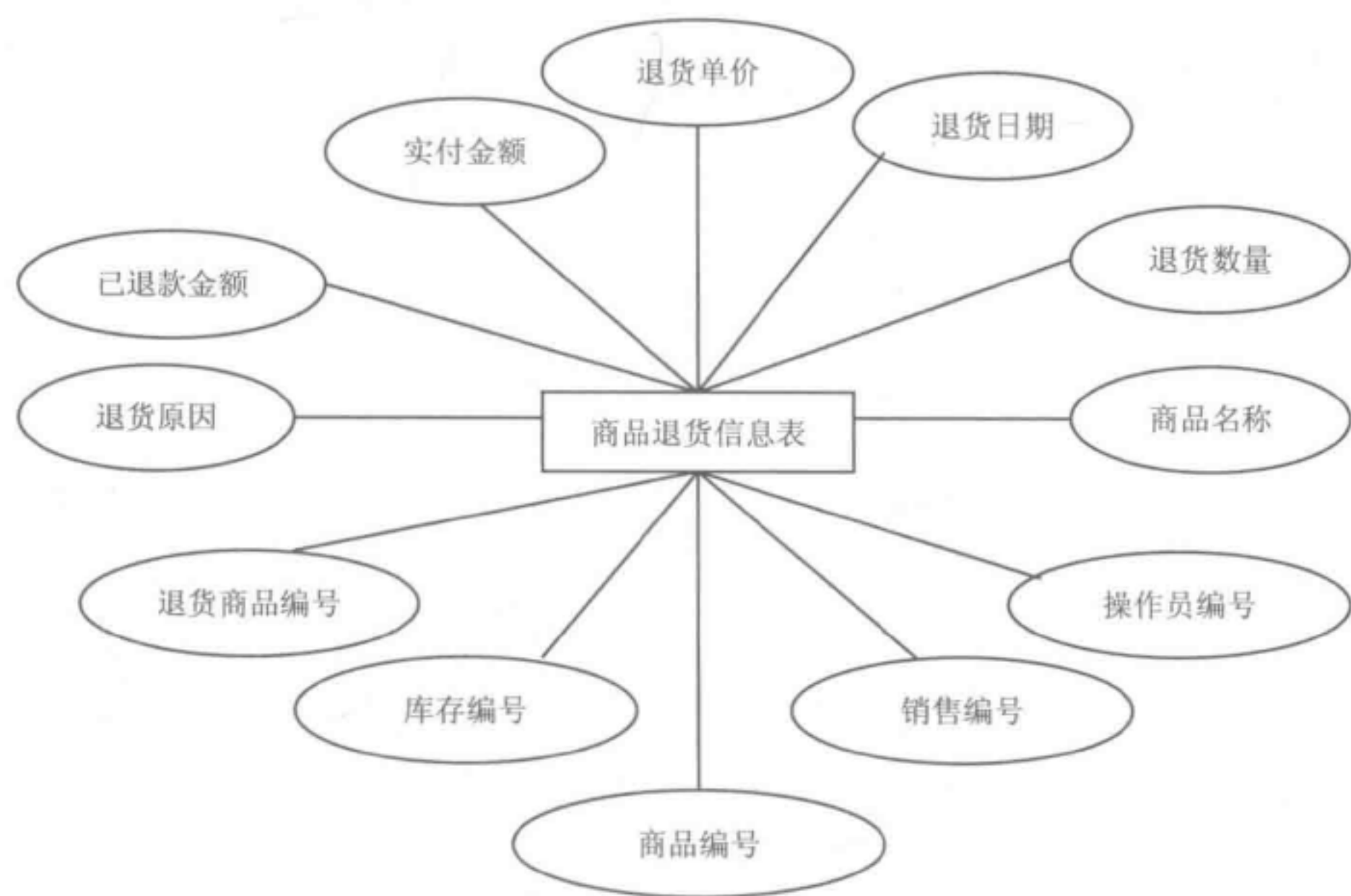


图 11-18 商品退货信息实体 E-R 图

11.5.2 逻辑结构设计

根据设计好的 E-R 图设计数据库 supermarket，然后分别建立各表，下面开始讲解各表的具体逻辑结构。

(1) 供应商信息表 tb\_Company 的结构如表 11-1 所示。

表 11-1 供应商信息表 tb\_Company

| 字段名             | 数据类型     | 长度  | 主键 | 描述    |
|-----------------|----------|-----|----|-------|
| CompanyID       | varchar  | 50  | 否  | 供应商编号 |
| CompanyName     | nvarchar | 100 | 否  | 供应商姓名 |
| CompanyDirector | nvarchar | 50  | 否  | 联系人姓名 |
| CompanyPhone    | nvarchar | 20  | 否  | 联系电话  |
| CompanyFax      | nvarchar | 20  | 否  | 传真    |
| CompanyAddress  | nvarchar | 200 | 否  | 地址    |
| CompanyRemark   | nvarchar | 400 | 否  | 备注    |
| ReDateTime      | datetime | 8   | 否  | 加入日期  |
| Flag            | int      | 4   | 否  | 是否发布  |

(2) 员工信息表 tb\_EmpInfo 的结构如表 11-2 所示。

表 11-2 员工信息表 tb\_EmpInfo

| 列名           | 数据类型     | 长度  | 主键 | 描述    |
|--------------|----------|-----|----|-------|
| EmpID        | nvarchar | 20  | 是  | 员工编号  |
| EmpName      | nvarchar | 20  | 否  | 员工姓名  |
| EmpLoginName | nvarchar | 20  | 否  | 登录 ID |
| EmpLoginPwd  | nvarchar | 20  | 否  | 登录密码  |
| EmpSex       | nvarchar | 4   | 否  | 员工性别  |
| EmpBirthday  | datetime | 8   | 否  | 员工生日  |
| EmpDept      | nvarchar | 20  | 否  | 所属部门  |
| EmpPost      | nvarchar | 20  | 否  | 员工职位  |
| EmpPhone     | nvarchar | 20  | 否  | 家庭电话  |
| EmpPhoneM    | nvarchar | 20  | 否  | 手机号码  |
| EmpAddress   | nvarchar | 200 | 否  | 家庭住址  |
| EmpFlag      | int      | 1   | 否  | 是否发布  |

(3) 进货信息表 tb\_JhGoodsInfo 表的结构如表 11-3 所示。

表 11-3 进货信息表 tb\_JhGoodsInfo

| 字段名        | 数据类型     | 长度  | 主键 | 描述     |
|------------|----------|-----|----|--------|
| GoodsID    | nvarchar | 20  | 是  | 商品编号   |
| EmpId      | nvarchar | 20  | 否  | 操作员编号  |
| JhCompName | nvarchar | 100 | 否  | 进货公司名称 |
| DepotName  | nvarchar | 20  | 否  | 仓库名称   |



续表

| 字 段 名          | 数据类型     | 长 度 | 主 键 | 描 述    |
|----------------|----------|-----|-----|--------|
| GoodsName      | nvarchar | 50  | 否   | 货物名称   |
| GoodsNum       | int      | 4   | 否   | 商品数量   |
| GoodsUnit      | nvarchar | 20  | 否   | 商品计量单位 |
| GoodsJhPrice   | nvarchar | 8   | 否   | 进货单价   |
| GoodsSellPrice | nvarchar | 8   | 否   | 销售单价   |
| GoodsNeedPrice | nvarchar | 8   | 否   | 应付金额   |
| GoodsNoPrice   | nvarchar | 8   | 否   | 实付金额   |
| GoodsRemark    | nvarchar | 200 | 否   | 备注     |
| GoodTime       | datetime | 8   | 否   | 进货时间   |
| Flag           | int      | 4   | 否   | 删除标记   |

(4) 库存信息表 tb\_KcGoods 的结构如表 11-4 所示。

表 11-4 库存信息表 tb\_KcGoods

| 字 段 名        | 数据类型     | 长 度 | 主 键 | 描 述    |
|--------------|----------|-----|-----|--------|
| KcID         | nvarchar | 50  | 否   | 库存编号   |
| GoodsID      | nvarchar | 50  | 是   | 商品编号   |
| JhCompName   | nvarchar | 100 | 否   | 供应商名称  |
| KcDeptName   | nvarchar | 20  | 否   | 仓库名称   |
| KcGoodsName  | nvarchar | 20  | 否   | 商品名称   |
| KcNum        | int      | 4   | 否   | 库存数量   |
| KcAlarmNum   | int      | 4   | 否   | 警报数量   |
| KcUnit       | nvarchar | 20  | 否   | 商品计量单位 |
| KcTime       | datetime | 8   | 否   | 进货时间   |
| KcGoodsPrice | nvarchar | 8   | 否   | 进货价格   |
| KcSellPrice  | nvarchar | 8   | 否   | 销售价格   |
| KcEmp        | nvarchar | 50  | 否   | 进货人    |
| KcRemark     | nvarchar | 200 | 否   | 备注     |

(5) 商品销售信息表 tb\_SellGoods 的结构如表 11-5 所示。

表 11-5 商品销售信息表 tb\_SellGoods

| 字 段 名   | 数据类型     | 长 度 | 主 键 | 描 述  |
|---------|----------|-----|-----|------|
| SellID  | nvarchar | 20  | 是   | 销售编号 |
| KcID    | nvarchar | 50  | 否   | 库存编号 |
| GoodsID | nvarchar | 20  | 否   | 商品编号 |
| EmpId   | nvarchar | 20  | 否   | 员工编号 |

续表

| 字段名           | 数据类型     | 长度  | 主键 | 描述   |
|---------------|----------|-----|----|------|
| GoodsName     | nvarchar | 50  | 是  | 商品名称 |
| SellGoodsNum  | int      | 4   | 否  | 销售数量 |
| SellGoodsTime | nvarchar | 8   | 否  | 销售时间 |
| SellPrice     | nvarchar | 8   | 否  | 销售单价 |
| SellNeedPay   | nvarchar | 8   | 否  | 应付金额 |
| SellHasPay    | nvarchar | 8   | 否  | 实付金额 |
| SellRemark    | nvarchar | 200 | 否  | 备注   |
| SellFlag      | int      | 4   | 否  | 删除标记 |

(6) 商品退货信息表 tb\_ThGoodsInfo 的结构如表 11-6 所示。

表 11-6 商品退货信息表 tb\_ThGoodsInfo

| 字段名           | 数据类型     | 长度  | 主键 | 描述     |
|---------------|----------|-----|----|--------|
| ThGoodsID     | nvarchar | 50  | 是  | 退货销售编号 |
| KcID          | nvarchar | 50  | 否  | 库存编号   |
| GoodsID       | nvarchar | 50  | 否  | 商品编号   |
| SellID        | nvarchar | 50  | 是  | 销售编号   |
| EmpID         | nvarchar | 20  | 是  | 操作员编号  |
| ThGoodsName   | nvarchar | 50  | 否  | 商品名称   |
| ThGoodsNum    | int      | 4   | 否  | 退货数量   |
| ThGoodsTime   | datetime | 8   | 否  | 退货日期   |
| ThGoodsPrice  | nvarchar | 8   | 否  | 退货单价   |
| ThNeedPay     | nvarchar | 8   | 否  | 实付金额   |
| ThHasPay      | nvarchar | 8   | 否  | 已退款金额  |
| ThGoodsResult | nvarchar | 400 | 否  | 退货原因   |

11.6 设计公共类

视频讲解 光盘：视频\第 11 章\设计公共类.avi

为了以后的编码方便考虑，决定预先为本系统设计公共类，这样，系统中可以直接或间接继承此类，从而允许以最少的代码修改来约束整个系统，并为之提供功能。

1. tb\_ThGoodsInfo 类

tb\_ThGoodsInfo 类是商品退货信息实体类，功能是传递和商品退货信息相关的参数。实现文件是 tb\_ThGoodsInfo.cs，具体实现代码如下所示：

```
using System;
```

```
using System.Collections.Generic;
using System.Text;
namespace CHEXC.ClassInfo
{
    public class tb_ThGoodsInfo
    {
        /// <summary>
        ///
        /// </summary>
        /// 退货编号
        private string ThGoodsID;
        public string strThGoodsID
        {
            get{ return ThGoodsID;}
            set{ ThGoodsID=value;}
        }

        private string KcID;
        public string strKcID
        {
            get{ return KcID;}
            set{ KcID=value;}
        }
        private string GoodsID;
        public string strGoodsID
        {
            get{ return GoodsID;}
            set{ GoodsID=value;}
        }
        private string SellID;
        public string strSellID
        {
            get{ return SellID;}
            set{ SellID=value;}
        }
        private string EmpId;
        public string intEmpId
        {
            get{ return EmpId;}
            set{ EmpId=value;}
        }
        private string ThGoodsName;
        public string strThGoodsName
        {
            get{ return ThGoodsName;}
            set{ ThGoodsName=value;}
        }
        private int ThGoodsNum;
        public int intThGoodsNum
        {
            get{ return ThGoodsNum;}
            set{ ThGoodsNum=value;}
        }
        private DateTime ThGoodsTime;
        public DateTime daThGoodsTime
        {
            get{ return ThGoodsTime;}
            set{ ThGoodsTime=value;}
        }
        private string ThGoodsPrice;
        public string deThGoodsPrice
        {

```



```

        get{ return ThGoodsPrice;}
        set{ ThGoodsPrice=value;}
    }
    private string ThNeedPay;                                //应付金额
    public string deThNeedPay
    {
        get{ return ThNeedPay;}
        set{ ThNeedPay=value;}
    }
    private string ThHasPay;                                   //已退款金额
    public string deThHasPay
    {
        get{ return ThHasPay;}
        set{ ThHasPay=value;}
    }
    private string ThGoodsResult;                             //退货原因
    public string deThGoodsResult
    {
        get{ return ThGoodsResult;}
        set{ ThGoodsResult=value;}
    }
}

```

## 2. getSqlConnection 类

getSqlConnection 类的功能是建立与数据库的连接，实现文件是 getSqlConnection.cs，对应的代码如下所示：

```

using System;
using System.Collections.Generic;
using System.Text;

using System.Data.SqlClient;
namespace CHEXC.GoodMenhod
{
    public class getSqlConnection
    {
        #region 代码中用到的变量
        string G_Str_ConnectionString =
            "Data Source=(local);database=supermarket;uid=sa;pwd=8888888";
        SqlConnection G_Con; //声明链接对象
        #endregion

        #region 构造函数
        /// <summary>
        /// 构造函数
        /// </summary>
        public getSqlConnection()
        {
        }
        #endregion

        #region 连接数据库
        /// <summary>
        /// 连接数据库
        /// </summary>
        /// <returns></returns>
        public SqlConnection GetCon()
        {
        }
        #endregion
    }
}

```

```
{
    G_Con = new SqlConnection(G_Str_ConnectionString);
    G_Con.Open();
    return G_Con;
}
#endregion
}
```

### 3. tb\_ThGoodsMethod

tb\_ThGoodsMethod 类封装退货信息表的自定义方法，用 tb\_ThGoodsMethod.cs 文件来实现，下面讲解其具体实现流程。

(1) 方法 tb\_ThGoodsAdd 的功能是添加退货信息，对应的代码如下所示：

```
public int tb_ThGoodsAdd(tb_ThGoodsInfo tbChGood)
{
    int intFlag = 0;
    try
    {
        string str_Add = "insert into tb_ThGoodsInfo values( ";
        str_Add += "'" + tbChGood.strThGoodsID + "','" + tbChGood.strKcID
            + "','" + tbChGood.strGoodsID + "','";
        str_Add += "'" + tbChGood.strSellID + "','" + tbChGood.intEmpId
            + "','" + tbChGood.strThGoodsName + "','";
        str_Add += "'" + tbChGood.intThGoodsNum + "','" + tbChGood.daThGoodsTime
            + "','" + tbChGood.deThGoodsPrice + "','";
        str_Add += "'" + tbChGood.deThHasPay + "','" + tbChGood.deThNeedPay
            + "','" + tbChGood.deThGoodsResult + "')";
        getSqlConnection getConnection = new getSqlConnection();
        conn = getConnection.GetCon();
        cmd = new SqlCommand(str_Add, conn);
        intFlag = cmd.ExecuteNonQuery();
        conn.Dispose();
        return intFlag;
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.ToString());
        return intFlag;
    }
}
```

(2) 方法 tb\_ThGoodsUpdate 的功能是修改退货信息，对应的代码如下所示：

```
public int tb_ThGoodsUpdate(tb_ThGoodsInfo tbChGood)
{
    int intFlag = 0;
    try
    {
        string str_Add = "update tb_ThGoodsInfo set ";
        str_Add += "KcID='" + tbChGood.strKcID + "',GoodsID='"
            + tbChGood.strGoodsID + "','";
        str_Add += "SellID='" + tbChGood.strSellID + "',EmpId='" + tbChGood.intEmpId
            + "',ThGoodsName='" + tbChGood.strThGoodsName + "','";
        str_Add += "ThGoodsNum=" + tbChGood.intThGoodsNum + "',ThGoodsTime='"
            + tbChGood.daThGoodsTime + "',ThGoodsPrice="
            + tbChGood.deThGoodsPrice + "','";
        str_Add += "ThHasPay=" + tbChGood.deThHasPay + "',ThNeedPay="
            + tbChGood.deThNeedPay + "',ThGoodsResult='"
```

```

        + tbChGood.deThGoodsResult + "' where ThGoodsID='"
        + tbChGood.strThGoodsID + "'";
    getSqlConnection getConnection = new getSqlConnection();
    conn = getConnection.GetCon();
    cmd = new SqlCommand(str_Add, conn);
    intFlag = cmd.ExecuteNonQuery();
    conn.Dispose();
    return intFlag;
}
catch (Exception ee)
{
    MessageBox.Show(ee.ToString());
    return intFlag;
}
}

```

在上述实现代码中，用到了 C# 程序的异常处理机制。任何完美的应用程序和技术高明的程序员，都不可能是绝对不出差错的。与其追求完美无错的代码，还不如将程序中可能预知的异常在发布前很好地进行处理，可能是最有价值的。

那么，C# 是如何处理异常的呢？首先从最普通的异常说起，即使用 try-catch-finally 块捕获异常，基本格式如下：

```

try
{
    //获取并使用资源，可能出现异常
}
catch(DivideByZeroException de)
{
}
catch(ArithmeticException ae)
{
}
catch(Exception e)
{
    //捕获并处理异常，当出现多个异常且异常类之间有继承关系(DivideByZeroException==>
    //ArithmeticException==>Exception)时，捕获顺序是子类在前，基类在后
}
finally
{
    //无论什么情况下(即使在 catch 块中有 return)，都会执行该块的代码(如关闭文件)
    //另外需要说明的是，在 finally 块中使用任何 break、continue、return 退出都是非法的
}

```

在 WinForm 应用程序中，除了可以用 try-catch-finally 方式外，如何实现全局的异常处理呢？可以通过委托的方式来实现，例如下面的演示代码：

```

internal class ThreadExceptionHandler
{
    //实现错误异常事件
    public void Application_ThreadException(object sender, ThreadExceptionHandlerEventArgs e)
    {
        try
        {
            //如果用户单击 Abort 按钮则退出应用程序
            DialogResult result = ShowThreadExceptionHandlerDialog(e.Exception);
            if (result == DialogResult.Abort)
            {
                Application.Exit();
            }
        }
    }
}

```



```

    }
}
catch
{
    try
    {
        MessageBox.Show(
            "严重错误", "严重错误", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }
    finally
    {
        Application.Exit();
    }
}
}
private DialogResult ShowThreadExceptionHandlerDialog(Exception e)
{
    string errorMsg =
        "错误信息:\t\t" + e.Message + "\t\t" + e.GetType() + "\t\t" + e.StackTrace;
    return MessageBox.Show(errorMsg, "Application Error",
        MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Stop);
}
}
static class Program
{
    ///<summary>
    /// The main entry point for the application.
    ///</summary>
    [STAThread]
    static void Main()
    {
        ThreadExceptionHandler handler = new ThreadExceptionHandler();
        Application.ThreadException +=
            new ThreadExceptionHandlerEventHandler(handler.Application_ThreadException);
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new frmEvent());
    }
}
public partial class frmEvent : Form
{
    private void btnException_Click(object sender, EventArgs e)
    {
        throw new InvalidOperationException("无效的操作异常!");
    }
}
}

```

另外,读者也可以自定义错误处理信息或想要进行的其他操作。

(3) 方法 `tb_ThGoodsID()` 的功能是自动生成商品流水号,对应的代码如下所示:

```

public string tb_ThGoodsID()
{
    int intYear = DateTime.Now.Day;
    int intMonth = DateTime.Now.Month;
    int intDate = DateTime.Now.Year;
    int intHour = DateTime.Now.Hour;
    int intSecond = DateTime.Now.Second;
    int intMinute = DateTime.Now.Minute;
    string strTime = null;
    strTime = intYear.ToString();
    if (intMonth < 10)

```

```
{
    strTime += "0" + intMonth.ToString();
}
else
{
    strTime += intMonth.ToString();
}
if (intDate < 10)
{
    strTime += "0" + intDate.ToString();
}
else
{
    strTime += intDate.ToString();
}
if (intHour < 10)
{
    strTime += "0" + intHour.ToString();
}
else
{
    strTime += intHour.ToString();
}
if (intMinute < 10)
{
    strTime += "0" + intMinute.ToString();
}
else
{
    strTime += intMinute.ToString();
}
if (intSecond < 10)
{
    strTime += "0" + intSecond.ToString();
}
else
{
    strTime += intSecond.ToString();
}
return ("TH-" + strTime);
}
```

在上述代码中，月份小于 10 则在月份前加 0；天数小于 10 则在天数前加 0，小时小于 10 则在小时前加 0；分钟小于 10 则在分钟前加 0；秒小于 10 则在秒前加 0。

(4) 方法 `tb_ThGoodsFind(Object DataObject)` 的功能是将退货信息显示在 `DataGridView` 控件中，对应的代码如下所示：

```
public void tb_ThGoodsFind(Object DataObject)
{
    int intCount = 0;
    string strSecar = null;
    try
    {
        strSecar = "select * from tb_ThGoodsInfo ";
        getSqlConnection getConnection = new getSqlConnection();
        conn = getConnection.GetCon();
        cmd = new SqlCommand(strSecar, conn);
        int ii = 0;
        qlddr = cmd.ExecuteReader();
        while (qlddr.Read())
```

```

{
    ii++;
}
qlddr.Close();
System.Windows.Forms.DataGridView dv = (DataGridView)DataObject;
if (ii != 0)
{
    int i = 0;
    dv.RowCount = ii;
    qlddr = cmd.ExecuteReader();
    while (qlddr.Read())
    {
        dv[0, i].Value = qlddr[0].ToString();
        dv[1, i].Value = qlddr[3].ToString();
        dv[2, i].Value = qlddr[5].ToString();
        dv[3, i].Value = qlddr[8].ToString();
        dv[4, i].Value = qlddr[6].ToString();
        i++;
    }
    qlddr.Close();
}
else
{
    if (dv.RowCount != 0)
    {
        int i = 0;
        do
        {
            dv[0, i].Value = "";
            dv[1, i].Value = "";
            dv[2, i].Value = "";
            dv[3, i].Value = "";
            i++;
        } while (i < dv.RowCount);
    }
}
}
catch (Exception ee)
{
    MessageBox.Show(ee.ToString());
}
}

```

(5) 方法 filltProd(object objTreeView, object obimage)的功能是将商品类别信息中的商品类别添加到 TreeView 控件中, 对应的代码如下所示:

```

public void filltProd(object objTreeView, object obimage)
{
    try
    {
        getSqlConnection getConnection = new getSqlConnection();
        conn = getConnection.GetCon();
        string strSecar = "select * from tb_SellGoods ";
        cmd = new SqlCommand(strSecar, conn);
        qlddr = cmd.ExecuteReader();

        if (objTreeView.GetType().ToString() == "System.Windows.Forms.TreeView")
        {
            System.Windows.Forms.ImageList imlist =
                (System.Windows.Forms.ImageList)obimage;

```



```

        System.Windows.Forms.TreeView TV =
            (System.Windows.Forms.TreeView)objTreeView;
        TV.Nodes.Clear();
        TV.ImageList = imlist;
        System.Windows.Forms.TreeNode TN = TV.Nodes.Add("A", "商品销售信息", 0, 1);
        while (qlddr.Read())
        {
            TreeNode newNode12 = new TreeNode(qlddr[0].ToString(), 0, 1);
            newNode12.Nodes.Add("A", qlddr[4].ToString(), 0, 1);
            TN.Nodes.Add(newNode12);
        }
        qlddr.Close();
        TV.ExpandAll();
    }
}
catch (Exception ee)
{
    MessageBox.Show(ee.ToString());
}
}

```

(6) 方法 `tb_ThGoodsDelete()` 的功能是删除商品信息，对应的代码如下所示：

```

public int tb_ThGoodsDelete(string striThid)
{
    int intFlag = 0;
    try
    {
        string str_Add = "delete from tb_thgoodsinfo where ThGoodsID='" + striThid + "'";
        getSqlConnection getConnection = new getSqlConnection();
        conn = getConnection.GetCon();
        cmd = new SqlCommand(str_Add, conn);
        intFlag = cmd.ExecuteNonQuery();
        conn.Dispose();
        return intFlag;
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.ToString());
        return intFlag;
    }
}

```

通过上述代码的实现过程，可以了解到基类的重要性。提到基类，就不得不说派生类，两者一直在 C# 体系中占据重要地位。为了使读者加深对基类和派生类的印象，接下来对基类和派生类进行简单的总结。

(1) 基类和派生类的一些基本概念。

`class B : A {}`：A 是基类，B 是派生类。一个类最多只允许从一个类中派生。

`class C : B {}`：B 还可以充当 C 的派生类。继承默认总是 `public` 的，`System.Object` 是所有类的根，编译器会把我们的类悄悄地编译成 `class A : System.Object {}`。

(2) 调用基类的构造函数：

```

class B : A {
    public B(string sqlstr) : base(sqlstr) {}
}

```

(3) 基类派生类对象之间的关系：

```

class A {}
class B : A {}
class C : A {}
B b = new B();
C c = b; //这样写是错误的, 因为类型不同
A a = b; //这样写是正确的, 但是一定要注意

```

这样做有一个明显的限制, a 对象只能访问基类中的成员, 不能访问派生类中的成员, 这就是为什么 C# 完全面向对象的原因, 因为 C# 中所有的东西都继承自 System.Object, 任何东西都能赋给一个 object 变量。

#### (4) 方法的隐藏。

如果一个基类有 n 个方法和 n 个派生类, 这 n 个派生类里又分别有 n 个方法, 这时会产生一个问题, 就是会遇到签名完全一样的方法(方法名\参数的数量\类型完全一样), 如果出现这样的情况, 在编译过程中将收到一个警告, 不要惊慌, 仅仅是警告而已, 不会影响编译的过程。但是应该认真地对待这个警告, 如果派生类的一个对象将调用基类的一个方法, 然而这个方法又在这个派生类里有相同签名的方法, 那编译器该怎么办呢? 对象调用的是派生类里的方法, 这种现象在微软官方叫方法的隐藏。

如果不想看到那个警告, 可以在派生类中与基类同签名的方法前面加个 new 关键字, 告诉编译器: 会为造成的结果负责, 不要警告! 注意这个关键字只是起个屏蔽警告的作用, 派生类的对象照样还是不能调用基类中同签名的方法。

#### (5) 方法的覆盖, 先看个例子:

```

class nvren {
    public virtual string leixing()
    { return "这是个女人"; }
}
class meinv : nvren
{
    public override string leixing()
    { return "这是个美女"; }
}
class weizhi : nvren
{
    ... //这里并没有任何覆盖原方法的方法
}
nvren a = new nvren();
meinv b = new meinv();
weizhi c = new weizhi();
Console.WriteLine(a.leixing());
Console.WriteLine(b.leixing());
Console.WriteLine(c.leixing());

```

程序很简单, 当调用第三个 WriteLine 时, 程序将输出“这是个女人”, 现在不用解释就会明白。这种同一个语句调用不同方法的现象称为多态性。

另外, 在使用 virtual 和 override 时一定要注意:

- 两个方法必须签名相同。
- 两个方法均不能是 private 方法, 且必须有相同的可访问性。
- 不要试图 override 没有 virtual 的方法。
- 不要试图不用 override 就覆盖基类的 virtual 方法(那就成了方法的隐藏了)。
- override 方法将隐式地成为 virtual 方法, 它本身可在未来的派生类里被覆盖。

## 11.7 具体编码

视频讲解 光盘：视频\第 11 章\具体编码.avi

本节开始步入具体的编码工作。现在既有项目规划书，也有公共类。通过这些资料，整个项目的编码思路就变得十分清晰了，只需遵循规划书的指引即可轻松实现。

### 11.7.1 用户登录模块

此模块的功能是验证登录信息，确保只有是系统的合法用户才能登录系统。设计后的登录窗体 Login.cs 的界面效果如图 11-19 所示。



图 11-19 登录窗体界面

文件 Login.cs 的具体实现代码如下所示：

```
using System.Windows.Forms;
using CHEXC.GoodMenhod;
namespace CHEXC
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }
        private void btnOK_Click(object sender, EventArgs e)
        {
            tb_EmpInfoMethod tbEmp = new tb_EmpInfoMethod();
            if (txtID.Text == "")
            {
                MessageBox.Show("用户名不能为空!");
                return;
            }
            if (txtPwd.Text == "")
            {
                MessageBox.Show("密码不能为空!");
                return;
            }
            if (tbEmp.tb_EmpInfoFind(txtID.Text, txtPwd.Text, 2) == 1)
```



```

{
    Main frm = new Main(txtID.Text);
    frm.Show();
    this.Hide();
}
else
{
    MessageBox.Show("登录失败!");
}
}
private void btnCancel_Click(object sender, EventArgs e)
{
    Application.Exit();
}
private void frmLogin_FormClosing(object sender, FormClosingEventArgs e)
{
    Application.Exit();
}
private void Login_Load(object sender, EventArgs e)
{
}
}
}

```

### 11.7.2 主窗体模块

主窗体是用户登录后显示的主界面，分为如下三部分。

- 顶部菜单：实现一些对应的操作处理。
- 中间提示：显示对应的提示信息。
- 底部提示：显示系统的当前状态信息。

主窗体 EmplInfo.cs 的界面效果如图 11-20 所示。

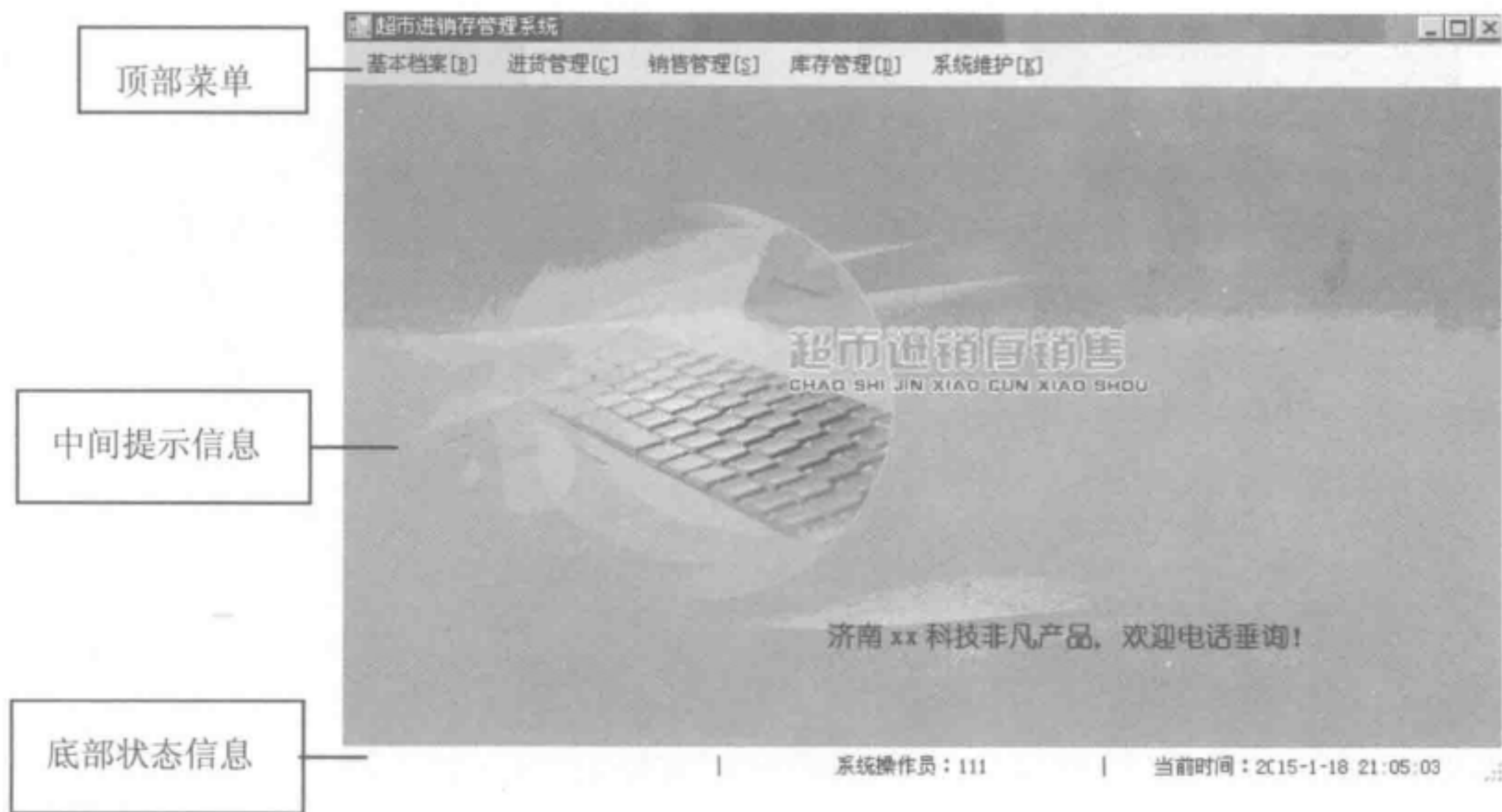


图 11-20 系统主界面

下面开始讲解实现文件 EmplInfo.cs 的具体实现流程。

(1) 载入后, 先显示当前的登录名和当前系统时间, 对应的代码如下所示:

```
public partial class Main : Form
{
    public Main()
    {
        InitializeComponent();
    }
    public Main(string strName)
    {
        InitializeComponent();
        SendNameValue = strName;
    }
    public string SendNameValue;

    private void frmMain_Load(object sender, EventArgs e)
    {
        timer2.Enabled = true;
        this.statusUser.Text = "系统操作员: " + SendNameValue;
    }
    private void timer2_Tick(object sender, EventArgs e)
    {
        this.statusTime.Text = "当前时间: " + DateTime.Now.ToString();
    }
}
```

(2) 定义“进货命令”菜单的处理事件, 对应的代码如下所示:

```
private void menuGoodsIn_Click(object sender, EventArgs e)
{
    //进货信息
    JhGoodsInfo jhGOOD = new JhGoodsInfo();
    jhGOOD.Owner = this;
    jhGOOD.ShowDialog();
}
```

(3) 定义“员工信息”菜单的处理事件, 对应的代码如下所示:

```
private void menuEmployee_Click(object sender, EventArgs e)
{
    //员工信息
    EmpInfo empinfo = new EmpInfo();
    empinfo.Owner = this;
    empinfo.ShowDialog();
}
```

(4) 定义“供应商信息”菜单的处理事件, 对应的代码如下所示:

```
private void menuCompany_Click(object sender, EventArgs e)
{
    //供应商信息
    CompanyInfo frmComp = new CompanyInfo();
    frmComp.Owner = this;
    frmComp.ShowDialog();
}
```

(5) 定义“进货查询”菜单的处理事件, 对应的代码如下所示:

```
private void menuFind_Click(object sender, EventArgs e)
{
    //商品信息查查询
    FindGood findgood = new FindGood();
}
```

```
findgood.Owner = this;
findgood.ShowDialog();
}
```

(6) 定义“库存报警”菜单的处理事件，对应的代码如下所示：

```
private void menuDepotAlarm_Click(object sender, EventArgs e)
{
    //库存警报
    KcGoods kcGood = new KcGoods();
    kcGood.Owner = this;
    kcGood.ShowDialog();
}
```

(7) 定义“库存查询”菜单的处理事件，对应的代码如下所示：

```
private void menuDepotFind_Click(object sender, EventArgs e)
{
    //库存查询
    KcGoodFind kcfrmFind = new KcGoodFind();
    kcfrmFind.Owner = this;
    kcfrmFind.ShowDialog();
}
```

(8) 定义“商品销售信息”菜单的处理事件，对应的代码如下所示：

```
private void menuSellGoods_Click(object sender, EventArgs e)
{
    //商品销售信息
    SellGoods frmSell = new SellGoods();
    frmSell.Owner = this;
    frmSell.ShowDialog();
}
```

(9) 定义“退货信息”菜单的处理事件，对应的代码如下所示：

```
private void menuSellFind_Click(object sender, EventArgs e)
{
    //退货信息
    ThGoodsInfo frmTh = new ThGoodsInfo();
    frmTh.Owner = this;
    frmTh.ShowDialog();
}
```

(10) 定义“数据备份”菜单的处理事件，对应的代码如下所示：

```
private void 数据备份 HToolStripMenuItem_Click(object sender, EventArgs e)
{
    //数据备份
    DataBack frmBack = new DataBack();
    frmBack.Owner = this;
    frmBack.ShowDialog();
}
```

(11) 定义“数据还原”菜单的处理事件，对应的代码如下所示：

```
private void 数据还原 IToolStripMenuItem_Click(object sender, EventArgs e)
{
    //数据还原
    DataReole frmReole = new DataReole();
    frmReole.Owner = this;
    frmReole.ShowDialog();
}
```



```

    }
    private void frmMain_FormClosing(object sender, FormClosingEventArgs e)
    {
        Application.Exit();
    }
}

```

在上述单击不同控件按钮的过程中，演示了 C# 的事件处理机制。任何进行过图形用户界面开发的编程人员都会知道事件的概念，当用户在使用程序的时候，必然要与程序进行一定的交互。比如当用户单击窗体上的一个按钮后，程序就会产生该按钮被点击的事件，并通过相应的事件处理函数来响应用户的操作，这样，用户的直观感觉就是程序执行了其所要求的任务了。当然，事件并不一定是在与用户交互的情况下才会产生的，系统的内部也会产生一些事件并请求处理的，比如时钟事件就是一个很好的例子。要想了解 C# 中的事件处理机制，需要先了解“委托”的基本概念。在 C# 中使用委托后，能够处理在其他编程语言中需要使用函数指针来处理的问题。

委托与函数比较类似，与函数相比，主要有如下 3 点区别：

- 委托是匿名的。
- 委托是面向对象和类型安全的，而函数的指针是不安全的类型。
- 委托同时封装了对象实例和方法，而函数指针仅指向函数成员。

委托不会关心它所封装的方法或所属的类，它只是负责实现这些方法和委托的类型相兼容。

### 11.7.3 进货管理模块

进货管理模块的功能是实现系统内进货信息的管理，主要包括添加、删除、修改、保存等操作。在 Visual Studio 2013 中设计窗体，效果如图 11-21 所示。

图 11-21 进货管理模块窗体的效果

此模块的实现文件是 JhGoodsInfo.cs，下面开始讲解其具体实现流程。

(1) 加载时显示所有的商品信息, 将结果绑定到 DataGridView 控件上。对应的实现代码如下所示:

```
private void frmJhGoodsInfo_Load(object sender, EventArgs e)
{
    jhMethod.tb_JhGoodsInfoFind("", 5, dataGridView1);
}
```

(2) 当单击 DataGridView 控件中的某条信息后, 对应的各项信息会在对应的文本框中显示。对应的实现代码如下所示:

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (intFlag == 2 || intFlag == 3)
    {
        FillControls();
    }
}
```

在上述代码中, 通过调用方法 FillControls() 显示了单击信息的详细内容。方法 FillControls() 的具体实现代码如下所示:

```
private void FillControls()
{
    try
    {
        SqlDataReader sqlldr = jhMethod.tb_JhGoodsInfoFind(this.dataGridView1[0,
            this.dataGridView1.CurrentCell.RowIndex].Value.ToString(), 1);

        sqlldr.Read();
        if (sqlldr.HasRows)
        {
            txtEmpId.Text = sqlldr[1].ToString();
            txtGoodsName.Text = sqlldr[4].ToString();
            cmbDepotName.Text = sqlldr[3].ToString();
            txtGoodsNum.Text = sqlldr[5].ToString();
            cmbGoodsUnit.Text = sqlldr[6].ToString();
            txtGoodsJhPrice.Text = sqlldr[7].ToString();
            txtGoodsNeedPrice.Text = sqlldr[9].ToString();
            txtGoodsNoPrice.Text = sqlldr[10].ToString();
            txtGoodsSellPrice.Text = sqlldr[8].ToString();
            txtGoodsRemark.Text = sqlldr[11].ToString();
            txtJhCompName.Text = sqlldr[2].ToString();
            txtGoodsID.Text = sqlldr[0].ToString();
            txtGoodsID.Enabled = false;
        }
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.ToString());
    }
}
```

(3) 单击“修改”按钮, 可以对进货信息进行修改, 单击“保存”按钮后, 可以将修改后的内容保存。对应的实现代码如下所示:

```
private void toolSave_Click(object sender, EventArgs e)
{
    if (getIntCount() == 1)
```

```
{
    if (intFlag == 1)
    {
        if (jhMethod.tb_JhGoodsInfoMethodAdd(jhGood) == 2)
        {
            MessageBox.Show("添加成功", "提示");
            intFlag = 0;
            jhMethod.tb_JhGoodsInfoFind("", 5, dataGridView1);
            ControlStatus();
            ClearContorl();
        }
        else
        {
            MessageBox.Show("添加失败", "提示");
            intFlag = 0;
            jhMethod.tb_JhGoodsInfoFind("", 5, dataGridView1);
            ControlStatus();
            ClearContorl();
        }
    }
    if (intFlag == 2)
    {
        if (jhMethod.tb_JhGoodsInfoMethodUpdate(jhGood) == 1)
        {
            MessageBox.Show("修改成功", "提示");
            intFlag = 0;
            jhMethod.tb_JhGoodsInfoFind("", 5, dataGridView1);
            ControlStatus();
            ClearContorl();
        }
        else
        {
            MessageBox.Show("修改失败", "提示");
            intFlag = 0;
            jhMethod.tb_JhGoodsInfoFind("", 5, dataGridView1);
            ControlStatus();
            ClearContorl();
        }
    }
    if (intFlag == 3)
    {
        if (jhMethod.tb_JhGoodsInfoMethodDelete(jhGood) == 1)
        {
            MessageBox.Show("删除成功", "提示");
            intFlag = 0;
            jhMethod.tb_JhGoodsInfoFind("", 5, dataGridView1);
            ControlStatus();
            ClearContorl();
        }
        else
        {
            MessageBox.Show("删除失败", "提示");
            intFlag = 0;
            jhMethod.tb_JhGoodsInfoFind("", 5, dataGridView1);
            ControlStatus();
            ClearContorl();
        }
    }
}
}
```



在上述实现代码中，也用到了事件处理机制。有很多专家认为，事件处理机制是 C++、Java 和 C# 的最大软肋之一。虽然 C++ 解决了对象之间的普通消息传递用成员函数的调用问题，并且成员函数调用够快，但是导致了消息发送者与接受者之间的紧耦合。在绝大多数情况下这无关痛痒，但偏偏在事件处理上需要松耦合，于是暴露出了 C++ 的问题。这样，在遇到此类问题时，需要编写额外的代码来解决问题，遗憾的是，这部分代码从原理到实现都是初学者难以理解的。

当遇到耦合问题时，其实我们可以借鉴 C 语言的做法：规定一个协议，我们把 event 数据准备好，放在一个地方，然后就不管了，事件的接受者自己去取数据，自己解析，自己决定如何处理。这样的接受者，在对象分类中，叫作主动对象(Active Object)。这两年在嵌入式系统的编程中，人们越来越发现主动对象具有种种优势。所以作者觉得主流编程领域也应该考虑一下这个问题。

主动对象的问题可能是类型不安全，但实际上，类型安全没有那么重要，没必要死抱住不放。原来在 C 语言里声明的指针(int (\*compare)();)可以指向任何返回整数的函数，不管其参数列表如何。而在 C++ 里面，这个指针只能指向一个返回整数的无参函数了，灵活性大大降低。也难怪开源程序员还是偏爱 C。虽然安全性和灵活性同等重要，但真正的高手可以用自己的技术来保障安全性，却无法接受灵活性的缺失。

(4) 定义方法 getIntCount()，功能是获取表单内的信息。在具体实现上，将通过 if 语句验证各字段输入的数据不为空。对应的实现代码如下所示：

```
public int getIntCount()
{
    int intReslut = 0;

    if (intFlag == 1)
    {
        if (txtGoodsID.Text == "")
        {
            MessageBox.Show("商品编号不能为空!");
            return intReslut;
        }
        if (txtGoodsName.Text == "")
        {
            MessageBox.Show("商品名称不能为空!");
            return intReslut;
        }
        if (txtJhCompName.Text == "")
        {
            MessageBox.Show("供应商名称不能为空!");
            return intReslut;
        }
        if (txtEmpId.Text == "")
        {
            MessageBox.Show("进货人姓名不能为空!");
            return intReslut;
        }
        if (txtGoodsNum.Text == "")
        {
            MessageBox.Show("数量不能为空!");
            return intReslut;
        }
        if (txtGoodsName.Text == "")
```

```
{
    MessageBox.Show("进货单价不能为空!");
    return intReslut;
}

if (intFlag == 2)
{
    if (txtGoodsID.Text == "")
    {
        MessageBox.Show("商品编号不能为空! ,选择要修改记录", "提示");
        return intReslut;
    }
}

if (intFlag == 3)
{
    if (txtGoodsID.Text == "")
    {
        MessageBox.Show("商品编号不能为空! ,选择要删除记录", "提示");
        return intReslut;
    }
}

jhGood.strGoodsID = txtGoodsID.Text;
jhGood.strEmpId = txtEmpId.Text;
jhGood.strJhCompName = txtGoodsName.Text;
jhGood.strDepotName = cmbDepotName.Text;
jhGood.strGoodsNum = Convert.ToInt32(txtGoodsNum.Text);
jhGood.strGoodsName = txtGoodsName.Text;
jhGood.strGoodsUnit = cmbGoodsUnit.Text;
jhGood.deGoodsJhPrice = txtGoodsJhPrice.Text;
jhGood.deGoodsNeedPrice = txtGoodsNeedPrice.Text;
jhGood.deGoodsNoPrice = txtGoodsNoPrice.Text;
jhGood.deGoodsSellPrice = txtGoodsSellPrice.Text;
jhGood.strGoodsRemark = txtGoodsRemark.Text;
jhGood.DaGoodTime = dateTimePicker1.Value;

if (intFlag != 3)
{
    jhGood.Flag = 0;
}
else
{
    jhGood.Flag = 1;
}

intReslut = 1;
return intReslut;
}
```

### 11.7.4 进货查询模块

此模块的功能是检索系统内的进货信息，在 Visual Studio 2013 中设计窗体，具体效果如图 11-22 所示。

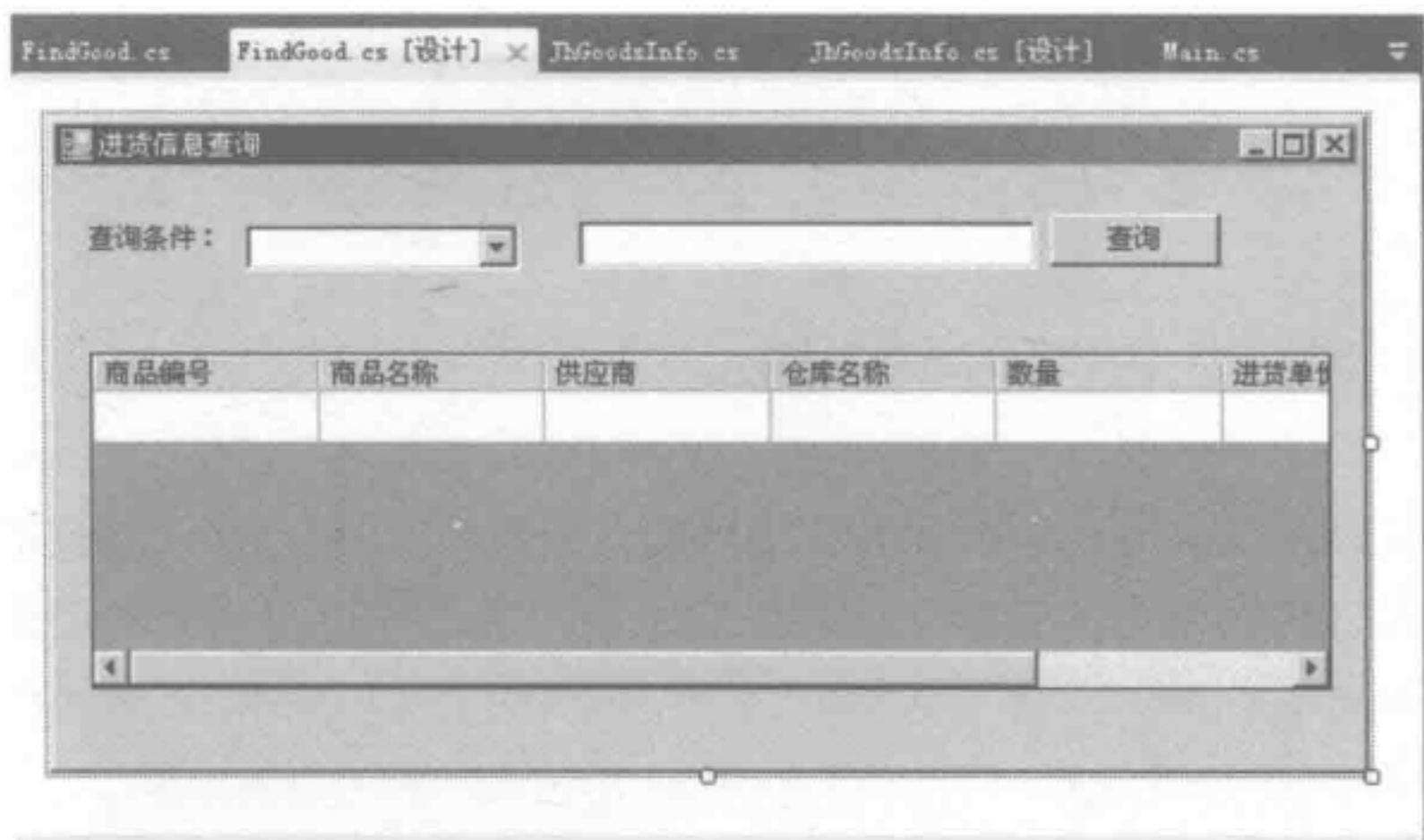


图 11-22 进货查询模块窗体的效果

此模块的实现文件是 FindGood.cs，具体实现代码如下所示：

```
namespace CHEXC
{
    public partial class FindGood : Form
    {
        public FindGood()
        {
            InitializeComponent();
        }

        tb_JhGoodsInfoMethod jhMethod = new tb_JhGoodsInfoMethod();

        private void button1_Click(object sender, EventArgs e)
        {
            if (comboBox1.Text == "")
            {
                MessageBox.Show("请选择查询条件!");
                return;
            }

            if (comboBox1.Text != "" && comboBox1.Text != "查询所有信息" && textBox1.Text == "")
            {
                MessageBox.Show("请输入查查信息");
                return;
            }

            switch (comboBox1.Text)
            {
                case "商品编号": // "商品编号":
                    jhMethod.tb_JhGoodsInfoFind(textBox1.Text, 1, dataGridView1);
                    comboBox1.SelectedIndex = 0;
                    break;
                case "商品名称": // "商品名称"
                    jhMethod.tb_JhGoodsInfoFind(textBox1.Text, 2, dataGridView1);
                    comboBox1.SelectedIndex = 0;
                    break;
                case "查询所有信息": // "所有信息":
                    jhMethod.tb_JhGoodsInfoFind(textBox1.Text, 5, dataGridView1);
                    comboBox1.SelectedIndex = 0;
                    break;
            }
        }
    }
}
```



```

    }
}

private void frmFindGood_Load(object sender, EventArgs e)
{
}
}
}

```

上述实现代码足以应付大型数据的查询功能。在现实 C# 程序中，很可能会遇到从上百万、上千万数据中筛选信息的情形，此时，经常会出现程序连接超时的错误，其中最为常见的错误是“Timeout expired. The timeout period elapsed prior to completion of the operation on the server”。针对查询超时的问题，我们提出如下所示的解决方案。

(1) 查看是否是 Connection 没关闭的问题，一般新手都会犯这个错误。

(2) 如果将 SQL 语句复制到查询分析器中执行，如果执行时间本来就超过 30 秒，那么一般采用如下所示的解决方案：首先分析引起 Timeout 的原因，一般是 Connection 没关闭或者 SqlConnection.ConnectionTimeout 超时，另外一种就是 SqlCommand.CommandTimeout 引起的，SqlCommand 的此方法为获取或设置在终止执行命令的尝试并生成错误之前的等待时间，默认为 30 秒，我们可以将其设置为 0，表示无限制。但是最好不要设置 0，否则会无限地等待下去的，只需要针对查询分析器的时间，去设置这个时间就可以了。

(3) 执行时间不是很长，但还是操作超时，也有很多原因，一般经常出现的有两种。

① ASP.NET 应用程序的请求超时，或者连接池的连接生存期结束，因为连接池默认值是 60 秒。为解决应用程序请求超时，在 web.config 中加上以下语句：

```

<system.web>
  <httpRuntime maxRequestLength="102400" executionTimeout="720" />
</system.web>

```

其中 executionTimeout 为允许执行请求的最大时间限制，单位为秒，maxRequestLength 是指示 ASP.NET 支持的最大文件上载大小。

② 解决连接池生存周期问题，在数据库连接字符串中修改为下面的形式：

```

database=AA;uid=sa;pwd=sa; Pooling=true; MAX Pool Size=1024;Min Pool Size=1;Connection
Lifetime=60

```

## 11.7.5 商品销售管理模块

商品销售信息窗体的功能，是显示系统内销售的商品信息，在 Visual Studio 2013 中设计的窗体效果如图 11-23 所示。

此模块的实现文件是 SellGoods.cs，下面开始讲解其具体实现流程。

(1) 载入后，从数据库中检索所有的商品信息，并绑定到 DataGridView 控件上。对应的代码如下所示：

```

private void frmSellGoods_Load(object sender, EventArgs e)
{
    sellMethod.tb_SellGoodsFind(dataGridView1);
}

```

图 11-23 商品销售信息窗体的效果

(2) 单击 dataGridView1 后, 可查看此条数据的信息详情。对应的代码如下所示:

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (intCount == 2 || intCount == 3)
    {
        FillControls();
    }
}
```

在上述代码中, 通过调用方法 FillControls(), 将指定编号的商品详情信息显示出来。对应的代码如下所示:

```
private void FillControls()
{
    try
    {
        SqlDataReader sqlldr = sellMethod.dtb_SellGoodsFind(this.dataGridView1[0,
            this.dataGridView1.CurrentCell.RowIndex].Value.ToString());
        sqlldr.Read();
        if (sqlldr.HasRows)
        {
            txtSellID.Text = sqlldr[0].ToString();
            txtSellID.Enabled = false;
            txtEmpID.Text = sqlldr[3].ToString();
            txtGoodsName.Text = sqlldr[4].ToString();
            txtSellGoodsNum.Text = sqlldr[5].ToString();
            DaSellGoodsTime.Value = Convert.ToDateTime(sqlldr[6].ToString());
            txtSellRemark.Text = sqlldr[10].ToString();
            txtdeSellPrice.Text = sqlldr[7].ToString();
            txSellNeedPay.Text = sqlldr[8].ToString();
            txtdeSellHasPay.Text = sqlldr[9].ToString();
        }
        sqlldr.Close();
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.ToString());
    }
}
```

(3) 定义处理事件 toolSave\_Click, 当对商品数据进行修改后, 单击“保存”按钮, 会完成对新数据的添加或修改操作。对应的代码如下所示:

```
//保存
private void toolSave_Click(object sender, EventArgs e)
{
    if (fillGetInfo() == 1)
    {
        if (intCount == 1)
        {
            if (sellMethod.tb_SellGoodsAdd(sellGoods) == 1)
            {
                MessageBox.Show("添加成功");
                Clear();
                ControlStatus();
                intCount = 0; //添加标记
                sellMethod.tb_SellGoodsFind(dataGridView1);
            }
            else
            {
                MessageBox.Show("添加失败");
                Clear();
                ControlStatus();
                intCount = 0; //添加标记
            }
        }
        if (intCount == 2)
        {
            if (sellMethod.tb_SellGoodsUpdate(sellGoods) == 1)
            {
                MessageBox.Show("修改成功");
                Clear();
                ControlStatus();
                intCount = 0; //添加标记
                sellMethod.tb_SellGoodsFind(dataGridView1);
            }
            else
            {
                MessageBox.Show("修改失败");
                Clear();
                ControlStatus();
                intCount = 0; //添加标记
            }
        }
        if (intCount == 3)
        {
            if (sellMethod.tb_SellGoodsDelete(sellGoods) == 1)
            {
                MessageBox.Show("删除成功");
                Clear();
                ControlStatus();
                intCount = 0; //添加标记
                sellMethod.tb_SellGoodsFind(dataGridView1);
            }
            else
            {
                MessageBox.Show("删除失败");
                Clear();
                ControlStatus();
            }
        }
    }
}
```



```
        intCount = 0; //添加标记
    }
}
}
```

### 11.7.6 退货管理模块

此模块的功能是实现对系统内退货信息的管理操作，在 Visual Studio 2013 中设计窗体，效果如图 11-24 所示。



图 11-24 退货管理窗体的效果

此模块的实现文件是 ThGoodsInfo.cs，下面开始讲解其具体实现流程。

(1) 对输入的数据类型进行验证处理，退货数量、退货单价、实付金额、应付金额这 4 个文本框内不能随意输入数据。对应的代码如下所示：

```
private void txThGoodsNum_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar!=8 && !char.IsDigit(e.KeyChar))
    {
        MessageBox.Show("输入数字");
        e.Handled = true;
    }
}

private void txtThGoodsPrice_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar!=8 && !char.IsDigit(e.KeyChar)&&e.KeyChar!='.')
    {
        MessageBox.Show("输入数字");
        e.Handled = true;
    }
}

private void txtThNeedPay_KeyPress(object sender, KeyPressEventArgs e)
{

```

```

        if (e.KeyChar!=8 && !char.IsDigit(e.KeyChar) && e.KeyChar!='.')
        {
            MessageBox.Show("输入数字");
            e.Handled = true;
        }
    }

private void txtThHasPay_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar!=8 && !char.IsDigit(e.KeyChar))
    {
        MessageBox.Show("输入数字");
        e.Handled = true;
    }
}

```

(2) 定义处理事件 toolSave\_Click, 实现数据添加和修改处理, 当单击“保存”按钮后, 完成对数据的修改和添加处理。对应的代码如下所示:

```

private void toolSave_Click(object sender, EventArgs e)
{
    if (retuCount() == 1)
    {
        if (intCoun == 1)
        {
            if (tbMenddd.tb_ThGoodsAdd(tbGoodinfo) == 1)
            {
                MessageBox.Show("添加成功");
                ControlStatus();
                getClear();
                tbMenddd.tb_ThGoodsFind(dataGridView1);
                intCoun = 0; //i 添加标记
            }
            else
            {
                MessageBox.Show("添加失败");
                ControlStatus();
                getClear();
                tbMenddd.tb_ThGoodsFind(dataGridView1);
                intCoun = 0; //i 添加标记
            }
        }
        if (intCoun == 2)
        {
            if (tbMenddd.tb_ThGoodsUpdate(tbGoodinfo) == 1)
            {
                MessageBox.Show("修改成功");
                ControlStatus();
                getClear();
                tbMenddd.tb_ThGoodsFind(dataGridView1);
                intCoun = 0; //i 添加标记
            }
            else
            {
                MessageBox.Show("修改失败");
                ControlStatus();
                getClear();
                tbMenddd.tb_ThGoodsFind(dataGridView1);
                intCoun = 0; //i 添加标记
            }
        }
    }
}

```

```

    }
}
if (intCoun == 3)
{
    if (tbMenddd.tb_ThGoodsDelete(txtThGoodsID.Text) == 1)
    {
        MessageBox.Show("删除成功");
        ControlStatus();
        getClear();
        tbMenddd.tb_ThGoodsFind(dataGridView1);
        intCoun = 0; //i 添加标记
    }
    else
    {
        MessageBox.Show("删除失败");
        ControlStatus();
        getClear();
        tbMenddd.tb_ThGoodsFind(dataGridView1);
        intCoun = 0; //i 添加标记
    }
}
}
}

```

### 11.7.7 库存管理模块

此模块的功能是实现对系统内库存信息的管理操作，在 Visual Studio 2013 中设计窗体，效果如图 11-25 所示。

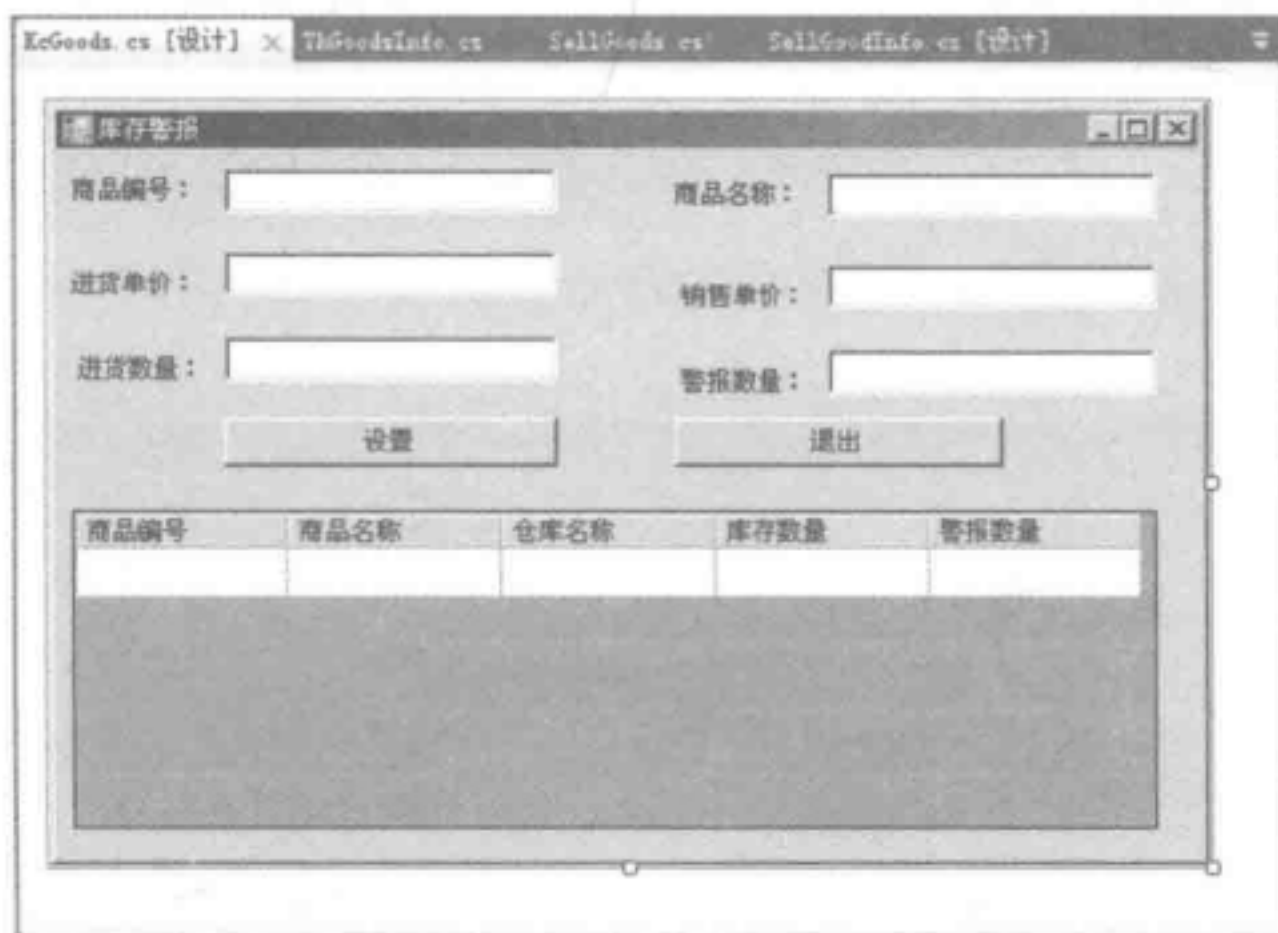


图 11-25 库存管理窗体的效果

此模块的实现文件是 KcGoods.cs，下面开始讲解其具体实现流程。

(1) 载入时，检索出数据库中的库存信息并绑定到 DataGridView 控件，对应的代码如下所示：

```

private void frmKcGoods_Load(object sender, EventArgs e)
{
    tb_GoodMenthdd.tb_ThGoodsFind(dataGridView1, 4, kcGood);
}

```



(2) 当单击 DataGridView 控件中的某条退货信息时, 将显示其详情信息, 对应的代码如下所示:

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    FillControls();
}
```

上述功能是通过调用方法 FillControls() 实现的, 对应的实现代码如下所示:

```
private void FillControls()
{
    try
    {
        SqlDataReader sqlldr = tb_GoodMenthd.tb_ThGoodsFind(this.dataGridView1[0,
            this.dataGridView1.CurrentRow.Index].Value.ToString());

        sqlldr.Read();
        if (sqlldr.HasRows)
        {
            txtid.Text = sqlldr[1].ToString();
            txtGoodsName.Text = sqlldr[2].ToString();
            txtGoodsJhPrice.Text = sqlldr[9].ToString();
            txtGoodsSellPrice.Text = sqlldr[10].ToString();
            txtGoodsNum.Text = sqlldr[5].ToString();
        }
        sqlldr.Close();
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.ToString());
    }
}
```

(3) 定义 btnAdd\_Click 处理事件, 可以对库存数量进行修改, 修改后, 单击“添加”按钮后, 实现对库存数量的修改。对应的代码如下所示:

```
private void btnAdd_Click(object sender, EventArgs e)
{
    if (txtid.Text == "")
    {
        MessageBox.Show("请选择商品信息");
        return;
    }
    if (txtnum.Text == "")
    {
        MessageBox.Show("请输入商品警报数量");
        return;
    }
    int intResult =
        tb_GoodMenthd.tb_KcGoodsUpdate(txtid.Text, Convert.ToInt32(txtnum.Text));
    if (intResult == 1)
    {
        MessageBox.Show("添加成功!");
        tb_GoodMenthd.tb_ThGoodsFind(dataGridView1, 4, kcGood);
        ClearFill();
    }
    else
    {
        MessageBox.Show("添加失败!");
    }
}
```

```

        ClearFill();
    }
}

```

### 11.7.8 库存查询模块

库存查询模块的功能是快速检索出某个商品的库存信息，在 Visual Studio 2013 中设计窗体，效果如图 11-26 所示。

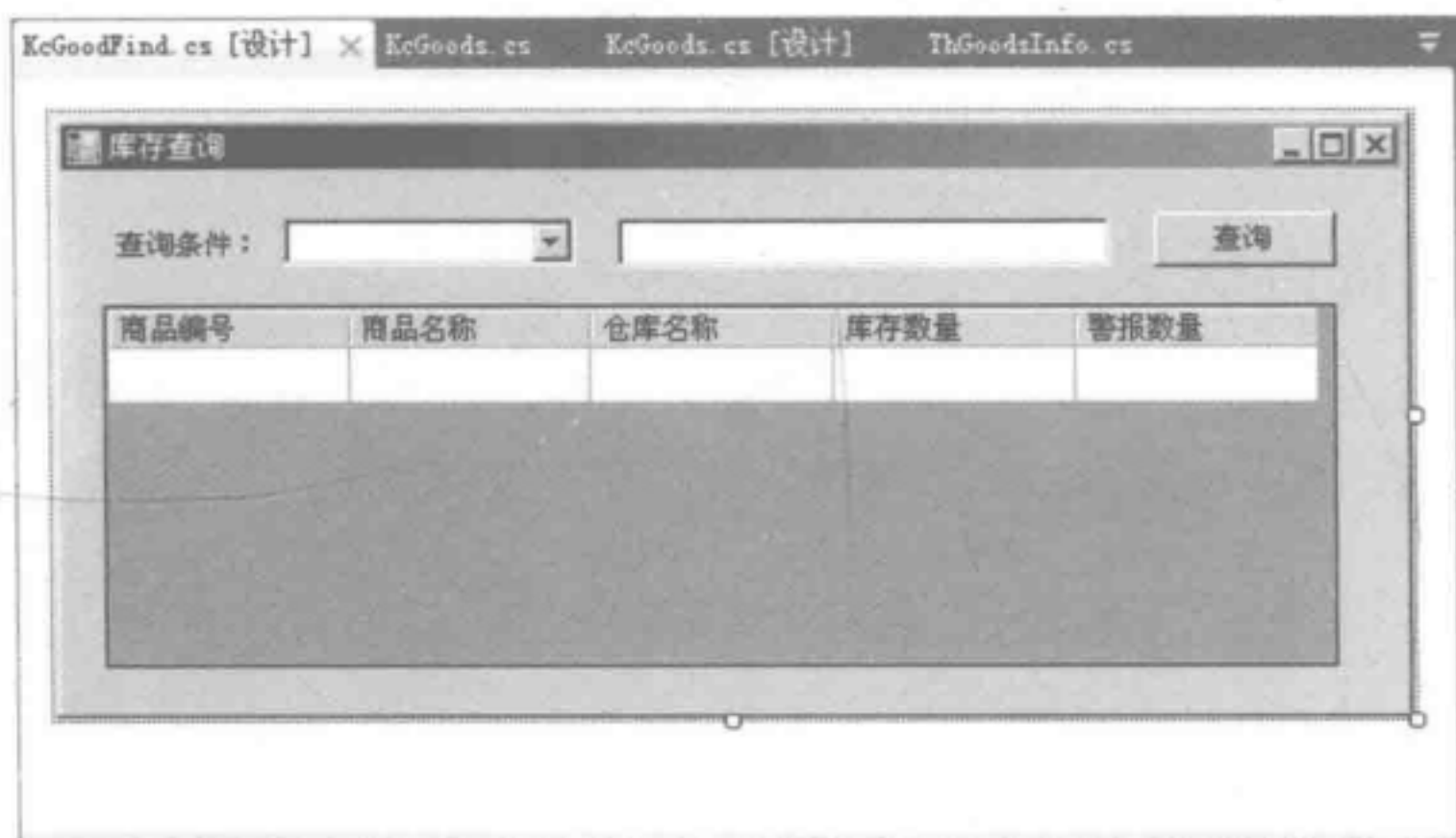


图 11-26 库存查询窗体的效果

此模块的实现文件是 KcGoodFind.cs，具体实现代码如下所示：

```

namespace CHEXC
{
    public partial class KcGoodFind : Form
    {
        public KcGoodFind()
        {
            InitializeComponent();

            tb_KcGoodsMethod tb_GoodMenthd = new tb_KcGoodsMethod();
            tb_KcGoods kcgood = new tb_KcGoods();

            private void button1_Click(object sender, EventArgs e)
            {
                if (comboBox1.Text == "")
                {
                    MessageBox.Show("请选择查询条件!");
                    return;
                }
                if (txtkey.Text == "")
                {
                    MessageBox.Show("请输入查询信息");
                    return;
                }
                switch (comboBox1.Text)
                {
                    case "商品编号": //"商品编号":
                        kcgood.strGoodsID = txtkey.Text;
                        tb_GoodMenthd.tb_ThGoodsFind(dataGridView1, 1, kcgood);

```

```

        break;
    case "商品名称": //商品名称
        kcgood.strKcGoodsName = txtkey.Text;
        tb_GoodMenthd.tb_ThGoodsFind(dataGridView1, 2, kcgood);
        break;
    }
}

private void frmKcGoodFind_Load(object sender, EventArgs e)
{
}
}

```

## 11.7.9 数据备份模块

此模块的功能，是实现对数据库信息的备份处理，在 Visual Studio 2013 中设计窗体，效果如图 11-27 所示。

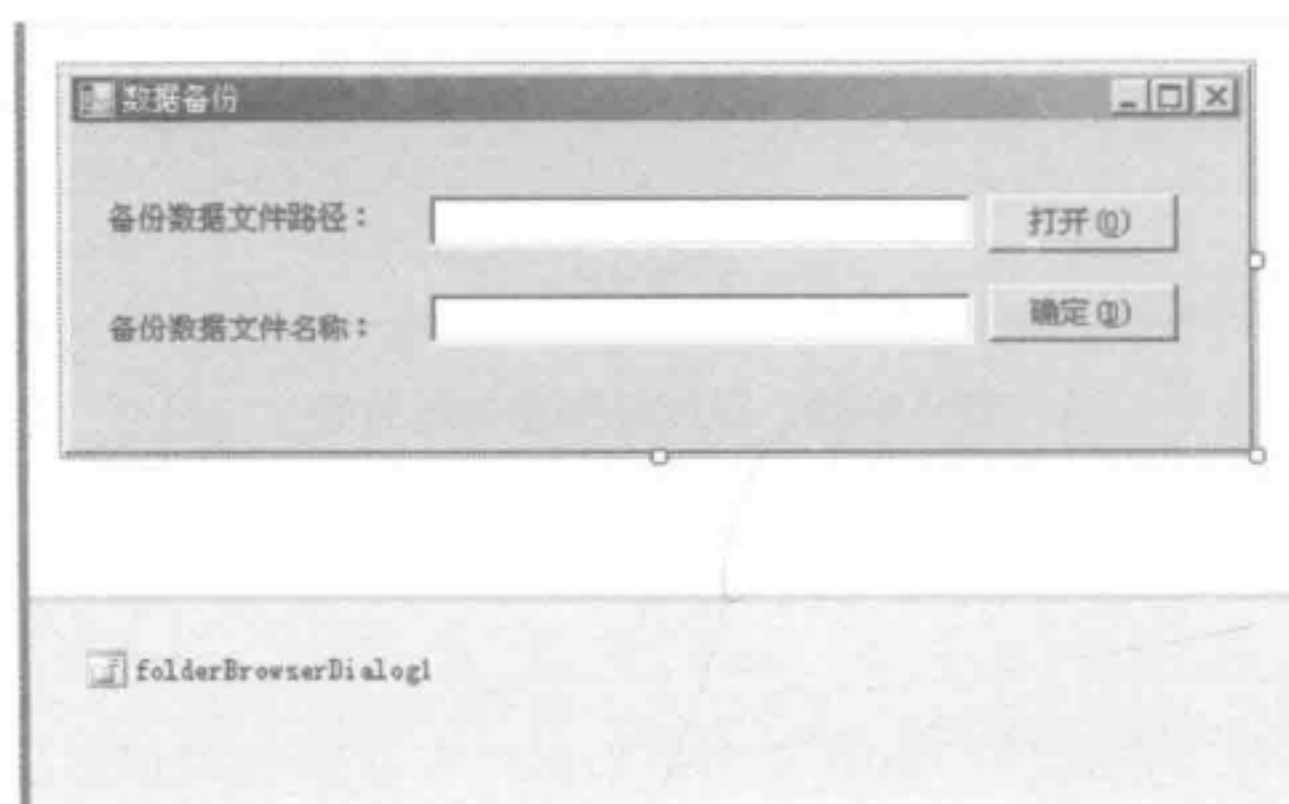


图 11-27 数据备份窗体的效果

此模块的实现文件是 DataBack.cs，具体的实现代码如下所示：

```

public partial class DataBack : Form
{
    public DataBack()
    {
        InitializeComponent();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
        {
            txtPath.Text = folderBrowserDialog1.SelectedPath.ToString();
        }
    }

    private void frmDataBack_Load(object sender, EventArgs e)
    {
    }

    private void button1_Click(object sender, EventArgs e)
    {
    }
}

```



```

try
{
    if (txtPath.Text != "" && txtName122.Text != "")
    {
        getSqlConnection geCon = new getSqlConnection();
        SqlConnection con = geCon.GetCon();

        string strBac1 = "backup database db_CSManage to disk='"
            + txtPath.Text.Trim() + "\\\" + txtName.Text.Trim() + ".bak'";
        SqlCommand Cmd = new SqlCommand(strBac1, con);
        if (Cmd.ExecuteNonQuery() != 0)
        {
            MessageBox.Show("数据备份成功!", "提示框", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            this.Close();
        }
        else
        {
            MessageBox.Show("数据备份失败!", "提示框", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
    }
    else
    {
        MessageBox.Show("请填写备份的正确位置及文件名!", "提示框",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
} // end
}
catch (Exception ee)
{
    MessageBox.Show(ee.Message.ToString());
}
}
}

```

到此为止，整个项目的编码工作已经结束。在此阶段，深刻体会到数据库是软件项目的核心，保存的数据通常是企业的战略核心机密，所以数据库的安全变得十分重要，除了传统意义的安全外，操作人员的失误也可能会误删数据库，所以数据库备份和还原就成了一个必需的功能。

(1) 下面是作者收集的数据库备份通用代码：

```

SqlConnection conn =
    new SqlConnection("Server=.;Database=master;User ID=sa;Password=sa;");

SqlCommand cmdBK = new SqlCommand();
cmdBK.CommandType = CommandType.Text;
cmdBK.Connection = conn;
cmdBK.CommandText = @"backup database test to disk='C:\ba' with init";

try {
    conn.Open();
    cmdBK.ExecuteNonQuery();
    MessageBox.Show("Backup succeeded.");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

```

```
finally
{
    conn.Close();
    conn.Dispose();
}
```

(2) 下面是作者收集的数据库还原通用代码:

```
SqlConnection conn = new SqlConnection(
    "Server=.;Database=master;User ID=sa;Password=sa;Trusted_Connection=False");

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT spid FROM sysprocesses, sysdatabases
    WHERE sysprocesses.dbid=sysdatabases.dbid AND sysdatabases.Name='test'", conn);

SqlDataReader dr;
dr = cmd.ExecuteReader();
ArrayList list = new ArrayList();

while(dr.Read())
{
    list.Add(dr.GetInt16(0));
}

dr.Close();

for(int i=0; i<list.Count; i++)
{
    cmd = new SqlCommand(string.Format("KILL {0}", list), conn);
    cmd.ExecuteNonQuery();
}

SqlCommand cmdRT = new SqlCommand();
cmdRT.CommandType = CommandType.Text;
cmdRT.Connection = conn;
cmdRT.CommandText = @"restore database test from disk='C:\ba'";

try
{
    cmdRT.ExecuteNonQuery();
    MessageBox.Show("Restore succeeded.");
}
catch(Exception ex)
{
    MessageBox.Show(ex.Message);
}
finally
{
    conn.Close();
}
```

## 11.8 项目调试

视频讲解  光盘: 视频\第 11 章\项目调试.avi

将项目命名为“supermarket”，编译运行后，首先显示登录界面，如图 11-28 所示。

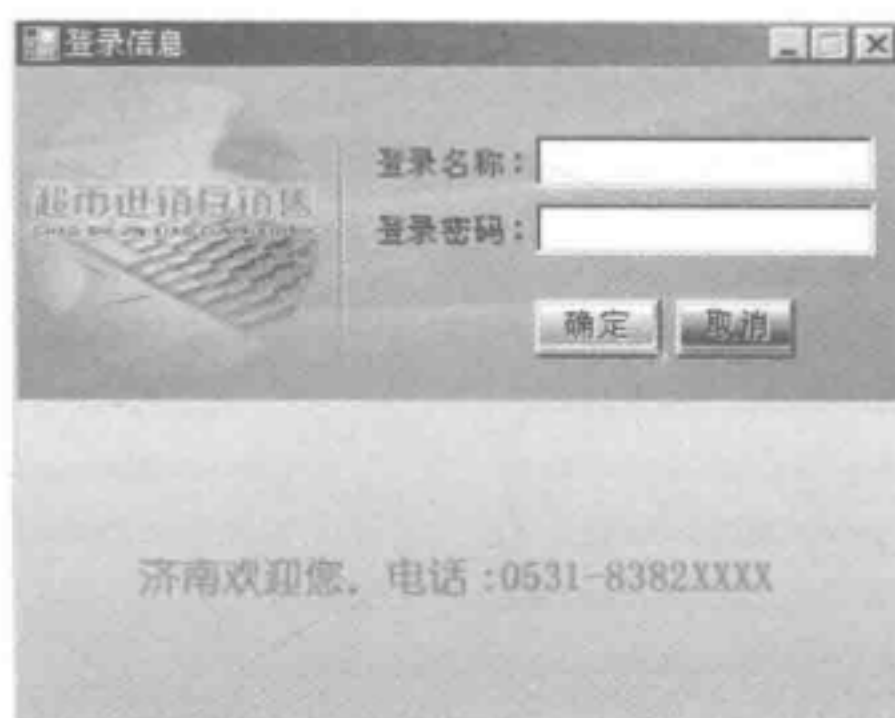


图 11-28 登录界面

登录后的主界面效果如图 11-29 所示。



图 11-29 登录后的主界面

员工信息界面如图 11-30 所示。

员工基本信息

保存

取消

添加

修改

删除

查询条件:

查找

退出

员工姓名:

出生日期:

2015年 1月18日

性别:

系统登录名

登录密码:

员工职位:

家庭电话:

手机号码:

所属部门:

家庭地址:

| 员工编号              | 员工姓名 | 性别 | 所在部门 | 员工职位 | 联系电话     |
|-------------------|------|----|------|------|----------|
| YG-12122007154151 | XXX  | 男  | 食品部  | 组长   | 8988**** |
| YG-13122007090625 | YYY  | 女  | 食品部  | 组长   | 555***   |

图 11-30 员工信息界面



进货信息界面如图 11-31 所示。

商品进货信息

保存 取消 添加 修改 删除 退出

进货编号:

商品名称:

供应商名称:

进货数量:

进货单价:

销售单价:

进货日期:

进货员工:

所属仓库:

应付金额:

实付金额:

备注:

| 商品编号 | 商品名称 | 供应商 | 仓库名称 | 数量 | 进货单价 |
|------|------|-----|------|----|------|
|------|------|-----|------|----|------|

图 11-31 进货信息界面

商品销售界面如图 11-32 所示。

商品销售信息

保存 取消 添加 修改 删除 退出

销售编号:

商品名称:

时间:

销售单价:

销售数量:

员工:

应付金额:

实付金额:

备注:

| 销售编号         | 商品名称 | 销售单价 | 销售数量 | 销售员工 | 销售日期        |
|--------------|------|------|------|------|-------------|
| XS-190920101 | 5555 | 34   | 12   | bb   | 2015年 1月18日 |

图 11-32 商品销售界面

库存查询界面如图 11-33 所示。

库存查询

查询条件:   查询

| 商品编号 | 商品名称 | 仓库名称 | 库存数量 | 警戒数量 |
|------|------|------|------|------|
|------|------|------|------|------|

图 11-33 库存查询界面

## 第 12 章 家庭视频监控系统

高帆项目开发

社会在进步，技术在发展。随着人们生活水平的提高，公民的安全意识日益提高。如今很多小区、道路等场所都安装了视频监控系统。

在本章的内容中，将通过“C# + 视频采集卡”，制作一个简单的家庭视频监控系统，并把系统划分为几个模块进行详细讲解。通过对这些模块的学习，读者完全可以掌握视频监控系统的开发技术及原理。

### 赠送的超值电子书

- 111 隐藏
- 112 虚方法和虚方法重写
- 113 重写方法的特点
- 114 多态的应用
- 115 使用 base
- 116 抽象类和抽象方法
- 117 密封类和密封成员
- 118 静态类
- 119 定义接口
- 120 接口的实现和继承

## 12.1 走向架构师之路

视频讲解 光盘：视频\第12章\走向架构师之路.avi

程序员的水平有高有低，职位也不尽相同。如果你想立志向技术方面发展，那么，在众多的职位中，究竟有什么职位值得大家羡慕呢？答案是架构师。

### 12.1.1 什么是架构师

架构师是软件开发行业中的一种新兴职业，工作职责是在一个软件项目开发过程中，将客户的需求转换为规范的开发计划及文本，并制定这个项目的总体架构，指导整个开发团队完成这个计划。架构师的主要任务不是从事具体的软件程序编写的工作，而是从事更高层次的开发架构工作。架构师必须对开发技术非常了解，并且需要有良好的组织管理能力。可以这样说，一个架构师工作的好坏，决定了整个软件开发项目的成败。

架构师实际上就是软件的总体设计师，又称项目首席设计师。架构师一定是在工程实践中成长起来的，而并非上了几次培训班，读了几本书就可以成为架构师。

架构师是客户需求和开发者之间的桥梁。在软件行业中，一般提到的架构师是技术架构师，而忽略了领域架构师或者领域工程师的概念。一个好的领域专家一定是业务领域的架构师，他能够给出某一个业务领域的架构，即业务架构，只有技术架构和业务架构紧密结合，才有可能真正创造出一个好的系统。软件架构师在整个软件开发过程中都起着重要的作用，并随着开发进程的推进，其职责或关注点也在不断地变化。在需求阶段，软件架构师主要负责理解和管理非功能性的系统需求，比如软件的可维护性、性能、复用性、可靠性、有效性和可测试性等，此外，架构师还要经常审查客户及市场人员所提出的需求，确认开发团队所提出的设计；在需求越来越明确后，架构师的关注点开始转移到组织参与开发的团队成员和开发过程的定义上；在软件设计阶段，架构师负责对整个软件体系结构、关键构件、接口和开发策略的设计；在编码阶段，架构师则成为详细设计者和代码编写者的顾问，并且经常性地要举行一些技术研讨会、技术培训班等；随着软件开始测试、集成和交付，集成和测试支持将成为软件架构师的工作重点；在软件维护开始时，软件架构师就要开始为下一版本的产品是否应该增加新的功能模块而进行决策了。

### 12.1.2 赢在架构——如何成为一名架构师

在软件开发过程中，一个优秀软件架构师的重要性是不应低估的。那么，究竟如何成为优秀的软件架构师呢？应当注意如下几点：

- 必须具有丰富的软件设计和开发经验。这有助于理解并解释所进行的设计是如何映射到实现中去的。
- 要具有领导能力和团队协作技能。软件架构师必须是一个得到承认的技术领导，能在关键的时候对技术的选择做出及时、有效的决定。
- 具有很强的沟通能力。其实，这一点好像无论什么角色最好都能具备。软件架构



师需要与各路人马经常打交道,例如与客户、市场人员、开发人员、测试人员、项目经理、网络管理员、数据库工程师等,而且,在很多角色之间,还要起沟通者的作用。

- 需要时刻注意新软件设计和开发方面的发展情况,并不断探索更有效的新方法。开发语言、设计模式和开发平台在不断地升级,软件架构师需要吸收这些新技术、新知识,并将它们用于软件系统开发工作中。
- 具备行业的业务知识。这有助于设计出一个能满足客户需求的体系结构。优秀的软件架构师常常因为要尽快获得对行业业务的理解,而必须快速学习并且敏锐地进行观察。

### 12.1.3 赢在架构——如何成就一个美丽的架构

无论是一个大规模的电信网络管理系统、大规模应用的互联网架构还是企业级的 ERP 系统,很多时候,不可能一开始就设计出最完美的解决方案。系统应该随着规模的变化,不断演进。这样的系统才是科学的、经济的。

架构之美体现在关注点的分离和结合。在软件设计中,我们需要考虑多方面的关注点,漂亮的架构就是让这些关注点尽可能分离,然后以最简单的方式结合,从而得到高内聚、低耦合的系统。

(1) 在现实应用中,一般有如下三种评估架构的方式。

① 通过建模或者模拟系统的一个或者多个方面来确定架构属性。

例如,通过性能建模,来评估系统的吞吐量和伸缩性;通过失效树模型,来评估系统的可靠性和可访问性;还有复杂性和耦合性指标,可用于评估可变性和可维护性。

② 通过对架构师的质询来评估架构。

有许多结构化的质询方法,通过组织内的专家或者一些领域专家的质询来评估架构。

③ 架构折中分析法。

这是质询评估法的变体,它通过寻找架构中不能满足品质关注点的风险来评估架构。使用特定的场景来分析,每种场景描述特定利益相关人对系统的品质关注点。然后由架构师来解释如何支持每一种场景。

(2) 要想成就一个美丽的架构,需要做到如下 4 点。

① 要明确系统的关注点。

要明确系统需要考虑哪些关注点,哪些关注点是需要重点关注的关注点。要知道,没有哪个系统能够完美地满足所有关注点,因为,对其中一个关注点的完美满足,就是对另外一个关注点的不完美满足,所以,架构是一种折中,一种针对特定系统的重要关注点的折中满足。所以,发现特定系统中的重要关注点,以及满足这些关注点的条件,是我们取得架构的方法。

另外,项目中的不同群体对系统有不同的关注点,具体说明如下所示。

- 投资人:关注的是项目是否可以在给定的资源和进度约束下完成。
- 架构师、开发人员、测试人员:关注的是系统最初的构建和以后的维护、演进。
- 项目经理:关注的是如何组织团队,指定迭代计划。
- 客户:关心的是所有关注点是否得到了合适的满足。

与相关利益群体沟通、明确这些关注点和约束，并排列优先级。

② 制定一组要遵循的规则。

这组规则有助于消除复杂性，并可以用于指导详细设计和系统验证。设计规则可能表现为特定的抽象，这些抽象总是以同样的方式使用；设计规则还表现为一种模式，如管道模式和过滤器模式，在系统中处处使用相同的设计原则，让设计概念具备完整性；一个好的架构反映的是一组设计思想，而不是很多“好的思想”，这些思想之间却彼此独立不协调；设计规则还体现在符合法规和安全性的要求。


③ 确保设计概念在实现时得到一致体现。

④ 好的架构来自于更好的架构师提供的现场指导，原因在于，一些关注点是很多系统的共性。

- 功能性：产品向其用户提供哪些功能。
- 可变性：软件将来可能要做哪些改变，哪些变化不可能发生，不需要特别容易做这些改变。
- 性能：产品将达到怎样的性能。
- 容量：多少用户将并发使用该系统？该系统为多少用户保存数据？
- 生态系统：在部署的生态环境中，该系统与其他系统进行哪些交互？
- 模块化：如何将编写软件的任务分解为工作指派？特别是，这些模块如何进行独立开发，并能够准确而容易地满足彼此的需要？
- 可构建性：如何将软件构建为一组组件，并能够独立实现和验证这些组件？哪些组件应该复用其他的产品？哪些应该从外部供应商处获得？
- 产品化：如果产品将以几种变体的形式存在，如何开发一个产品线并利用这些变体的共性？产品线中的产品以怎样的步骤开发？是否可以开发最小的产品，然后再添加、扩展组件？能否在不改变以前编写的代码的情况下，开发产品线的其他成员？
- 安全性：产品是否需要用户认证？数据的安全性如何得到保障？如何抵挡外来的攻击？

最后要说的是，架构之路并不平坦，需要我们不断探索、不断实践。在成长的道路上，每一滴汗水都是自己成长的记号。

## 12.2 新的项目

视频讲解  光盘：视频\第 12 章\新的项目.avi

本视频监控系统的开发团队成员的具体说明如下所示。

- 软件工程师 A：负责公共类的设计和具体编码。
- 软件工程师 B：负责项目分析、系统设计和数据库搭建。
- 软件工程师 C：系统后期调试。

本项目的具体实现流程如图 12-1 所示。



图 12-1 开发流程

## 12.3 需求分析

视频讲解 光盘：视频\第 12 章\需求分析.avi

所谓“需求分析”，是指对要解决的问题进行详细的分析，弄清楚问题的要求，包括需要输入什么数据，要得到什么结果，最后应输出什么。可以说，软件工程中的“需求分析”就是确定要计算机“做什么”，要达到什么样的效果。

### 12.3.1 系统背景介绍

XX 家庭在逐渐加快的生活节奏中，为了适应不断增加的工作压力，提高安全意识，现委托我公司开发一个视频监控系统。该系统的主要作用是：当用户外出时，可以使用该系统监控家里的各种情况，并可以将家里的变化情况录制成视频文件，以供后期查看。

### 12.3.2 系统需求

随着生活节奏的加快，工作压力的增加，人们用于照看家庭的时间会越来越少。年幼孩子的看护、年迈父母的照看、家庭财产的防窃等一系列问题经常缠绕着人们，成为人们忙碌中挥之不去的牵挂。这里，我们使用 C#语言结合视频采集卡，制作一个简单的家庭视频监控系统，以解决上面列出的各种问题。

### 12.3.3 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，制定可行性研究报告如下。

#### 1. 引言

##### (1) 编写目的

为了给企业的决策层提供是否进行项目实施的参考依据，现以文件的形式分析项目的



风险、项目需要的投资和效益。

## (2) 背景

XX 家庭为了在外出时可以监控家里的各种情况，现需要委托其他公司开发一个视频监控系统，项目名称为“家庭视频监控系统”。

## 2. 可行性研究的前提

### (1) 要求

家庭视频监控系统要求能够提供视频监控、快照、录像和自动监控等功能。

### (2) 目标

家庭视频监控系统的主要目标是保证家里的安全。

### (3) 条件、假定和限制

项目需要在 40 天内交付用户使用，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 5 天时间确认需求分析。那么程序开发人员需要在 35 天的时间内进行系统设计、程序编码、系统测试、程序调试和系统打包部署工作，其间，还包括了员工每周的休息时间。

## 3. 投资及我们的收益分析

### (1) 支出

根据系统的规模及项目的开发周期(一个月)，公司决定投入 3 个人。此外，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。

### (2) 我们的收益

用户提供项目资金 10 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 4 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

根据前面的分析，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 3 个人、一个月的时间，获利 4 万元，比较可观；在公司今后发展上可以储备软件开发的经验和资源，因此认为该项目可以开发。

## 12.3.4 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

#### (1) 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

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统一约定，以及项目生命周期内的所有项目活动的行动基础。

(2) 背景

家庭视频监控系统是由 XX 家庭委托我公司开发的小型视频监控系统，系统主要用于监控家里的人员活动情况，项目周期为一个月。项目的背景规划如表 12-1 所示。

表 12-1 项目的背景规划

| 项 目 名 称  | 项目委托实施单位 | 任务提出者 | 项目承担部门       |
|----------|----------|-------|--------------|
| 家庭视频监控系统 | XXX 软件公司 | DP    | 研发部门<br>测试部门 |

2. 概述

(1) 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。家庭视频监控系统的项目目标如下。

家庭视频监控系统的主要目的，是随时对家里的情况进行监控，并可以由用户灵活控制监控方向，另外，用户还可以设置自动监控、对监控画面进行快照和录像等。

(2) 应交付成果

项目开发完成后，交付的内容如下：

- 以光盘的形式提供家庭视频监控的源程序、系统数据库文件和系统使用说明书。
- 系统发布后，进行无偿维护和服务 6 个月，超过 6 个月进行系统有偿维护与服务。

(3) 项目开发环境

开发本项目所用的操作系统可以是 Windows 2000 Server、Windows XP、Windows Server 2003、Windows Vista、Windows 7，开发工具为 Visual Studio 2010 + 视频采集卡，数据库采用 Microsoft Access 2010。

(4) 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，交给用户进行外部验收，验收的主要依据为需求规格说明书。

3. 项目的团队组织

(1) 组织结构

为了完成家庭视频监控系统的开发，公司组建一个临时的项目团队，大体上由项目经理、软件工程师和测试人员构成。

(2) 人员分工

明确项目团队中每个人的任务：A 负责公共类的设计和具体编码；B 负责项目分析、系统设计和数据库搭建；C 负责系统后期调试。

## 12.4 系统设计

视频讲解 光盘：视频\第12章\系统设计.avi

本系统属于小型的家庭视频监控系统，目标是对指定的区域进行实时监控。根据系统功能分析，并结合单位实际情况，可以很快地完成系统设计书面文件的编写工作。

本系统的具体实现目标如下：

- 系统采用人机交互的方式，界面美观友好，视频监控灵活、方便。
- 灵活控制云台，以监控某一区域的各个角落。
- 适时对监控画面进行快照和录像操作。
- 选择观看已经录制的视频文件。
- 完备的系统注册功能。
- 系统可最大限度地实现易维护性和易操作性。

系统的功能结构如图 12-2 所示。

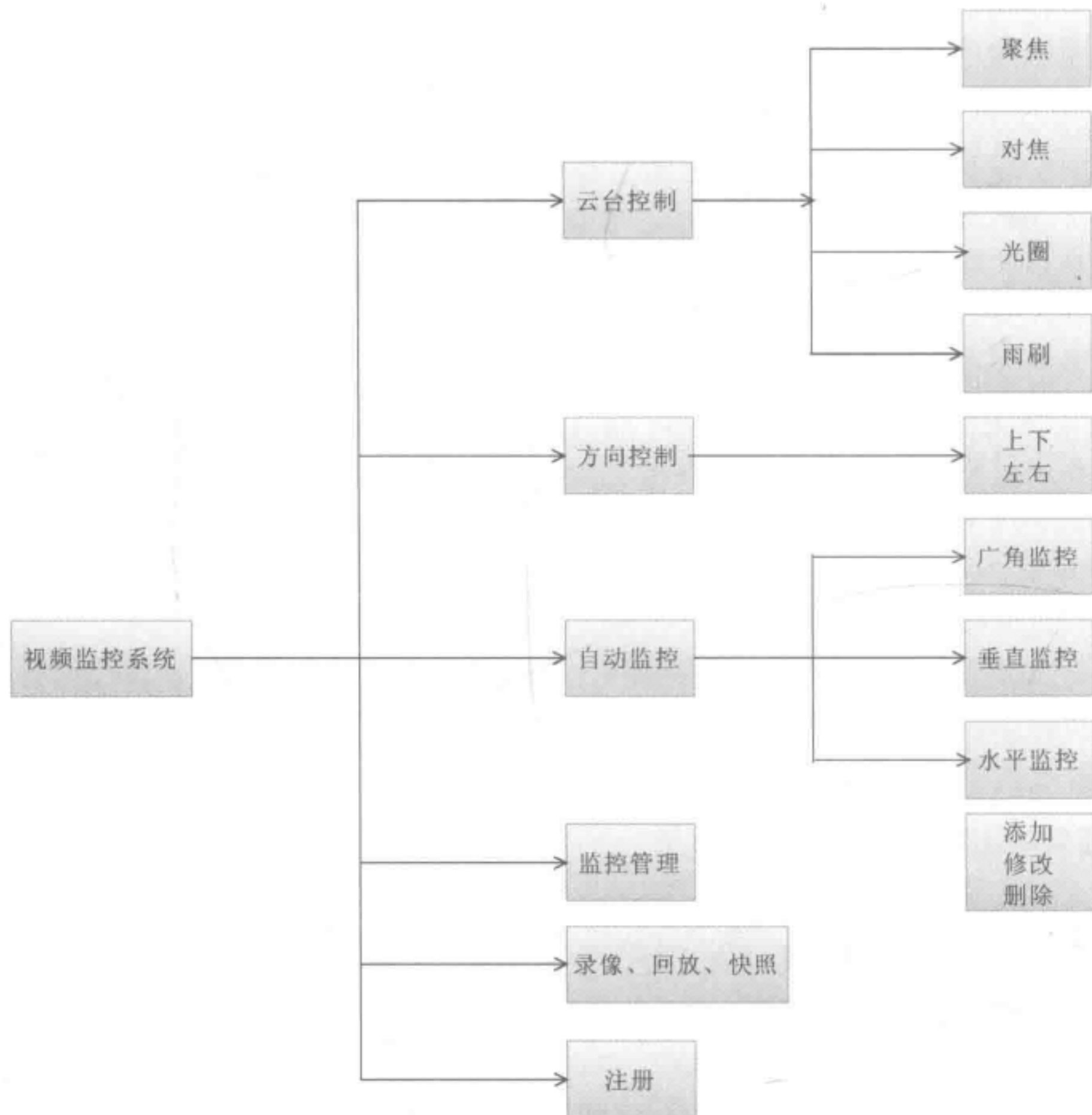


图 12-2 系统的功能结构



## 12.5 数据库设计

视频讲解 光盘：视频\第 12 章\数据库设计.avi

一个成功的软件系统，是由 50% 的业务加 50% 的软件所组成的，而 50% 的成功软件又是由 25% 的数据库加 25% 的程序所组成的，数据库设计得好坏是一个关键。如果把企业的数据比作生命所必需的血液，那么数据库的设计就是应用中的心脏部分。

### 12.5.1 数据库分析

在家庭视频监控系统中，因为系统的信息量不是很大，并且系统的项目成本不高，数据库主要用来存储用户登录系统的名字和密码，因此，对数据库的要求并不是很高，所以本系统采用 Microsoft Access 作为后台数据库，数据库命名为 VWMS，其中包含一张数据表，用于存储用户登录信息。

### 12.5.2 数据库的概念设计

系统开发过程中，数据库设计占有重要的地位，数据库设计的原则，是根据系统的整体需求而定的。例如，在本系统中，为了增加系统的安全性，每个用户首先都要通过系统登录模块的验证才能进入主窗体。这时，就要在数据库中创建一个存储登录名和登录密码的管理人员基本信息表。

管理员基本信息实体 E-R 图如图 12-3 所示。



图 12-3 管理员实体 E-R 图

### 12.5.3 数据库的逻辑结构设计

根据设计好的 E-R 图，在数据库中创建数据表，本系统中只有一个 tb\_admin 表，该表用于保存管理员登录的基本信息，其结构如表 12-2 所示。

表 12-2 管理员登录信息表

| 字段名  | 数据类型 | 主 键 | 描 述   |
|------|------|-----|-------|
| name | 文本   | 否   | 登录用户名 |
| pwd  | 文本   | 否   | 登录密码  |

## 12.6 设计公共类

视频讲解 光盘: 视频\第 12 章\设计公共类.avi

在开发项目中以类的形式来组织、封装一些常用的方法和事件,不仅可以提高代码的重用率,也大大方便了代码的管理。本系统中创建了 5 个公共类,分别为 DataCon、DataOperate、SoftReg、VideoOperate 和 PelcoD 类。

其中,DataCon 类用来访问 Microsoft Access 数据库,DataOperate 类对 Microsoft Access 数据库进行操作,SoftReg 类用来实现生成机器码和系统注册功能,VideoOperate 用来封装视频采集卡中的各种枚举和 API 函数,PelcoD 类用来实现 Pelco-D 协议。在程序开发时,窗体只须调用相应的方法即可。

下面分别对这 5 个类中的方法进行详细介绍。

### 12.6.1 DataCon 类

在 DataCon 类中,因为本系统使用的是 Access 数据库,所以在命名空间区域内引用 using System.Data.OleDb 来连接数据库。该类中定义了 getCon 方法,用来使用 OleDbConnection 对象连接 Access 数据库。

getCon 方法在文件 DataCon.cs 中定义,具体实现代码如下所示:

```
using System.Data;
using System.Data.OleDb;

namespace VWMS.CommonClass
{
    class DataCon
    {
        public OleDbConnection getCon()
        {
            string strDPath = Application.StartupPath;
            string strDataSource = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
                + strDPath.Substring(0, strDPath.LastIndexOf(@"\")).Substring(0,
                strDPath.Substring(0, strDPath.LastIndexOf(@"\")).LastIndexOf(@"\"))
                + @"\DataBase\VWMS.mdb";
            OleDbConnection oledbCon = new OleDbConnection(strDataSource);
            return oledbCon;
        }
    }
}
```

因为 C# 连接 Access 数据库需要导入额外的命名空间,所以在上述代码中有了最前面的两条 using 命令,这是必不可少的。

- Provider: 是指数据提供者,这里使用的是 Microsoft Jet 引擎,也就是 Access 中的数据引擎。
- Data Source: 是指明数据源的位置,在这里可以看出,所要连接的 Access 数据库名为“VWMS.mdb”。

## 12.6.2 DataOperate 类

DataOperate 类中, 首先实例化 datacon、oledbcon、oledbcom、oledbda 和 ds 这 5 个对象。其中, datacon 用来调用自定义类 DataCon 中的方法, oledbcon 对象用来连接 Access 数据库, oledbcom 对象用来执行 Command 命令语句, oledbda 对象表示用于填充 DataSet 数据集和更新 Access 数据库的一组数据命令和一个数据库连接, ds 对象为数据集。

用于实例化 datacon、oledbcon、oledbcom、oledbda 和 ds 这 5 个对象的实现文件是 DataOperate.cs, 具体实现代码如下:

```
namespace VWMS.CommonClass
{
    class DataOperate
    {
        DataCon datacon = new DataCon();
        OleDbConnection oledbcon;
        OleDbCommand oledbcom;
        OleDbDataAdapter oledbda;
        DataSet ds;

        public void getCom(string strCon)
        {
            oledbcon = datacon.getCon();
            oledbcom = new OleDbCommand(strCon, oledbcon);
            oledbcon.Open();
            oledbcom.ExecuteNonQuery();
            oledbcon.Close();
        }

        public DataSet getDs(string strCon, string tbname)
        {
            oledbcon = datacon.getCon();
            oledbda = new OleDbDataAdapter(strCon, oledbcon);
            ds = new DataSet();
            oledbda.Fill(ds, tbname);
            return ds;
        }
    }
}
```

## 12.6.3 SoftReg 类

SoftReg 类中自定义了 GetDiskVolumeSerialNumber、getCpu、getMNum 和 getRNum 这 4 个方法, 具体实现文件是 SoftReg.cs, 下面对几个方法分别进行介绍。

### 1. GetDiskVolumeSerialNumber 方法

GetDiskVolumeSerialNumber 方法用来使用 ManagementObject 对象的 GetPropertyValue 方法获得本机的硬盘标识号, 实现代码如下:

```
// 取得设备硬盘的卷标号
public string GetDiskVolumeSerialNumber()
{
    ManagementClass mc = new ManagementClass("Win32_NetworkAdapterConfiguration");
    ManagementObject disk = new ManagementObject("win32_logicaldisk.deviceid=\"d:\":");
```



```

        disk.Get();
        return disk.GetPropertyValue("VolumeSerialNumber").ToString();
    }

```

## 2. getCpu 方法

getCpu 方法用来获得本机的 CPU 序列号，实现代码如下：

```

//获得CPU的序列号
public string getCpu()
{
    string strCpu = null;
    ManagementClass myCpu = new ManagementClass("win32_Processor");
    ManagementObjectCollection myCpuConnection = myCpu.GetInstances();
    foreach (ManagementObject myObject in myCpuConnection)
    {
        strCpu = myObject.Properties["Processorid"].Value.ToString();
        break;
    }
    return strCpu;
}

```

## 3. getMNum()和 getRNum()方法

getMNum()方法用来生成机器码，方法 getRNum()用于生成注册码。实现代码如下：

```

//生成机器码
public string getMNum()
{
    string strNum = getCpu() + GetDiskVolumeSerialNumber(); //获得24位CPU和硬盘序列号
    string strMNum = strNum.Substring(0, 24); //从生成的字符串中取出前24个字符作为机器码
    return strMNum;
}

public int[] intCode = new int[127]; //存储密钥
public int[] intNumber = new int[25]; //存储机器码的ASCII值
public char[] Charcode = new char[25]; //存储机器码字
public void setIntCode() //给数组赋值小于10的数
{
    for (int i=1; i<intCode.Length; i++)
    {
        intCode[i] = i % 9;
    }
}

//生成注册码
public string getRNum()
{
    setIntCode(); //初始化127位数组
    for (int i=1; i<Charcode.Length; i++) //把机器码存入数组中
    {
        Charcode[i] = Convert.ToChar(this.getMNum().Substring(i-1, 1));
    }
    for (int j=1; j<intNumber.Length; j++) //把字符的ASCII值存入一个整数组中
    {
        intNumber[j] =
            intCode[Convert.ToInt32(Charcode[j])] + Convert.ToInt32(Charcode[j]);
    }
    string strAsciiName = ""; //用于存储注册码
    for (int j=1; j<intNumber.Length; j++)
    {
        if (intNumber[j]>=48 && intNumber[j]<=57) //判断字符ASCII值是否在0~9之间

```

```

{
    strAsciiName += Convert.ToChar(intNumber[j]).ToString();
}
else if (intNumber[j] >= 65 && intNumber[j] <= 90) //判断字符 ASCII 值是否在 A~Z 之间
{
    strAsciiName += Convert.ToChar(intNumber[j]).ToString();
}
else if (intNumber[j] >= 97 && intNumber[j] <= 122) //判断字符 ASCII 值是否在 a~z 之间
{
    strAsciiName += Convert.ToChar(intNumber[j]).ToString();
}
else //判断字符 ASCII 值不在以上范围内
{
    if (intNumber[j] > 122) //判断字符 ASCII 值是否大于 z
    {
        strAsciiName += Convert.ToChar(intNumber[j] - 10).ToString();
    }
    else
    {
        strAsciiName += Convert.ToChar(intNumber[j] - 9).ToString();
    }
}
}
return strAsciiName;
}

```

在 Windows 系统中提供了一个以“Win32\_”为前缀的类，用于获取当前系统的硬件信息。为了取得硬件信息，首先需要创建一个 ManagementObjectSearcher 对象：

```
ManagementObjectSearcher searcher = new ManagementObjectSearcher("select * from " + Key);
```

获取各个硬件信息和系统信息的具体说明如下所示：

```

// 硬件
Win32_Processor, // CPU 处理器
Win32_PhysicalMemory, // 物理内存条
Win32_Keyboard, // 键盘
Win32_PointingDevice, // 点输入设备，包括鼠标
Win32_FloppyDrive, // 软盘驱动器
Win32_DiskDrive, // 硬盘驱动器
Win32_CDROMDrive, // 光盘驱动器
Win32_BaseBoard, // 主板
Win32_BIOS, // BIOS 芯片
Win32_ParallelPort, // 并口
Win32_SerialPort, // 串口
Win32_SerialPortConfiguration, // 串口配置
Win32_SoundDevice, // 多媒体设置，一般指声卡
Win32_SystemSlot, // 主板插槽 (ISA & PCI & AGP)
Win32_USBController, // USB 控制器
Win32_NetworkAdapter, // 网络适配器
Win32_NetworkAdapterConfiguration, // 网络适配器设置
Win32_Printer, // 打印机
Win32_PrinterConfiguration, // 打印机设置
Win32_PrintJob, // 打印机任务
Win32_TCPIPPrinterPort, // 打印机端口
Win32_POTSModem, // MODEM
Win32_POTSModemToSerialPort, // MODEM 端口
Win32_DesktopMonitor, // 显示器
Win32_DisplayConfiguration, // 显卡
Win32_DisplayControllerConfiguration, // 显卡设置

```



```
Win32_VideoController, // 显卡细节
Win32_VideoSettings, // 显卡支持的显示模式

// 操作系统
Win32_TimeZone, // 时区
Win32_SystemDriver, // 驱动程序
Win32_DiskPartition, // 磁盘分区
Win32_LogicalDisk, // 逻辑磁盘
Win32_LogicalDiskToPartition, // 逻辑磁盘所在分区及始末位置
Win32_LogicalMemoryConfiguration, // 逻辑内存配置
Win32_PageFile, // 系统页文件信息
Win32_PageFileSetting, // 页文件设置
Win32_BootConfiguration, // 系统启动配置
Win32_ComputerSystem, // 计算机信息简要
Win32_OperatingSystem, // 操作系统信息
Win32_StartupCommand, // 系统自动启动程序
Win32_Service, // 系统安装的服务
Win32_Group, // 系统管理组
Win32_GroupUser, // 系统组账号
Win32_UserAccount, // 用户账号
Win32_Process, // 系统进程
Win32_Thread, // 系统线程
Win32_Share, // 共享
Win32_NetworkClient, // 已安装的网络客户端
Win32_NetworkProtocol, // 已安装的网络协议
```

## 12.6.4 类 VideoOperate

类 VideoOperate 主要封装了操作视频采集卡的各种枚举及方法。

由于要调用 Sa7134Capture.dll 动态链接库，所以要引用 System.Runtime.InteropServices 命名空间。上述功能的实现文件是 VideoOperate.cs，封装 Sa7134Capture.dll 动态链接库中各种枚举及方法的关键代码如下：

```
#region 视频采集卡中的枚举
public enum DISPLAYTRANSTYPE
{
    NOT_DISPLAY = 0,
    PCI_VIEDOMEMORY = 1,
    PCI_MEMORY_VIDEOMEMORY = 2
}
//视频预览和视频捕捉数据流格式，目前版本只支持 UUY2 格式
public enum COLORFORMAT
{
    RGB32 = 0x0,
    RGB24 = 0x1,
    RGB16 = 0x2,
    RGB15 = 0x3,
    YUY2 = 0x4,
    BTYUV = 0x5,
    Y8 = 0x6,
    RGB8 = 0x7,
    PL422 = 0x8,
    PL411 = 0x9,
    YUV12 = 0xA,
    YUV9 = 0xB,
    RAW = 0xE
}
```



```

/*视频预览及视频捕获的显示属性, 其中:
BRIGHTNESS 为亮度, value 范围是 0~255, 最佳 80
CONTRAST 为对比度, value 范围是-128~127, 最佳 0x44
SATURATION 为饱和度, value 范围是-128~127, 最佳 0x40
HUE 为色度, value 范围是-128~127, 最佳 0x0,
只有当 COLORDEVICETYPE 等于 COLOR_DECODER 时才有效
SHARPNESS 为锐度, value 范围是-8~7, 最佳 0x0,
只有当 COLORDEVICETYPE 等于 COLOR_DECODER 时才有效
*/
public enum COLORCONTROL
{
    BRIGHTNESS = 0,
    CONTRAST = 1,
    SATURATION = 2,
    HUE = 3,
    SHARPNESS = 4
}

/*显示设备的显示属性, 其中:
COLOR_DECODER 为解码器的显示属性, 会影响视频预览和视频捕获的显示属性
COLOR_PREVIEW 为视频预览的显示属性
COLOR_CAPTURE 为视频捕获的显示属性
*/
public enum COLORDEVICETYPE
{
    COLOR_DECODER = 0,
    COLOR_PREVIEW = 1,
    COLOR_CAPTURE = 2,
}

/*音视频捕获方式, 其中:
CAP_NULL_STREAM 捕获无效
CAP_ORIGIN_STREAM 捕获为原始流回调
CAP_MPEG4_STREAM 捕获为 MPEG4
*/
public enum CAPMODEL
{
    CAP_NULL_STREAM = 0,
    CAP_ORIGIN_STREAM = 1,
    CAP_MPEG4_STREAM = 2,
}

/*音视频 MPEG4 捕获方式, 只有 CAPMODEL 等于 CAP_MPEG4_STREAM 时有效, 其中:
MPEG4_AVIFILE_ONLY          存为 MPEG4 文件
MPEG4_CALLBACK_ONLY         MPEG 数据回调
MPEG4_AVIFILE_CALLBACK      存为 MPEG 文件并回调
*/
public enum MP4MODEL
{
    MPEG4_AVIFILE_ONLY = 0,
    MPEG4_CALLBACK_ONLY = 1,
    MPEG4_AVIFILE_CALLBACK = 2,
}

/*MPEG4_XVID 压缩模式, 其中:
XVID_CBR_MODE
XVID_VBR_MODE
*/
public enum COMPRESSMODE

```

```

{
    XVID_CBR_MODE = 0,
    XVID_VBR_MODE = 1,
}

/*视频源的输入频率, 其中:
FIELD_FREQ_50HZ 50HZ    绝对多数为 PAL 制式
FIELD_FREQ_60HZ 60HZ    绝对多数为 NTSC 制式
FIELD_FREQ_0HZ    无信号
*/
public enum eFieldFrequency
{
    FIELD_FREQ_50HZ = 0,
    FIELD_FREQ_60HZ = 1,
    FIELD_FREQ_0HZ = 2,
}

/*电平状态, 其中:
HIGH_VOLTAGE 高电平
LOW_VOLTAGE 低电平
*/
public enum eVOLTAGELEVEL
{
    HIGH_VOLTAGE = 0,
    LOW_VOLTAGE = 1,
}

```

### 12.6.5 类 PelcoD

类 PelcoD 主要封装了控制云台的 Pelco-D 协议, 该协议由 7 个字节构成。第一个字节为同步字节, 始终为 FFH; 第 2 个字节为地址码, 也就是摄像头的逻辑地址号; 第 3、4 个字节表示指令码, 即执行哪项操作, 第 5、6 个字节表示数据码, 用于指定摄像头的水平、垂直方向移动速度; 第 7 个字节为校验码。

上述功能的实现文件是 PelcoD.cs, PelcoD 类中, 通过将 Pelco-D 协议中的 7 个字节返回为串口消息值来对云台进行控制, 其关键代码如下:

```

#region 云台控制方法
//雨刷控制
public byte[] CameraSwitch(uint deviceAddress, Switch action)
{
    byte m_action = CameraOnOff;
    if (action == Switch.On)
        m_action = CameraOnOff + Sense;
    return Message.GetMessage(deviceAddress, m_action, 0x00, 0x00, 0x00);
}
//光圈控制
public byte[] CameraIrisSwitch(uint deviceAddress, Iris action)
{
    return Message.GetMessage(deviceAddress, (byte)action, 0x00, 0x00, 0x00);
}
//聚焦控制
public byte[] CameraFocus(uint deviceAddress, Focus action)
{
    if (action == Focus.Near)
        return Message.GetMessage(deviceAddress, (byte)action, 0x00, 0x00, 0x00);
    else

```

```

        return Message.GetMessage(deviceAddress, 0x00, (byte)action, 0x00, 0x00);
    }
    //对焦控制
    public byte[] CameraZoom(uint deviceAddress, Zoom action)
    {
        return Message.GetMessage(deviceAddress, 0x00, (byte)action, 0x00, 0x00);
    }
    //上下控制
    public byte[] CameraTilt(uint deviceAddress, Tilt action, uint speed)
    {
        if (speed < TiltSpeedMin)
            speed = TiltSpeedMin;
        if (speed > TiltSpeedMax)
            speed = TiltSpeedMax;
        return Message.GetMessage(deviceAddress, 0x00, (byte)action, 0x00, (byte)speed);
    }
    //左右控制
    public byte[] CameraPan(uint deviceAddress, Pan action, uint speed)
    {
        if (speed < PanSpeedMin)
            speed = PanSpeedMin;
        if (speed > PanSpeedMax)
            speed = PanSpeedMax;
        return Message.GetMessage(deviceAddress, 0x00, (byte)action, (byte)speed, 0x00);
    }
    //停止云台的移动
    public byte[] CameraStop(uint deviceAddress)
    {
        return Message.GetMessage(deviceAddress, 0x00, 0x00, 0x00, 0x00);
    }
    //自动和手动控制
    public byte[] CameraScan(uint deviceAddress, Scan scan)
    {
        byte m_byte = AutoManualScan;
        if (scan == Scan.Auto)
            m_byte = AutoManualScan + Sense;
        return Message.GetMessage(deviceAddress, m_byte, 0x00, 0x00, 0x00);
    }
}
#endregion

public struct Message
{
    public static byte Address;
    public static byte CheckSum;
    public static byte Command1, Command2, Data1, Data2;
    public static byte[] GetMessage(
        uint address, byte command1, byte command2, byte data1, byte data2)
    {
        if (address < 1 || address > 256)
            throw new Exception("Pelco D 协议只支持 256 设备");
        Address = Byte.Parse((address).ToString());
        Data1 = data1;
        Data2 = data2;
        Command1 = command1;
        Command2 = command2;
        CheckSum = (byte)(STX ^ Address ^ Command1 ^ Command2 ^ Data1 ^ Data2);
        return new byte[] { STX, Address, Command1, Command2, Data1, Data2, CheckSum };
    }
}

```



## 12.7 具体编码

视频讲解 光盘：视频\第 12 章\具体编码.avi

到目前为止，前面所有的准备工作都已经结束了。从本节内容开始，将详细讲解本项目的编码工作的实现过程。

### 12.7.1 登录模块

系统登录主要用于对进入家庭视频监控系统的用户进行安全性检查，以防止非法用户进入该系统。在登录时，只有合法的用户才可以进入该系统。

系统登录模块的运行结果如图 12-4 所示。

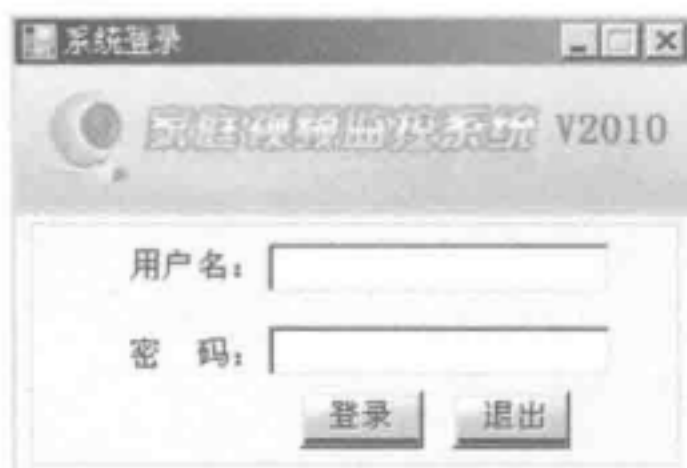


图 12-4 登录模块

登录模块的重点，在于将用户输入的用户名和密码与数据库中的用户名和密码进行比较，如果相同，将允许用户进入系统的操作界面；否则，会弹出提示框，提示用户输入的用户名或者密码错误。该模块的实现原理，是根据用户输入的用户名和密码在数据库中查找是否有相符的记录，并将查询结果填充到 DataSet 数据集中，然后判断该数据集中所包含表的行数是否大于零，如果大于零，则表示输入的用户名和密码正确，从而成功登录系统；否则，弹出提示信息。

登录模块的实现过程并不复杂，具体实现步骤如下。

- (1) 新建一个 Windows 窗体，命名为 frmLogin.cs，主要用于实现系统的登录功能。
- (2) 在 frmLogin.cs 代码文件中，首先实例化公共类 DataCon 和 DataOperate 的两个全局对象，通过类对象调用类中的功能方法。

实例化 DataCon 和 DataOperate 类对象的关键代码如下：

```
DataCon datacon = new DataCon();  
DataOperate dataoperate = new DataOperate();
```

- (3) 单击“登录”按钮，首先判断是否输入了用户名，如果没有输入用户名，就弹出信息提示框，提示用户名不能为空；否则，系统会判断输入的用户名和密码是否与数据库中的记录相符。如果符合，则隐藏当前窗体，并显示主窗体；否则，弹出“用户名或密码错误”信息提示。“登录”按钮的 Click 事件代码如下：

```
private void btnLogin_Click(object sender, EventArgs e)  
{  
    if (txtName.Text == "")
```

```

{
    errorProName.SetError(txtName, "用户名不能为空!");
}
else
{
    errorProName.Clear();
    string strSql = "select * from tb_admin where name='" + txtName.Text
        + "' and pwd='" + txtPwd.Text + "'";
    DataSet ds = dataoperate.getDs(strSql, "tb_admin");
    if (ds.Tables[0].Rows.Count > 0)
    {
        this.Hide();
        Main frmmain = new Main();
        frmmain.Show();
    }
    else
    {
        MessageBox.Show("用户名或密码错误!", "警告",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
}
}

```

(4) 当输入用户名或密码之后,还可以按 Enter 键将鼠标焦点移动到下一个控件上。实现原理是在输入用户名或密码的文本框的 KeyPress 事件下,判断是否按了 Enter 键,如果是,则将焦点移动到下一个控件上。按 Enter 键移动鼠标焦点的实现代码如下:

```

private void txtName_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        txtPwd.Focus();
        e.Handled = true;
    }
}

private void txtPwd_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        btnLogin.Focus();
        e.Handled = true;
    }
}

```

(5) 单击“退出”按钮,退出当前应用程序。“退出”按钮的 Click 事件代码如下:

```

private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}

```

## 12.7.2 视频监控模块

家庭视频监控系统的主要功能就是视频监控,视频监控模块主要用来监控某一区域的日常变化情况,用户还可以通过“云台控制”和“方向控制”两大功能监控其他区域的日常变化情况。另外,如果用户临时需要离开,可以将该区域的变化情况录制为视频文件,

以便后期查看。视频监控模块的运行结果如图 12-5 所示。

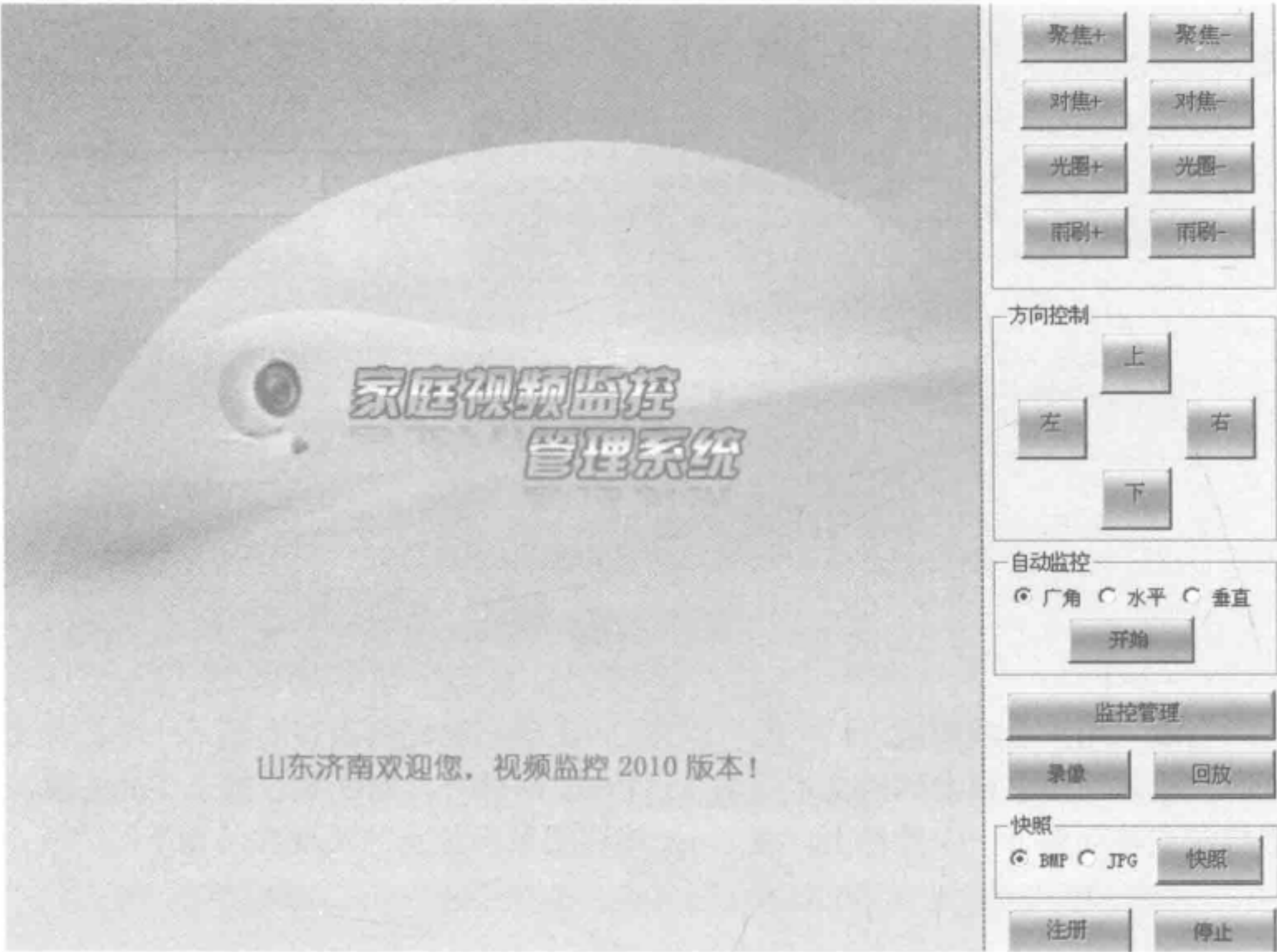


图 12-5 视频监控模块

1. 技术分析

(1) SDK 开发包

SDK 是视频采集卡厂商提供的开发视频监控系统的一组库函数，使用 SDK 开发包中的库函数，程序开发人员可以在不了解视频压缩、回放和网络传输等技术的前提下，进行视频程序开发(SDK 开发包中包含了这些技术的实现，程序开发人员可以直接调用)。SDK 开发包中所包含的库函数一般在购买产品的光盘中有详细说明。

(2) 串口通信技术

该模块中主要使用串口通信技术实现了对云台和方向的控制。C#中提供了 SerialPort 类来实现串口通信，该类位于 System.IO.Ports 命名空间下，主要用于控制串行端口文件资源，它提供同步 I/O 和事件驱动的 I/O、对管脚和中断状态的访问以及对串行驱动程序属性的访问。另外，此类还可以包装在内部 Stream 对象中，可通过 BaseStream 属性访问，并且可以传递给包装或使用流的类。

SerialPort 类的常用属性及说明如表 12-3 所示。

表 12-3 SerialPort 类常用的属性及说明

| 属 性        | 说 明                           |
|------------|-------------------------------|
| BaseStream | 获取 SerialPort 对象的基础 Stream 对象 |
| BaudRate   | 获取或设置串行波特率                    |



| 属 性             | 说 明                                |
|-----------------|------------------------------------|
| BreakState      | 获取或设置中断信号状态                        |
| BytesToRead     | 获取接收缓冲区中数据的字节数                     |
| BytesToWrite    | 获取发送缓冲区中数据的字节数                     |
| CtsHolding      | 获取“可以发送”行的状态                       |
| DataBits        | 获取或设置每个字节的标准数据位长度                  |
| Encoding        | 获取或设置传输前后文本转换的字节编码                 |
| Handshake       | 获取或设置串行端口数据传输的握手协议                 |
| IsOpen          | 获取一个值，该值指示 SerialPort 对象的打开或关闭状态   |
| ParityReplace   | 获取或设置一个字节，该字节在发生奇偶校验错误时替换数据流中的无效字节 |
| PortName        | 获取或设置通信端口，包括但不限于所有可用的 COM 端口       |
| ReadBufferSize  | 获取或设置 SerialPort 输入缓冲区的大小          |
| ReadTimeout     | 获取或设置读取操作未完成时发生超时之前的毫秒数            |
| StopBits        | 获取或设置每个字节的标准停止位数                   |
| WriteBufferSize | 获取或设置串行端口输出缓冲区的大小                  |

SerialPort 类的常用方法及说明如表 12-4 所示。

表 12-4 SerialPort 类的常用方法及说明

| 方 法          | 说 明                                          |
|--------------|----------------------------------------------|
| Close        | 关闭端口连接，将 IsOpen 属性设置为 false，并释放内部的 Stream 对象 |
| GetPortNames | 获取当前计算机的串行端口名称数组                             |
| Open         | 打开一个新的串行端口连接                                 |
| Read         | 从 SerialPort 输入缓冲区中读取                        |
| ReadByte     | 从 SerialPort 输入缓冲区中同步读取一个字节                  |
| ReadChar     | 从 SerialPort 输入缓冲区中同步读取一个字符                  |
| ReadExisting | 在编码的基础上，读取 SerialPort 对象的流和输入缓冲区中所有立即可用的字节   |
| ReadLine     | 一直读取到输入缓冲区中的 NewLine 值                       |
| ReadTo       | 一直读取到输入缓冲区中的指定 value 的字符串                    |
| Write        | 将数据写入串行端口输出缓冲区                               |
| WriteLine    | 将指定的字符串和 NewLine 值写入输出缓冲区                    |

通过下面的代码，使用 SerialPort 类向 COM1 端口中写入数据，并进行异步读取：

```
using System.IO.Ports;
using System.Threading;
//实例化串口对象(默认: COM1,9600,e,8,1)
SerialPort SPort = new SerialPort();
private void Form1_Load(object sender, EventArgs e)
{
    //更改参数
    SPort.PortName = "COM1";
```

```

    SPort.BaudRate = 9600;
    SPort.Parity = Parity.None;
    SPort.DataBits = 8;
    SPort.StopBits = StopBits.One;
    SPort.ReadBufferSize = 4096;
}
//发送并接收串口信息
private void button1_Click(object sender, EventArgs e)
{
    //打开串口
    if (SPort.IsOpen == false)
    {
        SPort.Open();
    }
    byte[] bytesSend = System.Text.Encoding.Default.GetBytes(richTextBox1.Text);
    SPort.Write(bytesSend, 0, bytesSend.Length);
    MessageBox.Show("发送串口信息成功");
    ReceiveData(SPort);
}
//异步读取
private void AsyReceiveData(object serialPortobj)
{
    SerialPort serialport = (SerialPort)serialPortobj;
    System.Threading.Thread.Sleep(1000);
    MessageBox.Show(serialport.ReadExisting());
    serialport.Close();
}
//开启接收数据线程
private void ReceiveData(SerialPort serialPort)
{
    //异步接收数据线程
    Thread threadReceiveSub = new Thread(new ParameterizedThreadStart(AsyReceiveData));
    threadReceiveSub.Start(serialPort);
}

```

## 2. 具体实现过程

(1) 新建一个 Windows 窗体，命名为 frmMain.cs，主要用于实现视频监控功能及各种监控控制。

(2) frmMain.cs 代码文件中，首先实例化公共类 PelcoD 和 SoftReg 的两个对象，以便调用其中的方法，然后实例化一个 SerialPort 类对象，用来向云台发送串口消息，声明一个 int 类型的变量、两个 byte 类型的变量和一个 byte 类型的数组，分别用来获取视频采集卡的路数、存储 Pelco-D 协议中的地址码、数据码和要向云台传送的串口消息。

对应的代码如下：

```

PelcoD pelcod = new PelcoD();
SoftReg softreg = new SoftReg();
SerialPort serialPort =
    new SerialPort("COM1", 2400, Parity.None, 8); //实例化串口信息类，指定串口号
int m_dwDevNum = 0; //存储视频路数
byte addressin = Byte.Parse(Convert.ToString(0x01)); //PelcoD 协议中的第一地址码
byte speedin = Byte.Parse(Convert.ToString(0xff)); //PelcoD 协议中的第二地址码
byte[] messagesend; //存储要发送的串口消息

```

(3) 自定义了两个方法 startMonitor 和 stopMove。其中，startMonitor 方法用来实现开始监控功能，其实现代码如下：

```
//开始监控
protected void startMonitor()
{
    if (VideoOperate.VCAInitSdk(this.Handle,
        VideoOperate.DISPLAYTRANSType.PCI_MEMORY_VIDEOMEMORY, false))
    {
        m_dwDevNum = VideoOperate.VCAGetDevNum();
        if (m_dwDevNum == 0)
        {
            MessageBox.Show("VC404 卡驱动程序没有安装");
        }
        else
        {
            for (int i=0; i<m_dwDevNum; i++)
            {
                VideoOperate.VCAOpenDevice(i, plVideol.Handle);
                VideoOperate.VCAStartVideoPreview(i);
            }
        }
    }
}
```

在上述代码中，有如下 3 个重要方法。

- **VCAGetDevNum**: 该方法用来返回系统中的卡号数量，即 SAA7134 硬件的数目。
- **VCAOpenDevice**: 该方法用来打开指定卡号的设备，并分配相应的系统资源。
- **VCAStartVideoPreview**: 该方法用来开始视频预览。

(4) **stopMove** 方法用来实现停止移动视频监控画面的功能，其实现代码如下：

```
//停止移动
protected void stopMove()
{
    messagesend = pelcod.CameraStop(addressin);
    serialPort.Open();
    serialPort.Write(messagesend, 0, 7);
    serialPort.Close();
}
```

在上述代码中，有如下 3 个重要方法。

- **Open**: 打开一个新的串行端口连接。
- **Write**: 将数据写入串行端口输出缓冲区。
- **Close**: 关闭端口连接。

(5) **frmMain** 窗体在加载时，首先从注册表中提取注册信息，以判断该软件是否注册，如果已经注册，则在窗体标题栏中显示“已注册”字样，同时开始视频监控；否则，在窗体标题栏中显示“未注册”字样，并提示用户已试用次数。**frmMain** 窗体的 Load 事件代码如下：

```
//窗体加载时，初始化视频卡，并开始预览视频
private void frmMain_Load(object sender, EventArgs e)
{
    plVideol.BackgroundImage = null;
    RegistryKey retkey = Microsoft.Win32.Registry.CurrentUser.OpenSubKey(
        "software", true).CreateSubKey("wxk").CreateSubKey("wxk.INI");
    foreach (string strRNum in retkey.GetSubKeyNames()) //判断是否注册
    {
        if (strRNum == softreg.getRNum())
        {
            //已注册
        }
    }
}
```



```

    {
        this.Text = "家庭视频监控系统(已注册)";
        btnReg.Enabled = false;
        startMonitor();
        return;
    }
}
this.Text = "家庭视频监控系统(未注册)";
btnReg.Enabled = true;
btnSetMonitor.Enabled = btnAutoMonitor.Enabled = false;
startMonitor();
MessageBox.Show("您现在使用的是试用版, 该软件可以免费试用 30 次!", "提示",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
Int32 tLong;
try
{
    tLong = (Int32)Registry.GetValue(
        "HKEY_LOCAL_MACHINE\\SOFTWARE\\angel", "UseTimes", 0);
    MessageBox.Show("感谢您已使用了" + tLong + "次", "提示",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
catch
{
    Registry.SetValue("HKEY_LOCAL_MACHINE\\SOFTWARE\\angel",
        "UseTimes", 0, RegistryValueKind.DWord);
    MessageBox.Show("欢迎新用户使用本软件", "提示",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
tLong = (Int32)Registry.GetValue(
    "HKEY_LOCAL_MACHINE\\SOFTWARE\\angel", "UseTimes", 0);
if (tLong < 30)
{
    int Times = tLong + 1;
    Registry.SetValue("HKEY_LOCAL_MACHINE\\SOFTWARE\\angel", "UseTimes", Times);
}
else
{
    MessageBox.Show("试用次数已到", "警告",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    Application.Exit();
}
}

```

(6) 当用鼠标拖动 frmMain 窗体时, 调用公共类中的 VCAUpdateOverlayWnd 和 VCAUpdateVideoPreview 方法, 更新在 Panel 控件中显示的视频监控画面, 使视频监控画面随之移动。frmMain 窗体的 Move 事件代码如下:

```

//移动窗体位置时, 视频随之移动
private void frmMain_Move(object sender, EventArgs e)
{
    for (int i=0; i<m_dwDevNum; i++)
    {
        plVideol.Invalidate();
        VideoOperate.VCAUpdateOverlayWnd(this.Handle);
        VideoOperate.VCAUpdateVideoPreview(i, plVideol.Handle);
    }
}

```

(7) 当按下“聚焦+”按钮时, 调用公共类 PelcoD 中的 CameraFocus 方法, 实现增加

聚焦功能。“聚焦+”按钮的 MouseDown 事件代码如下：

```
#region 云台控制
//增加聚焦
private void btnAHighlights_MouseDown(object sender, MouseEventArgs e)
{
    messagesend = pelcod.CameraFocus(addressin, PelcoD.Focus.Near);
    serialPort.Open();
    serialPort.Write(messagesend, 0, 7);
    serialPort.Close();
}
```

“聚焦-”、“对焦+”、“对焦-”、“光圈+”、“光圈-”、“雨刷+”和“雨刷-”按钮的 MouseDown 事件的实现过程与上述“聚焦+”按钮的代码大致相同，只须调用 PelcoD 类中的相应方法，并传递不同的参数即可，在此将不再详细描述。

(8) 当按下“上”按钮时，调用公共类 PelcoD 中的 CameraTilt 方法将云台向上转动。“上”按钮的 MouseDown 事件代码如下：

```
//方向控制——上
private void btnUp_MouseDown(object sender, MouseEventArgs e)
{
    messagesend = pelcod.CameraTilt(addressin, PelcoD.Tilt.Up, speedin);
    serialPort.Open();
    serialPort.Write(messagesend, 0, 7);
    serialPort.Close();
}
```

“下”、“左”和“右”按钮的 MouseDown 事件的实现过程与上述“上”按钮的代码大致相同，只须调用 PelcoD 类中的相应方法，并传递不同的参数即可，不再详细描述。

(9) 当从“上”按钮上释放鼠标时，调用自定义方法 stopMove 停止移动视频画面。“上”按钮的 MouseUp 事件代码如下：

```
private void btnUp_MouseUp(object sender, MouseEventArgs e)
{
    stopMove();
}
```

“聚焦+”、“聚焦-”、“对焦+”、“对焦-”、“光圈+”、“光圈-”、“雨刷+”、“雨刷-”、“下”、“左”和“右”按钮的 MouseUp 事件的实现过程与上述“上”按钮的实现代码大致相同，只须调用 PelcoD 类中的相应方法，并传递不同的参数即可，在此将不再详细描述。

在“自动监控”区域，选择要自动监控的方向，单击“开始”按钮，开始自动监控。自动监控的实现原理是根据选择的方向控制云台的转动方向，因此根据需要调用“上”、“下”、“左”和“右”按钮的 MouseUp 事件中的代码即可。

(10) 单击“监控管理”按钮，实例化监控管理窗体 frmSetMonitor 的一个对象，并调用其 ShowDialog 方法显示该窗体。“监控管理”按钮的 Click 事件代码如下：

```
//打开监控管理窗体
private void btnSetMonitor_Click(object sender, EventArgs e)
{
    SetMonitor frmsetmonitor = new SetMonitor();
    frmsetmonitor.ShowDialog();
}
```



(11) 单击“录像”按钮，设置要保存文件的格式及默认路径，打开“保存视频文件”对话框，然后调用公共类 VideoOperate 中的相应方法，对要保存的内容进行处理后，保存为视频文件，同时设置“录像”按钮的 Text 值为“停止录像”，这时，再次单击该按钮，调用公共类 VideoOperate 中的 VCASStop VideoCapture 方法停止录像。

“录像”按钮的 Click 事件代码如下：

```
//录像
private void btnVideo_Click(object sender, EventArgs e)
{
    if (btnVideo.Text == "录像")
    {
        sfDialog.Filter = "*.avi|*.avi";
        sfDialog.Title = "保存视频文件";
        sfDialog.InitialDirectory = Application.StartupPath.Substring(0,
            Application.StartupPath.LastIndexOf(@"\")).Substring(0,
            Application.StartupPath.Substring(0,
            Application.StartupPath.LastIndexOf(@"\")).LastIndexOf(@"\")) + @"\Video\";

        if (sfDialog.ShowDialog() == DialogResult.OK)
        {
            btnVideo.Text = "停止录像";
            VideoOperate.VCASetKeyFrmInterval(0, 250);
            VideoOperate.VCASetBitRate(0, 256);
            VideoOperate.VCASetVidCapFrameRate(0, 25, false);
            VideoOperate.VCASetVidCapSize(0, 320, 240);
            VideoOperate.VCASetXVIDQuality(0, 10, 3);
            VideoOperate.VCASetXVIDCompressMode(
                0, VideoOperate.COMPRESSMODE.XVID_VBR_MODE);
            VideoOperate.VCAStartVideoCapture(
                0, VideoOperate.CAPMODEL.CAP_MPEG4_STREAM,
                VideoOperate.MP4MODEL.MPEG4_AVIFILE_CALLBACK, sfDialog.FileName);
        }
    }
    else if (btnVideo.Text == "停止录像")
    {
        btnVideo.Text = "录像";
        VideoOperate.VCASStopVideoCapture(0);
    }
}
```

(12) 单击“回放”按钮，实例化视频回放窗体 frmPlay 的一个对象，并调用其 ShowDialog 方法显示该窗体。“回放”按钮的 Click 事件代码如下：

```
//回放
private void btnPlay_Click(object sender, EventArgs e)
{
    Play frmplay = new Play();
    frmplay.ShowDialog();
}
```

(13) 单击“快照”按钮，首先检查 BMP 和 JPG 单选按钮哪个处于选中状态，如果 BMP 单选按钮处于选中状态，则调用公共类 VideoOperate 中的 VCASaveAsBmpFile 方法将当前视频监控画面保存为 BMP 文件；如果 JPG 单选按钮处于选中状态，则调用公共类 VideoOperate 中的 VCASaveAsJpegFile 方法将当前视频监控画面保存为 JPG 文件。

“快照”按钮的 Click 事件代码如下：



```
//快照
private void btnSnapShots_Click(object sender, EventArgs e)
{
    if (rbtnBMP.Checked)
    {
        VideoOperate.VCASaveAsBmpFile(0, Application.StartupPath.Substring(
            0, Application.StartupPath.LastIndexOf(@"\")).Substring(0,
            Application.StartupPath.Substring(
            0, Application.StartupPath.LastIndexOf(@"\")).LastIndexOf(@"\"))
            + @"\Photo\" + DateTime.Now.ToFileTime() + ".bmp");
    }
    else
    {
        VideoOperate.VCASaveAsJpegFile(0, Application.StartupPath.Substring(0,
            Application.StartupPath.LastIndexOf(@"\")).Substring(0,
            Application.StartupPath.Substring(
            0, Application.StartupPath.LastIndexOf(@"\")).LastIndexOf(@"\"))
            + @"\Photo\" + DateTime.Now.ToFileTime() + ".jpg", 100);
    }
}
```

(14) 单击“注册”按钮，实例化软件注册窗体 frmRegister 的一个对象，并调用其 ShowDialog 方法显示该窗体，同时调用 Hide 方法隐藏当前窗体。“注册”按钮的 Click 事件代码如下：

```
//打开软件注册窗体
private void btnReg_Click(object sender, EventArgs e)
{
    Register frmregister = new Register();
    frmregister.Show();
    this.Hide();
}
```

### 12.7.3 监控管理模块

监控管理模块主要用来对系统登录用户进行管理，通过此模块，可以添加、修改和删除用户信息。监控管理模块的运行结果如图 12-6 所示。

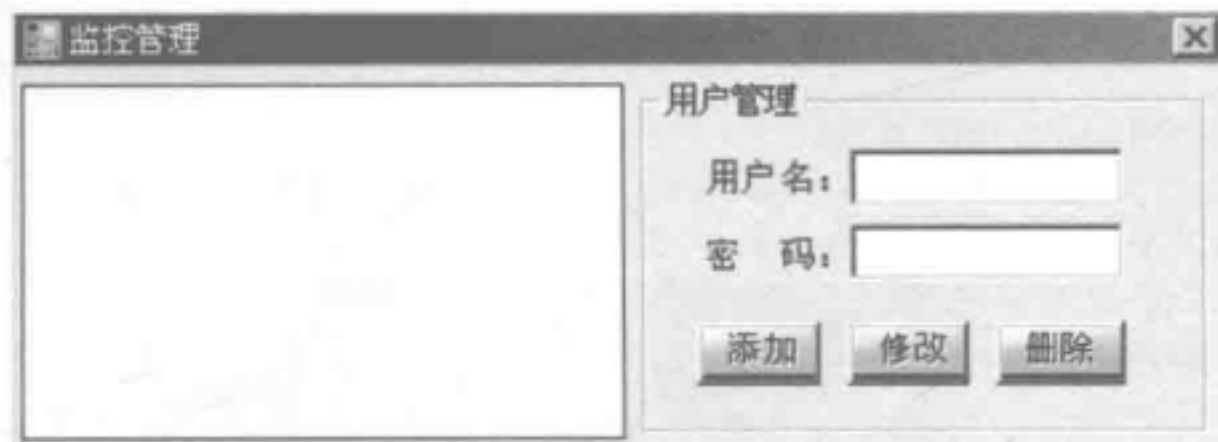


图 12-6 监控管理模块

监控管理模块的具体实现步骤如下。

(1) 新建一个 Windows 窗体，命名为 frmSetMonitor.cs，主要用于实现系统用户的管理功能。

(2) frmSetMonitor.cs 代码文件中，实例化公共类 DataOperate 的一个对象，用来调用其中的方法，然后实例化一个 DataSet 对象，用来作为数据集。

具体代码如下：

```
DataOperate dataoperate = new DataOperate();
DataSet ds;
```

(3) frmSetMonitor 窗体加载时, 调用自定义方法 lviewBind 对 ListView 控件进行数据绑定, 显示数据库中已经存在的用户。frmSetMonitor 窗体的 Load 事件代码如下:

```
private void frmSetMonitor_Load(object sender, EventArgs e)
{
    lviewBind();
}
```

(4) 在文件 frmSetMonitor.cs 中自定义了一个 lviewBind 方法, 该方法用来对 ListView 控件进行数据绑定, 以显示数据库中已经存在的用户。lviewBind 方法的实现代码如下:

```
public void lviewBind()
{
    lview.Items.Clear();
    ds = dataoperate.getDs("select name from tb_admin", "tb_admin");
    foreach (DataRow dr in ds.Tables[0].Rows)
    {
        ListViewItem lvItem = new ListViewItem(dr[0].ToString(), 0);
        lvItem.SubItems.Add(dr[0].ToString());
        lview.Items.Add(lvItem);
    }
}
```

(5) 单击“添加”按钮, 首先判断“用户名”文本框是否为空, 如果为空, 则弹出“用户名不能为空”提示信息; 否则, 判断数据库中是否存在该用户, 如果存在, 则弹出“该用户已经存在”信息提示; 否则, 将“用户名”文本框中的用户名存储到数据库中。

“添加”按钮的 Click 事件代码如下:

```
private void btnAdd_Click(object sender, EventArgs e)
{
    if (txtName.Text == string.Empty)
    {
        MessageBox.Show("用户名不能为空!", "提示",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        ds = dataoperate.getDs(
            "select * from tb_admin where name='" + txtName.Text + "'", "tb_admin");
        if (ds.Tables[0].Rows.Count > 0)
        {
            MessageBox.Show("该用户已经存在!", "提示",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            dataoperate.getCom("insert into tb_admin (name,pwd) values('"
                + txtName.Text + "','" + txtPwd.Text + "')");
            lviewBind();
            txtName.Text = txtPwd.Text = string.Empty;
        }
    }
}
```

(6) 单击“修改”按钮, 首先判断修改后的“用户名”是否为空, 如果是, 弹出“用

户名或密码不能为空”的信息提示，否则成功修改选中的用户信息。

“修改”按钮的 Click 事件代码如下：

```
private void btnEdit_Click(object sender, EventArgs e)
{
    if (txtName.Text == string.Empty && txtPwd.Text == string.Empty)
    {
        MessageBox.Show("用户名或密码不能为空!", "提示",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        dataoperate.getCom("update tb_admin set pwd='" + txtPwd.Text
            + "' where name='" + txtName.Text + "'");
        lviewBind();
        txtName.Text = txtPwd.Text = string.Empty;
    }
}
```

(7) 单击“删除”按钮，首先判断选中的用户是不是超级用户，如果是，弹出“该用户是超级用户，不能删除”信息提示；否则，删除选中的用户。

“删除”按钮的 Click 事件代码如下：

```
private void btnDel_Click(object sender, EventArgs e)
{
    if (txtName.Text.ToLower() == "tsoft")
    {
        MessageBox.Show("该用户是超级用户，不能删除!", "警告",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        dataoperate.getCom("delete from tb_admin where name='" + txtName.Text + "'");
        lviewBind();
        txtName.Text = lview.Items[0].Text;
    }
}
```

## 12.7.4 录像回放模块

录像回放模块主要用来选择播放已录制的视频文件。该模块的运行结果如图 12-7 所示。



图 12-7 录像回放模块



该模块中，单击“回放”按钮，将打开“选择视频文件”对话框，用户选择完要播放的视频文件后，单击“确定”按钮，即可在该模块中观看选择的视频文件。

录像回放模块的具体实现步骤如下。

(1) 新建一个 Windows 窗体，命名为 frmResvice.cs，主要用于实现查看录制的视频文件功能。

(2) 单击“回放”按钮，设置要打开文件的格式及默认路径，然后打开“选择视频文件”对话框，选择完视频文件后，将选择的视频文件赋值给 axWindowsMediaPlayer1 控件的 URL 属性。

“回放”按钮的 Click 事件代码如下：

```
private void btnPlay_Click(object sender, EventArgs e)
{
    ofDialog.Filter = "*.avi|*.avi";
    ofDialog.Title = "选择视频文件";
    ofDialog.InitialDirectory = Application.StartupPath.Substring(0,
        Application.StartupPath.LastIndexOf(@"\")).Substring(0,
        Application.StartupPath.Substring(0,
        Application.StartupPath.LastIndexOf(@"\")).LastIndexOf(@"\")) + @"\Video\\";
    if (ofDialog.ShowDialog() == DialogResult.OK)
    {
        this.axWindowsMediaPlayer1.URL = ofDialog.FileName;
    }
}
```

(3) 单击“关闭”按钮，调用 Close 方法关闭当前窗体。

“关闭”按钮的 Click 事件代码如下：

```
private void btnClose_Click(object sender, EventArgs e)
{
    this.Close();
}
```

## 12.8 项目调试

视频讲解 光盘：视频\第 12 章\项目调试.avi

(1) 编译运行后的用户登录界面如图 12-8 所示。

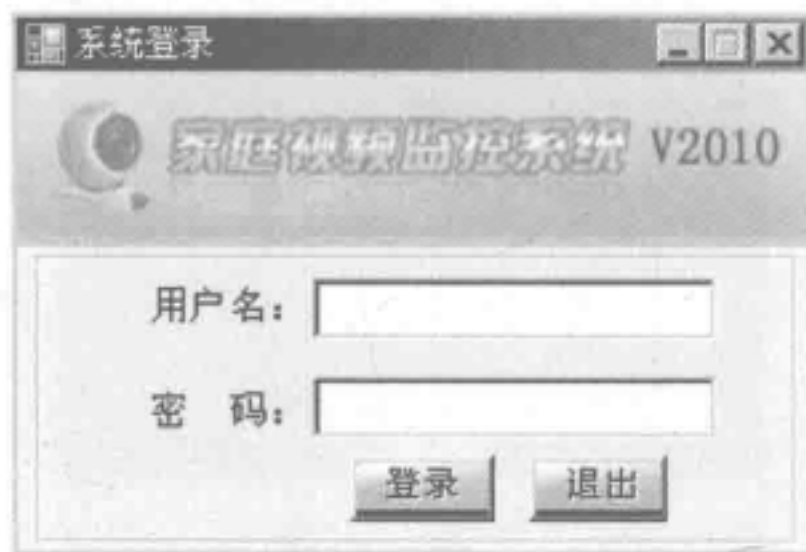


图 12-8 用户登录界面的效果

(2) 监控主界面如图 12-9 所示。



图 12-9 监控主界面的效果

(3) 监控管理界面如图 12-10 所示。

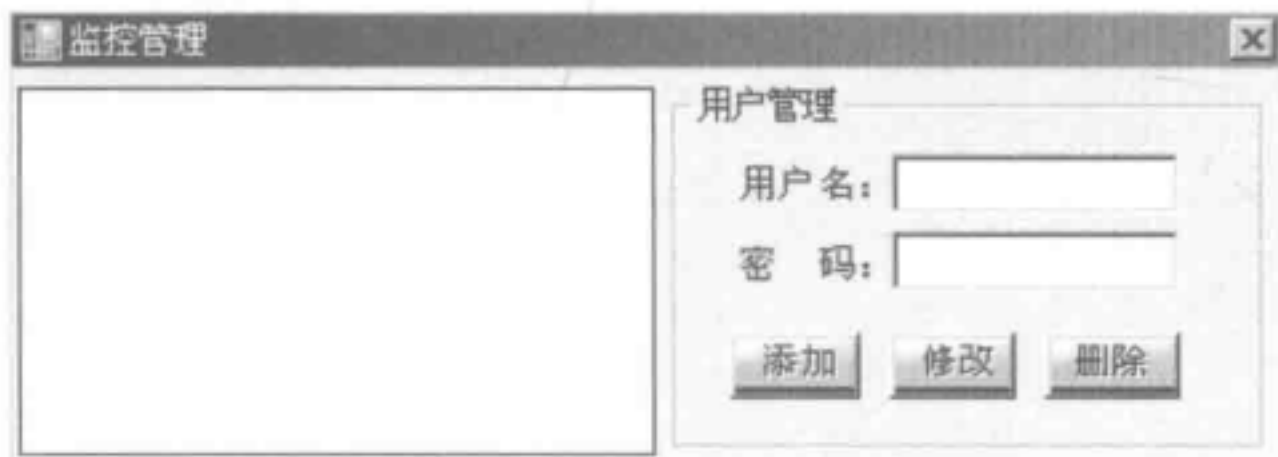


图 12-10 监控管理界面的效果

(4) 录视频界面如图 12-11 所示。



图 12-11 录视频界面的效果