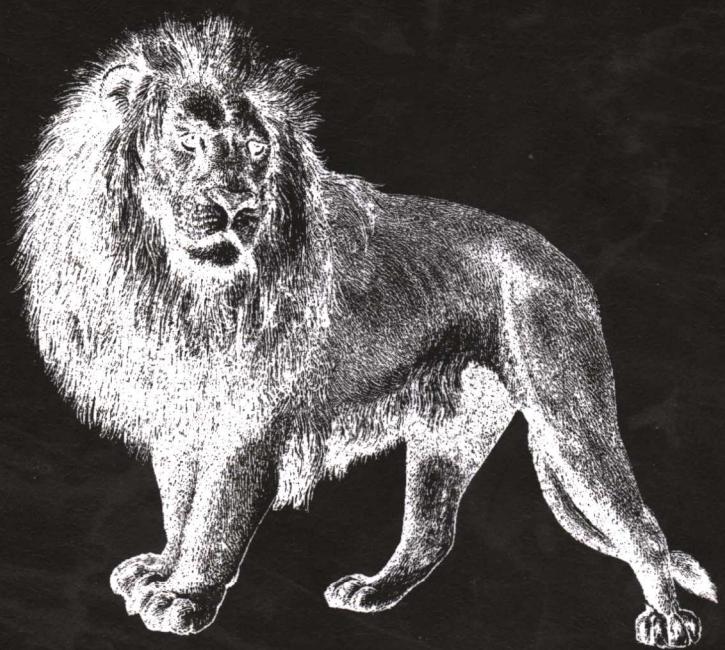
软·件·工·程·师·典·藏

超量代码·实用工具

Visual C++ 视频技术方案宝典

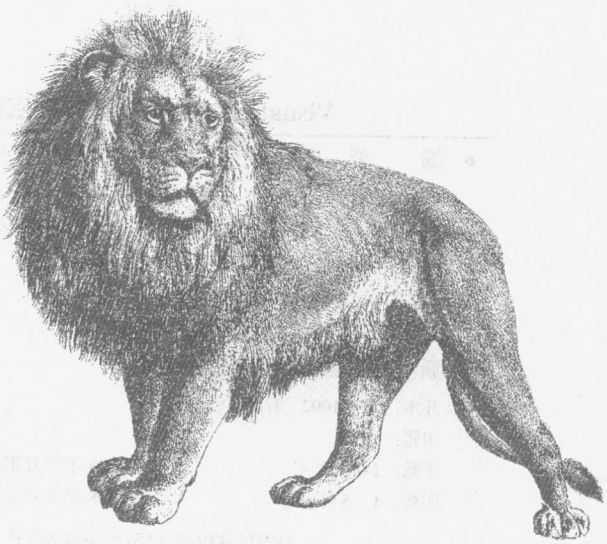
■ 明日科技 宋坤 刘锐宁 马文强 编著



人民邮电出版社
POSTS & TELECOM PRESS

Visual C++ 视频 技术方案宝典

■ 明日科技 宋坤 刘锐宁 马文强 编著



人民邮电出版社
北京

图书在版编目（CIP）数据

Visual C++视频技术方案宝典 / 宋坤，刘锐宁，马文强编著. —北京：人民邮电出版社，2008.2
（软件工程师典藏）
ISBN 978-7-115-17364-5

Ⅰ. V… Ⅱ. ①宋…②刘…③马… Ⅲ. C语言—程序设计
Ⅳ. TP312

中国版本图书馆CIP数据核字（2007）第196752号

内 容 简 介

本书从使用Visual C++进行视频开发所必须掌握的核心技术入手，通过各种实用方案深入介绍各种核心技术在实际开发中的应用。全书分为9章，分别是系统设计方案、视频采集方案、视频编码方案、音频采集方案、数据压缩方案、数据传输方案、数据安全方案、多媒体接收与显示方案、系统发行与维护方案等。通过本书的学习，读者不但可以掌握Visual C++视频开发必须掌握的各种核心应用技术，更能触类旁通，学以致用，领会应用开发的精髓。

本书附有配套光盘。光盘提供了书中所有实例的全部源代码，所有实例都经过精心调试，在Windows XP/2003下全部通过，保证能够正常运行。

本书内容详实，突出技术本质，具有非常强的实用性。适合于各级软件开发人员学习使用，也可供大、中专院校师生学习参考。

软件工程师典藏

Visual C++视频技术方案宝典

◆ 编　　著　明日科技　宋　坤　刘锐宁　马文强
责任编辑　屈艳莲

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：39.75
字数：1 083千字　　2008年2月第1版
印数：1－5 000册　　2008年2月北京第1次印刷

ISBN 978-7-115-17364-5/TP

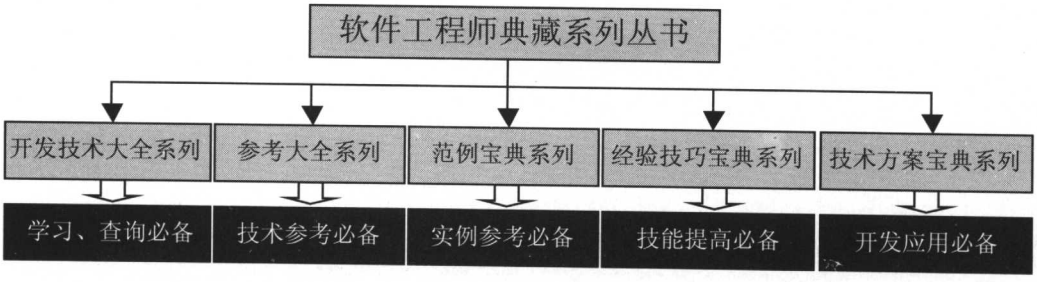
定价：75.00元（附光盘）

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

丛 书 序

随着信息技术的不断发展，软件业已经成为我国经济发展的战略性产业，社会上急需大量的 IT 人才以满足发展的需求，其中熟练进行软件开发的程序员更得到企业的青睐。但因软件需求千变万化，代码实现也是千差万别，要想成为一名合格的开发人员，通常需要较长时间的学习和历练。缺乏实际开发经验仍是横亘在众多编程人员面前的“鸿沟”。如果能将实际开发经验和技能传授给他们，无疑会对软件开发人员的成才起到催化剂的作用。所以，为软件开发人员量身打造的、专业全面的 “软件工程师典藏”系列丛书就应运而生了。

一、丛书框架



- **技术大全**

知识系统全面、易学易查、实例丰富，是软件开发人员学习和备查的理想参考书。

- **参考大全**

深入解析开发中的技术细节，对每一项专题技术都给出了深入的解析和实用的例子，是开发人员不可缺少的技术参考书。

- **范例宝典**

每本书精选数百个常用的编程范例，不但能解决开发中的常见问题，更可帮助读者提升编程水平。

- **经验技巧宝典**

每本书收录了 1000 多条经验技巧，这些经验技巧都是开发团队在长期的实践中不断积累的。提供给读者随查随用，帮助读者少走弯路、拓展编程思路。

- **技术方案宝典**

实际应用方案剖析核心技术在开发中的各种实际应用，是软件开发人员必备的参考技术方案。

二、丛书特点

1. 给出语言学习的整体解决方案

本套丛书覆盖了绝大部分主流语言和工具，可以满足不同读者学习之需，也非常适合专业开发人士参考。

2．积淀一线编程高手的经验技巧

丛书作者均具有多年的实际开发经验，他们从程序开发的不同角度为开发人员提供全面、实用的开发理念，从而保证内容不再是演示性的实例，最大程度地贴近了实战。

3．完善的答疑服务

丛书作者将通过网站、论坛、热线电话等方式为读者提供疑难解答，随时解决书中遇到的各种技术问题。

感谢参与本丛书写作的每位作者，有他们的不懈努力和辛勤工作，本系列丛书才能顺利推出。愿本丛书能成为广大软件开发人员的良师益友。

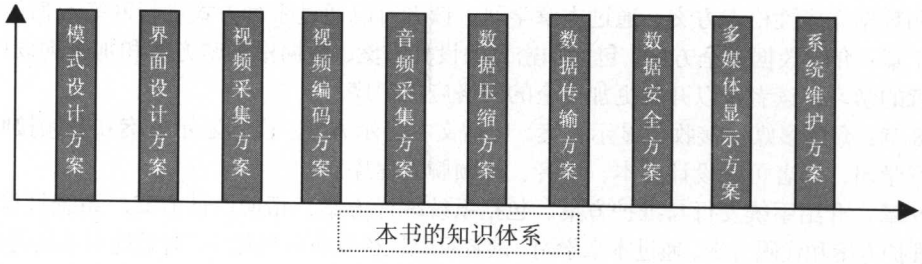
本书编委会

2007 年 12 月

前 言

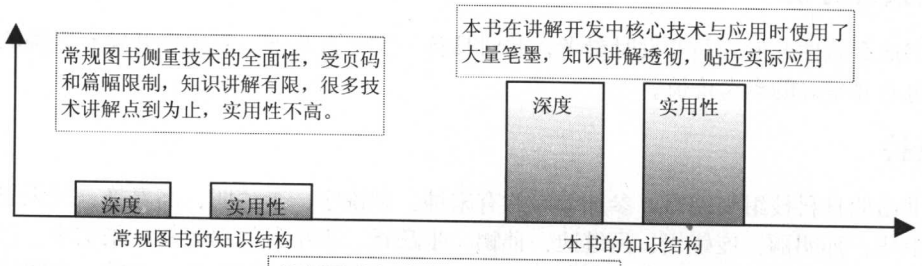
Visual C++ 6.0 是 Microsoft 公司推出的基于 Windows 环境的一种面向对象的可视化编程环境。因其功能强大、代码执行效率高、深入地植根于 C\C++语言，深受广大开发人员的喜爱。此外，从底层的驱动程序开发，到应用层的数据库、网络程序开发，再到网络游戏设计、图形\图像处理、多媒体应用等众多领域都存在它的身影，使得 Visual C++ 6.0 成为世界上使用最广泛的程序开发工具之一。

目前，虽然介绍 Visual C++ 6.0 的图书众多，但绝大多数是为初级用户介绍基础知识的入门书籍。真正从应用角度讲解 Visual C++ 6.0 实际开发与应用的图书不多。本书从 Visual C++ 6.0 进行视频开发必须掌握的核心技术出发，将实际开发中与该技术相关的应用整理成方案，通过各种实际开发方案讲透各种技术的具体应用，真正授人以渔，满足实际开发者的需求。读者学完本书后，可以顺利进行实际项目的开发。

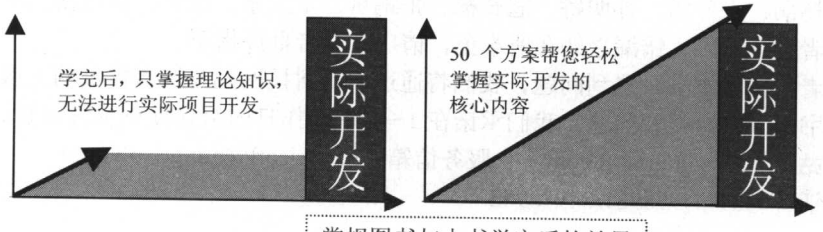


本书的知识体系

下面列出了常规图书和本套图书在知识结构和学习效果上的比较。



常规图书与本书知识结构的比较



常规图书与本书学完后的效果

本书内容

本书主要以 Visual C++视频开发中必须掌握的核心技术入手，深入探讨开发中必须掌握的

技术和技术的各种应用方案，使读者不但可以了解知识的来龙去脉，更能学会利用学到的知识进行实际开发。本书共分 9 章，各章内容如下。

第 1 章：介绍开发模式技术及相关方案，主要包括设计模式与架构分析、数据库设计方案、界面设计方案等。学好本章，可以选择合适的开发模式，从而使程序设计更合理，代码更清晰，更容易维护。

第 2 章：介绍视频采集技术及相关方案，包括传统的 VFW 视频采集技术、Direct Show 视频采集技术、天敏 VC4000 SDK 视频采集技术和云台控制技术。通过本章学习，读者可以透彻掌握视频采集的相关技能，设计优秀的视频采集应用程序。

第 3 章：介绍视频编码技术及方案，包括 H263 编码方案、H264 编码方案和 MPEG-4 编码方案。通过本章的学习，读者能够了解各种视频编码技术，在开发视频程序时选择适当的编码方案。

第 4 章：介绍音频采集和编码方案。包括 MCI 音频采集方案、WaveForm 音频采集方案、Direct Show 音频采集方案和 MPEG 音频编码方案。通过本章学习，读者可以根据实际情况选择适当的音频采集方案进行音频程序开发。

第 5 章：介绍数据压缩技术及方案。包括 JPEG 数据压缩方案、AVI 数据方案和 ACM 音频压缩方案。通过本章学习，读者可以轻松对多媒体数据进行压缩。

第 6 章：介绍数据传输方案。包括 TCP 数据传输方案、UDP 数据传输方案、数据广播传输方案和数据完成性检查方案。通过本章学习，读者可以通过多种方式进行网络数据传输。

第 7 章：介绍数据安全方案，包括加密算法设计方案、数据报加密方案和加密狗设计方案。通过本章的学习，读者可以开发更加安全的网络应用程序。

第 8 章：介绍多媒体接收与显示方案。包括文本显示方案、视频显示方案和音频接收方案。通过本章学习，读者可以设计文本、语音、视频聊天程序。

第 9 章：介绍系统发行与维护方案。包括系统编译方案、帮助设计方案、系统打包方案、数据库维护方案和代码方案。通过本章学习，读者可以轻松发布应用程序，有效地对系统进行维护。

本书的读者对象

本书适合软件开发人员开发时参考，并对具备一定语言基础，对程序开发有兴趣的编程爱好者，也有非常好的参考价值。

技术支持

本书由明日科技组织编写，参加编写的有宋坤、刘锐宁、李伟明、高春艳、邹天思、潘凯华、刘中华、孙明丽、庞娅娟、吕继迪、孙鹏、张跃廷、王小科、房大伟、张宏宇、吕双、苏宇、梁水、张言、梁冰、刘彬彬、安剑、王斌、王茜、孙秀梅、刘玲玲、刘欣、梁晓岚、顾彦玲、黄锐、杨丽、王冬雪、孙明娇、寇长梅、张鹏斌、董大永、张艳、郭佳博、乔敏、赛奎春等。由于作者水平有限，错漏之处在所难免，请广大读者批评指正。

如果读者在使用本书时遇到问题，我们将通过明日科技网站为读者提供网上服务和支持。读者使用本书遇到的错误和问题，我们承诺在 1～6 个工作日内给您提供及时回复。

服务网站：www.mingrisoft.com　　服务信箱：mingrisoft@mingrisoft.com

服务电话：0431-84978981/84978982

明日科技

2007 年 12 月

目 录

第 1 章

系统设计方案

- 面向对象设计方案
- 系统建模设计方案
- 界面设计方案

系统设计在软件开发过程中占居非常重要的位置，系统设计的好坏直接影响到系统功能、系统维护。一个优秀的系统设计方案不但能够满足用户的需求，还能够易于扩展，在用户需求改变时，能够以最小的变动适应用户需求。

1.1 面向对象设计方案

传统的程序开发采用结构化的程序设计方法，即面向过程。针对某一需求，自上而下，逐步细化，将需求通过模块的形式实现，然后对模块中的问题进行结构化编码。可以说，这种方式是针对问题求解。随着用户需求的不断增加，软件规模越来越大，传统的面向过程开发方式暴露出许多缺点，软件开发周期长，工程难于维护。20 世纪 80 年代后期，人们提出了面向对象（Object Oriented Programming）的程序开发方式，简称 OOP。面向对象不仅是一种技术，而且是一种思想，它将客观世界中的事物描述为对象。对象具有属性和方法，例如，人具有手、脚、头发等，这些都是人的属性，此外，人能够说话、走路，这些是人的方法。在 OOP 中，将对象的内部处理细节封装在模型内部，对外界提供接口，对用户来说，它只需要知道如何使用接口完成某一功能就可以了，并不需要了解对象内部是如何运作和实现的。

1.1.1 用户界面与业务逻辑的分离

在开发应用程序时，用户界面与业务逻辑分离具有许多好处。例如，它能够使界面的改变不至于影响到业务逻辑，还能够使业务逻辑实现代码重用。著名的 MVC（Model-View-Controller）模式提出了用户界面与业务逻辑分离的设计方案。它将应用程序分为 3 层，模型（Model）、视图（View）和控制器（Controller）。

模型负责处理应用程序的业务逻辑，它是业务流程的处理过程。模型对于其他层来说是一个黑盒子，它接受视图传来的数据，并返回处理结果。从模型的实现角度看，模型主要包含业务实体对象和业务处理对象。业务实体对象描述了数据表的内容，业务处理对象封装了具体的逻辑操作，它能够读写业务实体对象。

视图代表与用户交互的界面，对于 Windows 窗体应用程序来说，可以表示为窗体。它只负责用户数据的输入和显示，而不进行业务处理。

控制器负责整合模型和视图，以完成用户的需求。在 Windows 窗体应用程序中，控制器通常与视图合并在一起，例如，MFC 中的文档/视图结构，文档对象负责保存数据，可以将其理解为模型，而视图对象负责读取和显示文档数据，可以认为是视图和控制器。

下面以员工信息管理为例介绍在 Visual C++中如何实现用户界面与业务逻辑的分离。员工信息管理用例视图如图 1.1 所示。

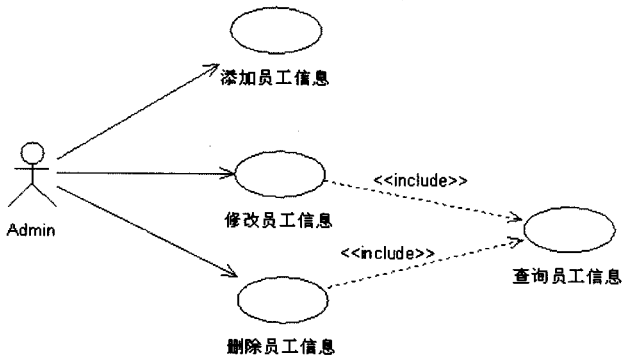


图 1.1 员工信息管理用例视图

为了实现用户界面与业务逻辑分离，需要确定模型。从图 1.1 中可以发现模型中的实体对象——员工。至于业务处理对象，它需要实现图 1.1 中的所有方法，其类图如图 1.2 所示。

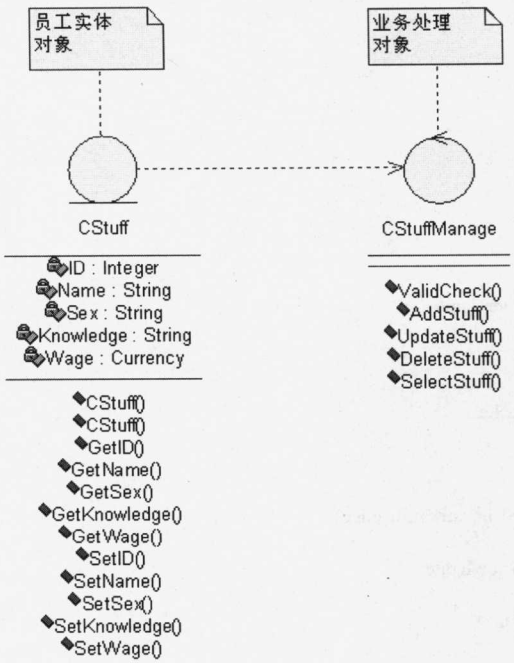


图 1.2 用户信息管理类图

在图 1.2 中，CStuffManage 类实现了对实体对象的添加、修改、删除、查询等操作。当用户需要实现员工信息的添加时，它只需要提供员工信息，并调用 CStuffManage 类的 AddStuff 方法就可以了，至于 AddStuff 方法是如何实现员工信息添加的，外界并不需要知道。CStuff 类及 CStuffManage 类代码如下。

实例位置：光盘\mr\1\1.1\1.1.1\01

```
//员工信息实体
class CStuff
{
public:
    CStuff();
    virtual ~CStuff();
private:
    int      ID;                //工号
    CString Name;               //名称
    CString Sex;                //性别
    CString Knowledge;          //学历
    float    Wage;              //工资
public:
    int GetID() const
    {
        return ID;
    }
    CString GetName() const
    {
        return Name;
    }
    CString GetSex() const
    {
```

```
        return Sex;
    }
    CString GetKnowledge() const
    {
        return Knowledge;
    }
    float GetWage() const
    {
        return Wage;
    }

    void SetID(int uID)
    {
        ID = uID;
    }
    void SetName(CString strName)
    {
        Name = strName;
    }
    void SetSex(CString strSex)
    {
        Sex = strSex;
    }
    void SetKnowledge(CString strKnowledge)
    {
        Knowledge = strKnowledge;
    }
    void SetWage(float fWage)
    {
        Wage = fWage;
    }
};
//员工信息管理类
class CStuffManage
{
public:
    void SelectStuff(_RecordsetPtr pRecord);
    BOOL DeleteStuff(CStuff& stuff);
    BOOL UpdateStuff(CStuff& stuff,CStuff &oldstuff);
    void IniStuff(CStuff& stuff);
    BOOL AddStuff(CStuff& stuff);
    BOOL ValidCheck(CStuff& stuff);
    CStuffManage();
    virtual ~CStuffManage();
};
//合法性检查
BOOL CStuffManage::ValidCheck(CStuff &stuff)
{
    if (stuff.GetKnowledge().IsEmpty() || stuff.GetSex().IsEmpty()
        || stuff.GetName().IsEmpty()||stuff.GetWage()<0)
        return FALSE;
    else
        return TRUE;
}

//添加员工信息
BOOL CStuffManage::AddStuff(CStuff &stuff)
{
    CString sql;
    sql.Format("insert into tb_StuffInfo values (%i,
     '%s','%s','%s',%f)",stuff.GetID(),stuff.GetName(),
        stuff.GetSex(),stuff.GetKnowledge(),stuff.GetWage());
```

```
    try
    {
        m_DataManage.ExecOpt(sql);
        return TRUE;
    }
    catch(...)
    {
        return FALSE;
    }
}
//初始化员工信息
void CStuffManage::IniStuff(CStuff &stuff)
{
    stuff.SetID(0);
    stuff.SetName("");
    stuff.SetKnowledge("");
    stuff.SetSex("");
    stuff.SetWage(0.0);
}

//修改员工信息
BOOL CStuffManage::UpdateStuff(CStuff &stuff,CStuff &oldstuff)
{
    CString sql;
    sql.Format("update tb_StuffInfo set id = %i, name = '%s',sex = '%s',
      knowledge = '%s',wage = %f where id = '%s'",stuff.GetID(),stuff.GetName(),
        stuff.GetSex(),stuff.GetKnowledge(),stuff.GetWage(),oldstuff.GetID());
    try
    {
        m_DataManage.ExecOpt(sql);
        return TRUE;
    }
    catch(...)
    {
        return FALSE;
    }
}

//删除员工信息
BOOL CStuffManage::DeleteStuff(CStuff &stuff)
{
    CString sql;
    sql.Format("delete   tb_StuffInfo   where id = '%s'",stuff.GetID());
    try
    {
        m_DataManage.ExecOpt(sql);
        return TRUE;
    }
    catch(...)
    {
        return FALSE;
    }
}

//查询员工信息
void CStuffManage::SelectStuff(_RecordsetPtr pRecord)
{
    pRecord = m_DataManage.ExecSelect("select * from tb_StuffInfo");
}
```

员工信息管理模块的业务逻辑设计完成后，在用户界面中需要调用业务逻辑以实现相关操作。下面给出用户界面中添加员工信息的程序代码。

```
void CStuffSysDlg::OnSavefinfo()
{
    //提供员工信息
    UpdateData();

    m_Stuff.SetID(m_StuffID);

    m_Stuff.SetName(m_StuffName);

    CString sex;
    m_StuffSex.GetWindowText(sex);
    m_Stuff.SetSex(sex);

    CString knowledge;
    m_StuffKnowledge.GetWindowText(knowledge);

    m_Stuff.SetKnowledge(knowledge);

    m_Stuff.SetWage(m_StuffWage);

    //检查员工信息的合法性
    if (m_StuffManage.ValidCheck(m_Stuff))
    {
        //添加员工信息
        if (m_StuffManage.AddStuff(m_Stuff))
                MessageBox("员工信息添加成功");
        else
                MessageBox("员工信息添加失败");
        OnClearinfo() ;
    }
    else
        MessageBox("员工信息非法!");
}
```

许多读者可能会问，采用上面的方式开发程序需要编写大量的程序代码，是否值得？对于开发专业的应用程序，如果将用户界面与业务逻辑混在一起，会出现许多问题。首先代码维护困难，业务代码的修改会牵连到与界面有关的代码。其次，程序不灵活，没有可重用的价值。例如，在一个按钮的单击事件中编写几百行的逻辑代码，代码的可读性和维护性难以想象。

1.1.2 数据抽象与封装

在开发面向对象应用程序时，一个比较重要的问题是如何设计类。设计类的过程也就是封装类的过程。所谓封装，是面向对象程序设计的主要特征之一，它将数据和处理过程结合在一起，并提供接口供用户访问。封装实际是一种抽象机制，它将事物的属性抽象为类的数据，将事物的功能抽象为类的方法。

封装的目的是隐藏实现细节，增强系统的灵活性和可维护性，因此在封装类时应遵循一定的原则。一个通用的基本原则是高内聚低耦合。所谓内聚是指模块内的类、数据和方法相互依赖的一种紧密关系。一个高内聚的模块，在修改其模块的代码时不会影响到其他模块，因此一个程序模块的内聚力越高越好。所谓耦合是指多个系统之间通过相互作用而彼此影响的现象。模块间的耦合度越弱越好，这样可以避免在修改某个模块时导致“动一发而牵全身”的结果。

此外，在设计类时还应该遵循一些设计模式。采用适当的设计模式能够使系统架构更灵活。例如，在设计图书销售管理系统时，对于不同种类的图书，其折扣方式会有所不同，有的图书有不同的折扣、有的图书没有折扣等。如果在客户端利用条件选择语句逐一判断，会导致客户端的代码复杂并难以维护。最好的方式是采用策略模式。所谓策略模式是指将一组算法封装到独立的类中，然后从这些类中抽象出公共接口，从而使得这些类可以相互代替。图 1.3 所示是