数据库**通用模块**开发与**系统移植**丛书

案例开发多媒体讲解+源代码

Visual C# 2005

数据库通用模块开发与系统移植

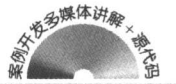
吴 晨 胡书敏 蔡 威 编著

本书主要内容：

- C#数据库访问模块的架构
- ADO.NET、XML和存储过程高级应用技术
- 通用登录模块
- 邮件发送管理模块
- 数据报表模块
- 图片管理系统模块
- 在线投票模块
- 信息发布模块
- 创建和发布自己的Blog网站
- 在线购物系统
- 新生报到注册系统
- 数据库系统的移植与升级

清华大学出版社

数据库通用模块开发与系统移植丛书



Visual C# 2005
数据库通用模块开发与系统移植

吴 晨　胡书敏　蔡 威　编著

清华大学出版社
北　京

内 容 简 介

Visual C# 2005 是 Microsoft 公司开发的新一代编程语言包，由于其内嵌在.NET Framework 中，所以不仅包含了 ADO.NET 这一功能强大的数据库开发组件，而且更具有“显示逻辑同业务逻辑分离”这一特性。由于其具有功能强大，使用方便的特点，已成为数据库编程必不可少的工具。

本书以面向对象思想和设计模式为指导，通过大量实例，详细介绍了 ADO.NET、基于数据库应用的通用模块、数据库系统综合实例设计和开发、数据库的移植升级、项目打包发布和安装等方面的应用技术，内容包括数据库开发的诸多技术和数据库模块的设计理念，用户登录、邮件管理、报表设计、图片管理、在线投票信息发布和管理等通用功能模块，Blog 和在线购物项目案例，C#应用程序的打包发布和安装，数据库以及项目案例的升级。

本书不仅适用于使用 Visual C# 2005 进行软件开发的具有编程经验的广大软件开发人员，也适合高等院校师生学习和参考使用，特别对高等院校计算机及相关专业的学生进行毕业设计具有非常好的参考价值，也可以作为广大计算机编程爱好者的自学与参考用书。

图书在版编目(CIP)数据

Visual C# 2005 数据库通用模块开发与系统移植/吴晨，胡书敏，蔡威 编著. —北京：清华大学出版社，2007.5
(数据库通用模块开发与系统移植丛书)
ISBN 978-7-302-15139-5

Ⅰ.V… Ⅱ.①吴… ②胡… ③蔡… Ⅲ.C 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2007)第 060794 号

责任编辑：王　定　鲍　芳
封面设计：久久度文化
版式设计：康　博
责任校对：胡雁翎
责任印制：孟凡玉
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 刷 者：北京嘉实印刷有限公司
装 订 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：195×260　**印　张**：31　**字　数**：774 千字
附光盘 1 张
版　　次：2007 年 5 月第 1 版　　**印　　次**：2007 年 5 月第1 次印刷
印　　数：1～5000
定　　价：48.00 元

PREFACE

前　言

Visual C# 2005 是 Microsoft 公司开发的最新一代编程语言包，由于其内嵌在.NET Framework 中，所以程序员在用其开发应用程序时，一方面能方便地使用.NET Framework 中的 ADO.NET 组件，高效快捷地开发基于数据库访问的应用代码，另一方面能利用其“显示逻辑同业务逻辑分离”的良好编程特性，从架构的层次上，分离窗体类的代码和业务功能类的代码，从而提高所开发项目的可维护性和可重用性(即可移植性)。

另外，从软件工程的角度来讲，由于在开发新项目时，要尽可能地重用现有的功能模块，所以在开发具有各功能的业务模块时，要尽量把其设计并构造成一个通用的组件。并且，由于应用项目会时常面临需求变更以及功能维护的工作，所以，在编写应用项目的代码前，有必要在分析其潜在变更点的前提下，合理地设计项目的架构以及功能模块的布局，为项目提供架构层次的维护和扩展空间。

本书在合理利用 Visual C# 2005 编程语言的优势基础上，以数据库应用技术为主线，以面向对象和设计模式等优秀设计理念为指导，综合分析 C#数据库应用方面的知识和实例。本书共分为 13 章，各章内容如下：

第 1 章主要介绍了面向对象思想的各要素，以及如何根据面向对象思想设计 C#的数据库访问模块的架构。

第 2 章围绕 ADO.NET 中的两种数据提供者(Data Provider)，在详细介绍其中 Connection、Command、DataAdapter、DataReader 和 DataSet 类的基础上，综合讲述基于 DataGrid 控件的 ADO.NET 综合应用技术。

第 3 章在 ADO.NET 组件的基础上，除了讲述 XML 和存储过程等数据库高级应用技术外，还结合数据库应用代码，给出能指导项目设计的设计模式原则。特别是 DAO 模式更能提高代码可移植性和重用性。

第 4~9 章详细介绍基于数据库应用的通用功能模块。其中，第 4 章介绍了用户登录模块，第 5 章介绍了邮件发送管理模块，第 6 章介绍了生成水晶报表和 Excel 报表模块，第 7 章介绍了基于 Windows 和 Web 的两种版本的图片管理模块，第 8 章介绍了在线投票模块，第 9 章介

绍了信息发布和管理模块。请读者们注意这些模块中的数据库访问代码。在第 4 章是直接把 ADO.NET 写在业务逻辑里，经不断改进，在第 9 章则是根据 DAO 原则设计通用的数据库访问模块，读者能从这个过程中体验到设计模式给项目开发带来的便利。

第 10 章和第 11 章分别讲述了 Blog 和在线购物系统这两个综合应用项目，其中不仅包含了面向 ADO.NET 的数据库应用，而且重用了前文给出的各功能模块。

第 12 章以一个基于 Windows 的项目为例，讲述 C#项目的打包、发布和安装的详细步骤。

第 13 章模拟客户对在线购物系统提出的“数据库迁移”、“功能升级”和“模块维护”等要求，根据“代码重用”和“架构维护”等原则，通过实例讲述代码升级、数据库迁移及项目维护的实施理念和步骤。

本书不仅注重模块的通用性和代码的实践性，更着重分析了根据面向对象思想和设计模式，来优化模块的设计和提高代码质量的目的、思考方式与实现步骤。本书的读者对象可以是具有 Visual C#编程经验的软件开发人员，也可以是高等院校的师生。特别对高等院校计算机及相关专业的学生进行毕业设计具有一定的参考价值。此外，本书也可以作为广大计算机编程爱好者的自学和参考用书。

本书由吴晨、胡书敏和蔡威执笔，在编写本书文稿和代码的过程中，得到了大连理工大学孟宪福先生的大力支持和指导，在此表示衷心的感谢。此外，还要感谢上海师范大学的梅晓东同学和上海畅想电脑有限公司的奚亦扬先生，他们参与了校对本书全部文稿和代码的工作。再者，蒋忠、沈学兵、韩建国、杨树平、孔凡、曹丽、严明志、华剑、朱伯芳、秦智育、尤鲲、许杨道、赵小燕、钱云、孙志新等同志在本书的编写过程中也做了大量工作，在此也表示感谢。

由于时间仓促，加之作者水平有限，书中不足之处在所难免，敬请读者批评指正。

编　者

2007 年 3 月

CONTENTS

目　录

CHAPTER 01

面向对象思想与C#数据库开发

面向对象思想并不是高深的理论，而是根据大量编程项目总结出来的一套分析问题和解决问题的方法。

以面向对象思想为指导，可以优化用 Visual Studio.NET 开发出来的代码，可以改善数据库访问模块的结构，更可以让数据库访问模块变得更有弹性—— 即让数据库访问模块能更好地适应项目需求的变化，从而大大提高代码的重用率，缩短开发周期。

1.1 面向对象思想的需求背景

1.1.1 编程语言发展历程

计算机语言刚出现时，采用的编程语言是机器语言，即用一大堆机器能识别的 0 和 1 来编写代码。这种编码方式需要很高的技巧，调试起来也比较困难。

为了提高编程语言的适用性，软件语言方面的专家们开发了以编程语言为代表的低级语言，即用一些助记符来代替晦涩的机器语言，也就是汇编语言。例如用 add a b，来计算 a+b 的值。用这种编程语言开发出来的代码有一个重大缺陷：由于在此类代码里大量充斥着 goto 类型的语句，所以结构性很差。例如，实现以下功能代码

```
if(i>10)
{   i = i-10;   }
else {   i = i + 10;  }
后继代码
```

的编写方式是：

```
0   if(i>10) goto 20
10  else goto 30
20  i = i + 10 goto 40
30  i = i - 10 goto 40
40  后继代码
```

从中可以看到，大量的 goto 语句导致了代码不可读和不可维护，而不可读和不可维护将大大增加代码后期的扩展和维护成本，并降低代码的逻辑，给开发带来了困难。为了解决这类问题，提高代码的逻辑性和可读性，软件语言方面的专家提出了结构化的面向过程语言。

结构化语言的重要特征是分离了代码逻辑和程序数据，并使用了分支(if)和循环(while 和 for)等分支结构来优化代码的逻辑结构。由于结构化的语言使用子模块顺序调用的方式控制代码流程，所以又叫面向过程语言，以 C 语言为代表。

和汇编语言相比，面向过程语言能在很大程度上改善代码结构，提高代码的逻辑性。具体表现为以下两点：

- 采用了函数的封装形式，这对项目的二次开发非常有利。
- 摒弃了使用 goto 语句的跳转方式，代码的结构得到大幅度的改善。

1.1.2 面向过程语言的缺陷

在使用过程中，程序员们发现面向过程的语言有本质上的缺陷，具体表现为：

- 在模块间调用函数时，由于无法屏蔽模块中不希望被修改的关键变量，这将导致模块间有很高的耦合度。
- 代码的重用级别仅仅是局限于函数级别的，这导致无法大量有效地利用已开发完成的代码成果，对代码的重复性利用非常不利。
- 函数的定义是针对具体动作和具体对象的，也就是说，面对业务逻辑相同但业务逻辑处理对象不同的需求，必须要写成多个不同名的函数。这导致了代码过于注重细节，而无法考虑逻辑性、重用性和维护性等大局方面的问题。

为了解决上述问题，软件语言的专家们提出了面向对象的指导思想。在结构化设计中，抽象的单位是函数；而在面向对象设计中，抽象的单位是对象。这种抽象能力可以让程序员在更高层次上对问题进行考虑。

1.2 面向对象思想概述

面向对象思想的三大要素是封装、继承和多态。封装机制屏蔽了对象本身的业务细节，能降低模块之间的耦合性，继承机制能实现类的重用，而多态机制能把具有相同业务类型的方法归结成同一个方法，从而能分离方法的调用接口和实现细节。

这三大要素并不能割裂，综合应用这三大要素，能很好地改善代码的结构，并消除由硬编

码(面向过程性编码，也可叫作抽象性编码，即没采用面向对象思想而写的代码)造成的效率低下。

1.2.1 封装与细节屏蔽

类是面向对象思想的基础，由于在类中封装了具有相同逻辑意义的数据变量和成员方法(也叫成员函数)，所以类可以说是对某一类具体事物的归纳和抽象。在面向对象的编程语言里，类也是代码重用的基础，在项目开发中，一般把众多类的共通方法和属性放在基类里，通过继承基类，子类可以在基类的基础上实现扩展，达到减少编码工作量的问题。C#使用 class 关键字来实现类的定义。定义类的格式是：

```
[类修饰符] class 类名
{    类的内容    }
```

其中，类修饰符可以是 abstract、public、protected、private、internal 和 sealed 等。而类的名字是一定要写的，如果不填类的内容，那么这个类就是无意义的空类。

通过下面的代码，可以定义学生类，其中包含学生的两个私有信息、一个构造函数和四个公有方法。

```
public class Student  //定义类
 {
    //以下定义 student 的属性
    private string name;
    private int age;
    //构造函数
    public Student(string input_name,int input_age)
    {
        name = input_name;
        age = input_age;
    }
}
```

这个类对学生对象进行了抽象，封装了学生的常用属性和方法。通过上面这段代码，来学习一下关于类的一些概念和使用方式。

1. 类和实例

外部代码(即类外面的代码，下同)可以通过 new 方法来创建类(也叫实例化类)，当调用 new 函数时，系统会自动根据 new 方法的参数，调用对应的类构造函数，或是系统默认的类构造函数。

比如，可以通过以下的语句来创建学生类：

```
Student s = new Student("Mike",12);
```

其中，Student 是类的类型，而 s 是使用 Student 类型创建的实例(也叫对象)。对象是程序在运行时其中的任何一个实体，而类则是查看程序清单时存在的一个静态实体。对象是在程序运行时具有特定值和属性的动态实体。例如，可以定义一个具有名字、年龄、性别等属性的人，而在运行时则会遇到 Peter、Mike 等人，也就是说，对象是类的特例。一般是对实例对象进行操作，而不是对类型进行操作。

2. 访问修饰符

如果类之间能相互访问对方的内部实现细节，比如私有属性，不仅会降低类的安全性，还将干涉其他类的私有业务逻辑。并且，如果类之间的耦合关系过于复杂，那么，修改其中的一个类的时候势必将会带动修改其他逻辑上没有关系的其他类，这对项目的需求变更、功能扩展和维护是相当不利的。

在 C#语言里，使用访问修饰符可以提高类的安全性，并降低类之间的耦合性。

使用 public、protected、private 和 internal 四个访问修饰符可以定义类以及它的方法和属性的可见范围。

- public 类型的数据在类的内部、类的子类里和类的外部都可以访问。
- protected 类型的数据在类的内部和类的子类里可以访问。
- private 类型的数据只能在类的内部被访问。
- internal 类型的数据在同一工作集(比如是同一项目)中能被访问。

一般来说，要控制类内部的变量，所以变量一般设置成 private，外部程序可以通过 get 和 set 类型的方法访问和设置数据。而类提供给外部的方法一般是 public 的，如果某些方法只希望在内部使用，那就需要把它们设置成 private。

3. 抽象类(abstract)和不可继承类(sealed)

类用来归纳一类事物的特性，而抽象类用来归纳抽象的概念，抽象类用 abstract 修饰符表示。抽象类既可以有实例变量，也可以同时有抽象方法和具体方法。在一个从抽象类到多个具体类的继承关系中，共同的代码应该尽量移动到抽象类里，在一个继承的等级结构中，共同的代码应向等级结构的上方移动。把重复的内容从子类里面移动到超类里面，可以提高代码的复用率。由于代码在共同的超类而不是几个子类中出现，在代码发生改变时，软件设计师只需要修改一个地方。这对代码的复用明显是有利的。

由于是对抽象概念进行归纳，所以其不必被实例化，因为用抽象类实例化出来的对象是概念，而不是实物，另外在其中的方法里，无须也无法定义具体的动作。抽象所带来的主要好处是可以忽略掉无关紧要的细枝末节问题，而专注于重要的特性。

在面向对象设计中，应尽力创造出与解决真实问题某一部分抽象程度相同的编程抽象，以便解决编程问题中的类似部分，而不是用编程语言实体来解决问题。

在 C#语言里，从语法上限制抽象类被实例化，也限制了抽象方法定义方法体。

抽象方法只有声明，没有方法体。例如，我们把动物的特性抽象到抽象类里，形成 animal

这个抽象类。代码如下：

```
public abstract class animal
{
    //定义动物呼吸的方法
    public abstract void breath();
}
```

在项目设计中，可以把相同特性类的共通方法和属性放到抽象类中，以达到代码重用的目的。

在设计中，有些类不希望被继承，或是从逻辑意义上来讲，不能被继承，这些类称作不可继承类，可以用 sealed 关键字来修饰。

不可继承类与抽象类相反，不可继承类主要用来定义具体的逻辑对象和业务动作。

4. 类与封装

在类的语法里，能体现类封装特性的特点有：

- 使用 private 关键字来屏蔽类内部的细节。
- 类通过 public 类型的方法，向外部提供类的功能。

例如，下面定义的电灯类拥有两个 private 的变量、一个获得电流的私有方法和一个开灯的公有方法。定义如下：

```
class light
{
    //电流值
    private int current;
    //电压值
    private int voltage;
    //从外界获得电流和电压的方法
    private void setValue()
    {
        // 获得电流和电压
        //设置 current 和 voltage
    }
    //向外界提供的开灯方法
    public void powerOn()
    {
        //调用本地的私有方法获得电流和电压
        setValue();
        //执行检查
        if(voltage > 220 || current>3)
        {
            //出错提示
        }
        else
```