HZ BOOKS

SAMS

Visual C# 2005 技术内幕

Microsoft Visual C# 2005 Unleashed

（美） Kevin Hoffman 著

李虎 许福 王晓博 等译

机械工业出版社
China Machine Press

Visual C# 2005 技术内幕

Microsoft Visual C# 2005 Unleashed

（美） Kevin Hoffman 著
李 虎 许 福 王晓博 等译

机械工业出版社
China Machine Press

本书提供了.NET框架下C#编程的详尽指南。书中详细介绍了.NET框架中的核心概念、使用GDI+编写高级用户界面、多线程程序设计、使用ClickOnce技术部署Windows窗体应用程序、创建智能客户端程序和使用Web服务，并详尽阐述了如何支持Ajax/Atlas风格的客户端回调技术，如何确保ASP.NET应用程序的安全性，以及使用新的ASP.NET提供者模型存取成员、大纲、站点地图、会话状态和角色信息，利用ASP.NET 2.0的Web 部件、主题和外观等新特征创建门户网站和个性化站点，还讨论了COM+、分布式应用、加密和安全保护等高级主题。本书内容丰富，讲解透彻，通过本书的学习，程序设计人员很容易就可以掌握C#编程的技巧。

本书版权登记号：图字：01-2006-3995

图书在版编目（CIP）数据

Visual C# 2005技术内幕/（美）霍夫曼（Hoffman, K.）著；李虎等译. －北京：机械工业出版社，2007.5

书名原文：Microsoft Visual C# 2005 Unleashed

ISBN 978-7-111-21210-2

Ⅰ. V…　Ⅱ. ①霍…　②李…　Ⅲ. C程序－程序设计　Ⅳ. TP312

中国版本图书馆CIP数据核字（2007）第041317号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：刘立卿

北京京北制版厂印刷 · 新华书店北京发行所发行

2007年5月第1版第1次印刷

186mm × 240mm · 32.25印张

定价：59.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

2004年11月7日，微软在美国旧金山正式发布了Visual Studio 2005，作为企业级软件生命周期管理及开发工具，Visual Studio 2005能够帮助广大开发人员及其团队开发出更加高效、安全和可靠的解决方案。Visual Studio 2005同SQL Server 2005等进行了紧密集成，打破了设计、测试、开发和项目管理之间的障碍，能够让开发者在分散的环境之间实现更好的协作，提供了高效安全的企业级应用程序开发平台。

Visual Studio 2005智能化集成的开发环境不仅提高了软件开发的效率，还降低了学习与掌握的门槛。Visual Studio 2005对.NET框架进行了全面升级，引入了静态与动态代码分析器，代码能够可视化表达并集成了单元测试工具与代码覆盖工具等。

本书对Visual Studio 2005集成环境下的C#语言及其软件开发技术进行了全面翔实的介绍。本书具有以下几个主要特点：

- 内容丰富实用。书中包含了C#语言、.NET框架、ASP.NET、ADO.NET、Web服务、分布式编程等方面的内容。既涵盖了最基本的C#语言程序设计知识，也介绍了Visual Studio 2005集成开发环境的特征和用法。不仅可满足Visual Studio 2005环境下一般程序设计人员的要求，也能满足想进一步提高应用技巧、解决疑难问题的高级程序设计人员的需求。
- 提供了大量应用实例。书中既有说明单个技术和函数调用的小程序，也有体现综合应用的较大的应用程序案例。
- 不仅讲解程序设计，也讲解软件设计。书中包括了大量与应用程序设计、开发有关的技巧、提示和注意事项，为读者设计大型的应用程序提供了良好的设计指导原则。

本书由李虎、许福、王晓博、范思怡负责主体翻译，全书由李虎统稿。此外，曹剑挺、宋淼、曹羽中、张军、万钧、郭荣锋等参加了本书的部分翻译工作。

李 虎

2007年1月30日

前　言

假定你强迫自己回忆10年之前的一些场景，想想那时技术的快速发展。在“对象链接与嵌入（OLE）”还很时髦的年代，就算走在繁华的大街上，或许也看不到有人在用移动电话，也不会看到有人紧张地敲打掌上电脑。你肯定看不到有人在网吧里用平板电脑将人工绘制的应用软件设计图发送给位于世界某个遥远角落的开发组。

在过去，为实现应用和应用构件之间的互联而进行的软件开发需要相当高超的技巧，要付出高昂的成本。确实，我们拥有了一些技术，如CORBA，但是它们只得到了有限的应用。

今天，“互联性”已经非常普及，它已经成为软件开发中的强制性需求。当人们在一家具有互联网设施的咖啡厅里打开掌上电脑时，他们完全有理由相信自己能够在这里上网。如果不能上网，他们就会表达对店主的不满。翻开新购置的手机的机盖，人们不仅希望这个手机能够拍照，而且希望它能够将所拍摄的照片上传到互联网的某个中央节点，以便让照片被朋友、家人乃至整个互联网络所共享。他们希望手机能够下载铃声、游戏，甚至音乐和视频，如果不能，他们就有“退货还钱”的念头。

互联性、性能和其他一些现代特征也是桌面软件和Web软件的强制性需求。人们希望他们的应用软件遵循“总能工作”这样一种准则，即使超负荷运转和恶意使用，也不用担心因为某种形式的数据丢失或损坏而带来不良后果。用户希望他们的Windows应用程序无论是否与互联网建立连接，都能正常工作。他们希望Web应用程序比以往有更快的响应速度，不仅看上去很棒，运行速度快，还希望它们记住他们喜欢什么，不喜欢什么，以及他们是谁。

简而言之，用户对软件质量、特征和功能所提出的严格要求是前所未有的。今天的程序员必须编写出比以往更强大的应用软件，并且软件的编写过程还要快速、可靠和成本低廉。

本书将引领软件开发者深入地探索运行在.NET框架2.0上的C#语言的最新版本Visual C# .NET 2005的特征、功能和能力。

通过阅读本书，你将学到以下内容：

- C#语言基础知识和.NET框架2.0的核心特征，例如面向对象程序设计、泛型和其他基本概念，包括委托和事件驱动的编程技术。
- 如何利用ADO.NET 2.0处理来自多个数据源的不同形式的数据，以及如何用C#编写SQL Server 2005的存储过程。
- 创建具有丰富用户界面的、响应速度快的Windows Forms应用程序。
- 通过鼠标点击部署来自Web或其他任何地方的Windows Forms应用程序。
- 创建能够通过Internet搜索和检查新版本和补丁程序从而动态改变自身的Windows Forms应用程序。
- 使用ASP.NET 2.0的新特征，创建ASP.NET 2.0应用程序。
- 利用主控页（master page）、主题（theme）和外观（skin）等新特征创建具有一致感官风格的页面，允许用户选择自定义风格主题的Web站点。

- 使用Web部件（Web Part）和Web部件页（Web Part Page）创建动态的、用户可控的内容页，其中包含了为特定用户或用户安全级别所定制的信息和功能。
- 利用ASP.NET中新的提供者模型（Provider Model），在SQL Server 2005中用15分钟甚至更少的时间创建能够鉴别用户身份、存储用户成员角色、存储用户profile数据以及存储用户个人信息（例如对Web部件的偏好）的应用程序。
- 在ASP.NET 2.0中利用客户端回调技术创建高度交互的、Ajax风格的用户界面。
- 创建可定制、可重用的ASP.NET 2.0控件。
- 创建和使用Web服务（Web Service）。
- 使用远程调用和/或COM+来创建分布式的、互联的应用程序。

目　录

第二部分 .NET框架2.0基础

第三部分 .NET 2.0中的数据访问

第四部分 开发ASP.NET 2.0 Web应用程序

第五部分 Web服务

第六部分 开发Windows Forms 2.0应用程序

第七部分 开发企业级和分布式应用程序

第一部分　C＃2.0基础

第1章　C# 2.0简介

本章将对C#做简要介绍。C#语言本身的语法很容易掌握。学习周期最长的，是如何使用C#语言在.NET框架中编写程序。在本章中，你将学习C#语言的基本语法，包括如何声明变量、编写程序块，当然，还包括用C#语言编写第一个应用程序“Hello World”。

使用过C#语言早期版本（1.0版或1.1版）的有经验的读者，可以跳过本章，继续学习后面各章。注意，请不要过多地跳过本书前面的一些章节，因为这些章节包含了C#语言中新增的和增强了的语法成分的介绍，譬如泛型（generics）和匿名方法（anonymous method）等。本章省略了一些细节，如果你对编程技术了解甚微的话，也许并不会注意到这一点。倘若你从来没有编写过任何Windows平台上（基于Windows或基于Web）的程序，那么本章对你来说，或许是错误的学习对象，甚至这本书对你来说都是错误的选择。

1.1　什么是.NET框架

在学习C#语言本身的机制之前，了解C#这类语言产生和发展的技术背景尤为重要。这一节首先对C#语言的发展情况做一个快速回顾，然后解释什么是公共语言运行时、公共类型系统，最后概述垃圾收集环境。

1.1.1　.NET的演化

在.NET框架公布于众之前，在Windows平台上所进行的面向构件的软件开发，大都采取COM构件的形式。组件对象模型（也称为构件对象模型，Component Object Model, COM）是用于创建可重用二进制构件的一个编程标准。按照这个标准，开发者可以编写较少的代码来解决较小的问题域中的问题。通过将问题分解为构件，从而简化解决方案，并且，用于解决某一问题的构件，也可用于解决其他类似问题。

COM在具有诸多优点的同时，也存在不少缺点。COM标准的最常见的一个问题，被业界比喻为“DLL地狱”。当COM接口在注册表中被登记和索引之后，如果又发布了COM构件的新版本，“DLL地狱”问题便会产生。由于COM标准与二进制构件的紧耦合性质，版本控制经常让软件开发者感到头疼。不仅如此，如果DLL在文件系统中的存储位置发生变化，而DLL在注册表中的信息没有被相应地修改，则DLL中的COM接口将成为不可访问的接口。举一个例子，当我们安装了一个新的应用程序后，如果这个应用程序使用了与其他应用程序共享的某个构件的新版本，便可能导致新安装的应用程序不能正常工作，还会破坏使用这个共享构件的所有其

他应用程序。

COM构件在软件开发中还存在其他一些问题，包括内存管理困难、拖延开发时间、所提供的典型GUI控件不能完全满足许多开发任务的需要，以及缺乏语言之间的互操作性，如C++和Visual Basic之间的互操作性。

为了解决软件开发中的上述问题和其他问题，Microsoft开始研究后来被称为COM+ 2.0的解决方案。这一解决方案旨在设计实现一个托管环境，在该环境中，执行的代码提供了增强的类型安全性、代码安全性以及丰富的几乎令人难以置信的实用类库和函数库，使得开发者更容易完成开发工作。此解决方案最终演化成现在的.NET框架。.NET框架的1.0版和1.1版此前已相继发布，现在2.0版也已问世。本章的下面几节将依次介绍.NET框架的几个关键特征，包括公共语言运行时、公共类型系统和垃圾收集环境中托管代码的概念。

1.1.2 公共语言运行时

在介绍C#语言的基础知识之前，需要了解一些C#语言（和所有其他.NET语言）的工作机理、这些语言在.NET框架中的位置，以及它们与.NET框架中其他组成部分之间的关系。

.NET框架的一个重要组成部分是公共语言运行时（Common Language Runtime，CLR）。C#与以往的C++语言及其他类似语言不同，C#语言运行于一个托管的（managed）环境。用C#语言编写的代码，在公共语言运行时的上下文中执行。公共语言运行时负责管理内存和安全性，并将C#代码与其他非托管代码隔离，以使应用程序的正常操作不受恶意或设计不良的代码所影响。

图1.1显示了.NET框架的层次结构。

.NET框架的层次结构

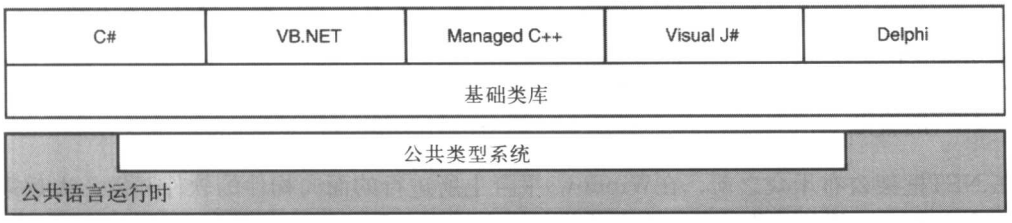

图1.1　.NET框架的层次结构

.NET框架中，最上层是各种.NET语言（图1.1中列出的只是CLR上能够运行的语言中的几种）。在这些语言之下是基础类库(Base Class Library, BCL)，它是在.NET框架中事先编写的各种类和实用程序的集合，它提供了创建应用程序所需的基本服务，例如用于加密、数据访问、文件I/O、创建Web应用和创建Windows应用的程序代码等。读完本书后你会发现，本书的主要任务不是让读者学习C#语言的语法结构，而是让读者掌握BCL中的各种类库。CLR是一种驱动大家在BCL中编写的各种代码和创建的各种应用程序的托管的执行引擎。

Don Box写了一本关于“公共语言运行时”的好书，书中描述了所有你能想到的关于CLR是什么及其如何工作的细节。这本书名为《*Essential.Net Volume I: The Common Language Runtime*》（ISBN 0201734117, Addison-Wesley出版社）。

1.1.3 公共类型系统

用一种语言编写的程序，缺乏与用另一种语言编写的程序进行数据交换的能力，这是许多编程语言存在的一个共同问题。例如，由于不同语言中参数的传递顺序存在差别，在一个Pascal程序中调用C语言编写的方法时，需要格外小心，在C程序中调用Pascal方法时，也是如此。另外，C语言中的字符串是一个以ASCII码字符NULL（在程序中通常用‘\0’表示）为结尾的字符数组。而在Pascal语言中，字符串实际上是一个字符数组，数组中的第一个字节实际上包含了字符串的长度。这个例子所反映的问题只是冰山一角。正如大家所知，不同语言之间的数据通信，很可能是一个噩梦般的任务（历史上确实曾经如此）。

上述问题的一个解决方案是，允许所有的语言共享对数据类型的某种公共表示。公共类型系统（Common Type System）正是提供了一个公共的数据类型集合。例如，如果你引用了一个VB.NET、C#、J#、Delphi (.NET)、托管C++或其他任何一种.NET语言的字符串，公共类型系统能够确保你所引用的字符串对这些不同的语言来说是完全相同的实体。这是因为string类型是在.NET框架本身中定义的数据类型，而不是在语言中定义的数据类型。让数据类型的定义与编程语言分离，便能够创建一个允许开发者采用VB.NET和C#语言混合编程，且不存在通信问题的编程环境。

1.1.4 清除废物：在垃圾收集环境中编码

此前提到，CLR能够为开发者分担许多内存管理工作。如果你使用过C# 1.0或1.1，那么对垃圾收集这一概念将不会陌生。

CLR中的一个构件是垃圾收集器（Garbage Collector, GC）。当你用C#语言声明了一些新的变量（有关C#中的变量声明，将在下一节介绍），这些变量就被纳入垃圾收集器的管理之下。当一个回收周期到来时，垃圾收集器将对变量执行检查，如果某些变量不再被使用，垃圾收集器将销毁这些变量（称为回收）。本书的第16章将进一步介绍有关垃圾收集器的更多内容，该章还将描述如何在编码和设计中有意识地利用垃圾收集器来改善应用程序的执行速度。现在，你只需知道在你和你声明的变量背后，隐藏了一个帮你管理内存和清除废物的垃圾收集器，便足够了。

1.2 C# 2.0中的变量

在解释“变量”这一基本概念时，我最喜欢用“桶”这个词作为变量的类比。变量可以看成是一个桶，你可以在桶中存放数据。一些桶并不在意你在其中放置的数据类型是什么，而另一些桶则对其中能够放入的数据类型有特殊要求。你可以将数据从一个桶转存到另一个桶。遗憾的是，当你考虑到一个桶中存放的仅仅是一个读作“实际数据请参考桶B”的注释时（1.2.3节将会介绍引用类型的概念），将变量作为桶的比喻或许会引起一些混淆。

在C#语言中声明一个变量，可使用下面的语法：

```
type variable_name;
```

声明变量的同时，可以对变量初始化：

```
type variable_name = initialization expression;
```

其中type是一个.NET数据类型。一些.NET中的核心数据类型，在下一节将会介绍。

1.2.1 公共.NET类型

表1.1列出了.NET框架中定义的几个基本数据类型。在后面几章你会看到，这里对.NET中的数据类型的介绍只是刚刚开始。当你开始学习并使用类后，可供使用的类型几乎是无限的。

表1.1　核心.NET数据类型

数据类型	说　明
System.Boolean	提供了存储逻辑数据“true/false”的方式
System.Byte	表示一个单字节数据
System.Char	单字符类型，与其他语言中的字符类型不同，该字符是一个2字节的Unicode字符
System.Decimal	有效位数为28至29位的十进制整数类型，数据范围：$\pm1.0\times10^{-28}\sim\pm7.9\times10^{28}$
System.Double	64位双精度浮点数类型，数据范围：$\pm5.0\times10^{-324}\sim\pm1.7\times10^{308}$
System.Single	32位单精度浮点数类型，数据范围：$\pm1.5\times10^{-45}\sim\pm3.4\times10^{38}$
System.Int32	有符号32位整数类型，数据范围：－2 147 483 648～2 147 483 647
System.Int64	有符号64位整数类型，数据范围：－9 223 372 036 854 775 808～9 223 372 036 854 775 807
System.SByte	有符号8位整数类型
System.Int16	有符号16位整数类型
System.UInt32	无符号32位整数类型
System.UInt64	无符号64位整数类型
System.UInt16	无符号16位整数类型
System.String	任意长度的Unicode字符串类型

如果你对.NET框架或C#语言还不够熟悉，那么一定想知道表1.1中的类型名里出现的“System”是什么含义。在.NET中，类型按照名字空间组织，一个名字空间是一个逻辑容器，它提供了数据类型的名字解析。.NET框架中的核心数据类型都属于“System”名字空间。在学习.NET的其他特征时，本书还会提及更多的名字空间。

1.2.2 类型简写

在声明核心数据类型时，C#提供了用于简化核心数据类型声明的类型简写。每一个类型简写是一个小写字母组成的单词，它是一个核心.NET类型的别名，在程序编译时，这些类型简写代表了相应的核心数据类型。表1.2列举了一些.NET核心数据类型和这些类型的简写。

表1.2　.NET数据类型在C# 2.0中的别名

类型简写	.NET核心数据类型	类型简写	.NET核心数据类型
bool	System.Boolean	long	System.Int64
byte	System.Byte	sbyte	System.SByte
char	System.Char	short	System.Int16
decimal	System.Decimal	uint	System.UInt32
double	System.Double	ulong	System.UInt64
float	System.Single	ushort	System.UInt16
int	System.Int32		