

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

TP312

2364

2007

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。

Authorized translation from the English language edition entitled *Microsoft Visual C# 2005 Unleashed* by Kevin Hoffman, published by Pearson Education, Inc., publishing as Sams, Copyright © 2006 by Sams Publishing.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanic, including photocopying, recording, or by any information storage retrieval system, without permission of Pearson Education, Inc.

Chinese simplified language edition published by China Machine Press.

Copyright © 2007 by China Machine Press.

本书中文简体字版由美国Pearson Education培生教育出版集团授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书法律顾问 北京市展达律师事务所

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、宋森、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

译者序

前言

第一部分 C# 2.0基础

第1章 C# 2.0简介	1
1.1 什么是.NET框架	1
1.1.1 .NET的演化	1
1.1.2 公共语言运行时	2
1.1.3 公共类型系统	3
1.1.4 清除废物：在垃圾收集环境中编码	3
1.2 C# 2.0中的变量	3
1.2.1 公共.NET类型	4
1.2.2 类型简写	4
1.2.3 值类型与引用类型	5
1.3 C#的基本语法	5
1.3.1 程序块	5
1.3.2 经典的“Hello World”例程	5
1.4 用C#能编写什么	7
1.5 小结	7
第2章 表达式和控制结构	8
2.1 分支和条件逻辑	8
2.1.1 布尔表达式介绍	8
2.1.2 简单条件语句	9
2.1.3 高级条件语句	11
2.2 循环和迭代	11
2.2.1 for 循环	12
2.2.2 while循环	12
2.2.3 do循环	13
2.3 小结	13
第3章 字符串和正则表达式	14
3.1 字符串	14
3.1.1 .NET框架中的字符串	14

3.1.2 格式化字符串	14
3.1.3 操纵和比较字符串	17
3.1.4 StringBuilder类介绍	18
3.2 正则表达式	19
3.2.1 输入确认	19
3.2.2 从输入中抽取数据	20
3.3 小结	20
第4章 数组和集合	21
4.1 数组	21
4.1.1 数组的声明和初始化	21
4.1.2 一维数组	22
4.1.3 多维数组	24
4.1.4 缺口型数组	26
4.2 集合	29
4.2.1 数组和集合的比较	29
4.2.2 ArrayList类	29
4.2.3 Hashtable类	31
4.2.4 Queue类	32
4.2.5 Stack类	34
4.2.6 SortedList类	35
4.3 小结	35
第5章 C#面向对象编程	36
5.1 面向对象设计	36
5.1.1 面向对象设计概述	36
5.1.2 类设计	36
5.1.3 接口设计	38
5.2 面向对象编程	38
5.2.1 创建简单的类	38
5.2.2 处理成员可见性	39
5.2.3 对象继承	41
5.2.4 多态简介	43
5.2.5 实现接口	44
5.3 小结	46

第6章 泛型简介	47	9.6 小结	89
6.1 泛型概述	47	第10章 多线程程序设计	90
6.1.1 泛型的好处	47	10.1 线程编程基础	90
6.1.2 类型参数介绍	48	10.2 编写第一个多线程应用程序	92
6.1.3 对类型参数施加约束	48	10.2.1 线程的创建和运行	92
6.2 创建泛型类型	50	10.2.2 线程终止	93
6.2.1 创建泛型类	50	10.2.3 线程挂起	95
6.2.2 创建泛型方法	50	10.2.4 线程休眠	95
6.2.3 创建泛型接口	52	10.2.5 线程合并	95
6.3 使用泛型集合	52	10.3 线程的同步和竞争	96
6.3.1 Dictionary类	52	10.3.1 lock关键字	97
6.3.2 List类	53	10.3.2 互斥体	97
6.3.3 Queue类	53	10.3.3 监控器	99
6.3.4 Stack类	54	10.3.4 Interlocked类	100
6.4 小结	54	10.3.5 ReaderWriterLock类	100
		10.3.6 使用手工或自动重置的事件	102
		10.4 ThreadPool类	104
		10.5 小结	105
		第11章 反射	106
		11.1 反射介绍	106
		11.2 获得方法信息	107
		11.3 获得成员信息	110
		11.4 检查事件	112
		11.5 创建和检查定制的代码属性	114
		11.6 小结	116
		第12章 程序集和应用程序域	117
		12.1 程序集介绍	117
		12.2 揭开程序集的面纱	117
		12.3 创建和使用程序集	119
		12.4 存储和获取程序集中的资源	121
		12.5 本地化和卫星资源程序集	123
		12.6 应用程序域介绍	125
		12.7 用AppDomain类编程	126
		12.8 小结	128
		第13章 COM和Windows互操作性	129
		13.1 C#中的互操作性	129
		13.2 在.NET框架中使用COM对象	129
		13.3 将.NET类包装为COM对象	132
第7章 I/O和持久化	55		
7.1 流	55		
7.2 基本文件I/O	58		
7.2.1 创建文件和在文件中添加数据	58		
7.2.2 从已有文件中读取数据	59		
7.2.3 使用目录和文件系统	60		
7.3 异步文件I/O	62		
7.4 独立存储	63		
7.5 小结	65		
第8章 使用XML	66		
8.1 XML文档的读写	66		
8.2 用Xpath查询XML文档	69		
8.3 用XSLT转换XML文档	71		
8.4 用XSD确认XML文档	73		
8.5 小结	75		
第9章 事件和委托	76		
9.1 委托简介	76		
9.2 匿名方法	78		
9.3 创建多播委托	80		
9.4 事件简介	82		
9.5 基于事件的高级编程	85		

第二部分 .NET框架2.0基础

13.4 访问非托管DLL中的代码	134
13.5 小结	136
第14章 代码访问安全性	137
14.1 代码访问安全性(CAS)介绍	137
14.2 使用和管理安全策略	138
14.3 强制安全	140
14.4 声明安全	144
14.5 小结	146
第15章 加密和数据保护	147
15.1 加密方法简介	147
15.1.1 私钥加密	147
15.1.2 公钥加密	148
15.1.3 散列	149
15.1.4 数字签名	149
15.2 使用私钥加密	149
15.3 使用公钥加密	151
15.4 使用散列和数字签名	153
15.5 使用数据保护API (DPAPI)	155
15.6 小结	157
第16章 优化.NET 2.0代码	158
16.1 理解装箱和拆箱	158
16.2 恰当地使用字符串操纵	160
16.3 构造高效的循环	160
16.4 加快应用程序启动速度	161
16.5 使用Performance Wizard进行 代码分析	162
16.6 小结	165

第三部分 .NET 2.0中的数据访问

第17章 ADO.NET基础	167
17.1 ADO.NET介绍	167
17.2 建立连接	167
17.2.1 构造连接字符串	167
17.2.2 使用DbConnection类	168
17.3 与数据源通信	170
17.3.1 执行命令	170
17.3.2 使用DataReader	173
17.3.3 使用模式发现	175
17.4 处理数据	176
17.4.1 DataSet介绍	177

17.4.2 使用DataAdapter	178
17.5 小结	181
第18章 ADO.NET高级技术	182
18.1 改进的DataTable	182
18.1.1 使用XML加载和保存DataTable	182
18.1.2 使用新的DataTableReader类	184
18.2 异步访问数据	185
18.3 批量更新数据	186
18.4 使用新的System.Transactions名字空间	189
18.4.1 显式使用事务	189
18.4.2 隐式使用事务	190
18.5 小结	193
第19章 ADO.NET数据提供者	194
19.1 ADO.NET中的数据提供者简介	194
19.2 使用提供者工厂	194
19.2.1 获得已安装的提供者工厂的列表	195
19.2.2 使用提供者工厂建立连接	195
19.3 使用连接字符串	197
19.4 数据源枚举	198
19.5 获得更多的提供者信息	199
19.5.1 使用RetrieveStatistics方法	199
19.5.2 从数据提供者获取模式信息	201
19.6 创建定制的ADO.NET数据提供者	201
19.7 小结	202
第20章 强类型数据集	203
20.1 有类型数据集简介	203
20.1.1 使用XSD模式创建有类型数据集	203
20.1.2 使用Designer创建有类型数据集	204
20.1.3 用有类型数据集编程	205
20.2 连接有类型数据集到活动数据	206
20.2.1 使用DataAdapter对象手工为 DataSet对象填充数据	206
20.2.2 使用TableAdapter对象为有类型 DataSet对象填充数据	207
20.2.3 为有类型DataSet添加更多的查询	210
20.3 修饰有类型数据集	211
20.4 使用部分类扩展有类型数据集	213
20.5 小结	214

第21章 SQL Server 2005编程	215	23.5 小结	266
21.1 SQL Server 2005 CLR宿主简介	215	第24章 使用主控页	267
21.2 用C#编写存储过程	215	24.1 主控页出现之前	267
21.3 用C#编写用户自定义函数	217	24.1.1 对GUI一致性的需求	267
21.4 用C#创建用户自定义类型	219	24.1.2 在ASP.NET 1.1中创建一致的GUI	267
21.5 使用新的服务器端SQL库	223	24.2 主控页介绍	268
21.6 使用多活动记录集	226	24.2.1 主控页和内容页	268
21.7 小结	228	24.2.2 创建第一个主控页	269
第4部分 开发ASP.NET 2.0 Web应用程序		24.2.3 创建第一个内容页	270
第22章 ASP.NET 2.0和Web Forms介绍	229	24.2.4 使用默认的主控页	271
22.1 ASP.NET 2.0简介	229	24.2.5 了解主控页的实质	271
22.1.1 页面和控件层次	230	24.3 高级主控页技术	272
22.1.2 ASP.NET 2.0编译器介绍	230	24.3.1 嵌套的主控页	272
22.1.3 安全性	230	24.3.2 Master属性	273
22.1.4 状态管理	230	24.3.3 强类型主控页	273
22.1.5 Web配置系统	231	24.3.4 处理相对路径	274
22.2 理解ASP.NET页的生命周期	231	24.4 小结	274
22.2.1 ASP.NET页的处理阶段	231	第25章 ASP.NET 2.0个性化和用户定制	275
22.2.2 ASP.NET页生命周期中的事件	232	25.1 利用主题和外观裁剪用户界面	275
22.3 ASP.NET中的控件	235	25.2 ASP.NET用户大纲	278
22.4 创建和调试ASP.NET应用程序	237	25.2.1 配置应用服务	278
22.4.1 ASP.NET应用程序的构造和部署	237	25.2.2 配置大纲提供者	279
22.4.2 调试ASP.NET应用程序	238	25.2.3 使用ASP.NET大纲	280
22.5 处理事件和回递	239	25.3 利用主题和大纲进行用户定制	284
22.6 使用客户端回调构建交互式 动态页面	241	25.4 小结	286
22.7 小结	248	第26章 Web Part介绍	287
第23章 ASP.NET 2.0中的状态管理	250	26.1 Web Part基础	287
23.1 应用程序状态	250	26.2 使用个性化提供者	289
23.2 会话状态	252	26.3 创建第一个Web Part页	290
23.2.1 默认的进程内状态提供者	253	26.4 创建Web Part	297
23.2.2 ASP.NET状态服务器提供者	256	26.5 创建可连接的Web Part	298
23.2.3 SQL Server会话状态提供者	259	26.6 小结	302
23.2.4 处理会话状态事件	261	第27章 构建表现力丰富的、数据驱动的 Web应用	303
23.3 视图状态	263	27.1 ASP.NET数据绑定介绍	303
23.4 Web Farm中的状态管理	264	27.1.1 数据源模型	303
23.4.1 在Web Farm中使用应用程序状态	264	27.1.2 创建数据源	303
23.4.2 在Web Farm中使用会话状态	265	27.1.3 数据绑定控件的层次结构	305
23.4.3 在Web Farm中使用视图状态	265	27.2 使用数据绑定控件	306

27.2.1 使用GridView控件	307
27.2.2 使用DetailsView控件	307
27.2.3 使用FormView控件	307
27.2.4 使用TreeView控件	308
27.3 高级数据绑定技术	309
27.4 小结	310
第28章 ASP.NET应用程序安全保护	311
28.1 通过身份认证保护应用程序安全	311
28.1.1 Windows身份认证	311
28.1.2 通行证身份认证	312
28.1.3 窗体认证	313
28.1.4 用户成员管理	314
28.2 通过用户授权保护应用程序安全	317
28.3 ASP.NET安全保护控件	319
28.3.1 Login控件	319
28.3.2 LoginName控件	320
28.3.3 LoginStatus控件	320
28.3.4 LoginView控件	320
28.3.5 PasswordRecovery控件	321
28.3.6 ChangePassword控件	322
28.3.7 CreateUserWizard控件	322
28.4 高级ASP.NET安全保护	323
28.5 小结	325
第29章 创建定制的ASP.NET提供者	327
29.1 用户成员提供者	327
29.1.1 MembershipProvider基类简介	327
29.1.2 实现一个用户成员模式	328
29.1.3 创建一个定制的用户成员提供者	329
29.1.4 配置和安装用户成员提供者	337
29.2 角色提供者	338
29.2.1 RoleProvider基类简介	338
29.2.2 实现一个角色模式	339
29.2.3 创建一个定制的角色提供者	339
29.2.4 配置和安装角色提供者	344
29.3 大纲提供者	345
29.3.1 ProfileProvider基类简介	345
29.3.2 实现一个大纲模式	345
29.3.3 创建一个定制的大纲提供者	346
29.3.4 配置和安装大纲提供者	351

29.4 其他的提供者	351
29.4.1 会话状态提供者简介	351
29.4.2 站点地图提供者简介	352
29.5 小结	353
第30章 开发ASP.NET控件	354
30.1 创建用户控件	354
30.2 创建服务器控件	357
30.3 使用服务器控件进行状态管理	360
30.4 小结	364
第31章 ASP.NET的管理和监控	365
31.1 新的健康监控系统简介	365
31.1.1 使用健康监控系统	366
31.1.2 创建自定义事件	369
31.1.3 创建自定义事件提供者	372
31.2 使用ASP.NET性能计数器	373
31.3 小结	375

第五部分 Web服务

第32章 提供Web服务	377
32.1 Web服务介绍	377
32.2 创建一个简单的“Hello World” Web服务	378
32.3 创建事务型Web服务	385
32.4 Web服务中的状态管理	386
32.5 小结	387
第33章 Web服务高级编程	388
33.1 面向服务的架构设计	388
33.1.1 松耦合和依赖	388
33.1.2 日常生活中的SOA	389
33.2 Web服务发现	390
33.3 定制的SOAP头	393
33.4 编写安全的Web服务	395
33.5 Windows Forms与Web服务之间的 数据绑定	397
33.6 小结	398

第六部分 开发Windows Forms 2.0 应用程序

第34章 Windows Forms 2.0介绍	399
34.1 Windows Forms基础	399

34.2 创建Windows Forms应用程序	401	35.2.4 SplitContainer控件	413
34.3 使用Windows Forms设计器	402	35.2.5 TabControl控件	413
34.3.1 文档大纲窗口	402	35.2.6 TableLayoutPanel控件	414
34.3.2 用基准线对齐控件	403	35.3 菜单和工具条	414
34.3.3 创建可调整大小的窗体	404	35.3.1 ContextMenuStrip控件	414
34.4 设计优秀的用户界面要考虑的因素	405	35.3.2 MenuStrip控件	414
34.4.1 颜色设计	405	35.3.3 StatusStrip控件	414
34.4.2 尺寸设计	405	35.3.4 ToolStrip控件	415
34.4.3 复杂性设计	405	35.3.5 ToolStripContainer控件	415
34.4.4 点击数设计	405	35.4 与数据有关的控件	415
34.4.5 感官直觉设计	406	35.4.1 DataSet控件	415
34.5 小结	406	35.4.2 DataGridView控件	415
第35章 Windows Forms 控件库	407	35.4.3 BindingSource控件	415
35.1 公共控件工具箱	407	35.4.4 BindingNavigator控件	416
35.1.1 Button控件	407	35.4.5 ReportViewer控件	416
35.1.2 CheckBox控件	407	35.5 构件工具箱组	416
35.1.3 CheckedListBox控件	407	35.5.1 BackgroundWorker构件	416
35.1.4 ComboBox控件	408	35.5.2 DirectoryEntry构件	416
35.1.5 DateTimePicker控件	408	35.5.3 DirectorySearcher构件	416
35.1.6 Label控件	408	35.5.4 ErrorProvider构件	416
35.1.7 LinkLabel控件	408	35.5.5 EventLog构件	417
35.1.8 ListBox控件	409	35.5.6 FileSystemWatcher构件	417
35.1.9 ListView控件	409	35.5.7 HelpProvider构件	417
35.1.10 MaskedTextBox控件	409	35.5.8 ImageList构件	418
35.1.11 MonthCalendar控件	410	35.5.9 MessageQueue构件	418
35.1.12 NotifyIcon控件	410	35.5.10 PerformanceCounter构件	418
35.1.13 NumericUpDown控件	410	35.5.11 Process构件	418
35.1.14 PictureBox控件	411	35.5.12 SerialPort构件	418
35.1.15 ProgressBar控件	411	35.5.13 ServiceController构件	418
35.1.16 RadioButton控件	411	35.5.14 Timer构件	418
35.1.17 TextBox控件	411	35.6 打印构件和打印控件	418
35.1.18 RichTextBox控件	411	35.6.1 PageSetupDialog构件	419
35.1.19 ToolTip控件	411	35.6.2 PrintDialog构件	419
35.1.20 TreeView控件	412	35.6.3 PrintDocument构件	419
35.1.21 WebBrowser控件	412	35.6.4 PrintPreviewControl构件	419
35.2 容器	412	35.6.5 PrintPreviewDialog构件	419
35.2.1 FlowLayoutPanel控件	412	35.7 对话框构件	419
35.2.2 GroupBox控件	413	35.7.1 ColorDialog构件	419
35.2.3 Panel控件	413	35.7.2 FolderBrowserDialog构件	419

35.7.3	FontDialog构件	419	38.4	用户授权和身份认证	456
35.7.4	OpenFileDialog构件	420	38.5	通过多线程使用Web服务	459
35.7.5	SaveFileDialog构件	420	38.6	使用BackgroundWorker控件	460
35.8	小结	420	38.7	小结	462
第36章	高级用户界面编程	421	第39章	使用ClickOnce部署应用程序	463
36.1	GDI+介绍	421	39.1	ClickOnce介绍	463
36.1.1	获得一个Graphics对象	421	39.2	发布ClickOnce应用程序	464
36.1.2	创建“Hello GDI+”示例	422	39.2.1	通过Web或网络共享部署	465
36.1.3	绘制和填充形体	423	39.2.2	部署到CD	465
36.1.4	使用渐变色刷	425	39.2.3	直接从Web或共享节点 发布应用程序	465
36.2	创建有形状的窗体和控件	427	39.2.4	部署一个ClickOnce应用程序	465
36.3	利用可视化继承	428	39.3	更新ClickOnce应用程序	468
36.4	用户界面的全球化	429	39.4	使用System.Deployment.Application 名字空间编程	470
36.5	小结	431	39.5	小结	474
第37章	Windows Forms 2.0中的数据绑定	432	第40章	使用企业服务	475
37.1	有类型数据集的数据绑定	432	40.1	注册服务构件	475
37.1.1	使用Data Source窗口	432	40.1.1	手工注册	475
37.1.2	在窗体中增加一个DataSet	433	40.1.2	自动注册	475
37.1.3	有类型DataSet绑定的示例	434	40.2	即时激活和对象池	478
37.2	BindingSource介绍	435	40.3	入队构件	479
37.3	使用BindingNavigator	437	40.4	基于角色的安全管理	481
37.3.1	BindingNavigator基础	437	40.5	事务	481
37.3.2	用户触发的绑定动作	437	40.6	共享属性	482
37.4	使用DataGridView	439	40.7	松耦合事件	485
37.4.1	DataGridView基础	439	40.8	小结	487
37.4.2	在DataGridView中使用一个 ComboBox列	441	第七部分	开发企业级和分布式应用程序	
37.4.3	单元格的高级定制	442	第41章	远程调用	489
37.4.4	DataGridView控件的“解绑定”	444	41.1	远程调用概述	489
37.5	对象的数据绑定	445	41.1.1	MarshalByRefObject类介绍	490
37.5.1	对象的数据绑定基础	445	41.1.2	一次调用对象与单体对象	490
37.5.2	使用IEditableObject和 INotifyProperty Changed接口	446	41.2	使用远程信道	492
37.6	父表-细节表绑定	450	41.2.1	IPC信道	492
37.7	小结	450	41.2.2	TCP信道	495
第38章	开发智能客户端	451	41.3	使用生命周期租期	498
38.1	实际使用Web服务	451	41.4	用泛型实现远程调用	499
38.2	使用新的应用程序设置系统	452	41.5	小结	502
38.3	支持离线和在线操作	454			

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。

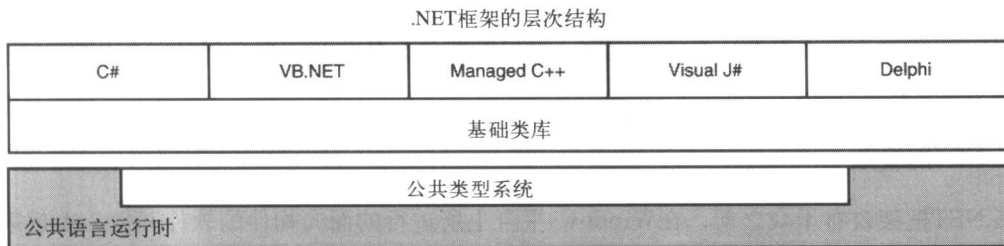


图1.1 .NET框架的层次结构

.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是一种驱向大家在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： $\pm 1.0 \times 10^{-28} \sim \pm 7.9 \times 10^{28}$
System.Double	64位双精度浮点数类型，数据范围： $\pm 5.0 \times 10^{-324} \sim \pm 1.7 \times 10^{308}$
System.Single	32位单精度浮点数类型，数据范围： $\pm 1.5 \times 10^{-45} \sim \pm 3.4 \times 10^{38}$
System.Int32	有符号32位整数类型，数据范围： $-2\,147\,483\,648 \sim 2\,147\,483\,647$
System.Int64	有符号64位整数类型，数据范围： $-9\,223\,372\,036\,854\,775\,808 \sim 9\,223\,372\,036\,854\,775\,807$
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词解析。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		