HZ BOOKS

碁峰
www.gotop.com.tw

附贈 Microsoft Visual C# 2005 Express Edition中文版

微软公司资深顾问及讲师章立民作品

Visual C# 2005

文件IO与数据存取秘诀

章立民研究室 著

机械工业出版社
China Machine Press

Visual C# 2005

文件IO与数据存取秘诀

章立民研究室 著

机械工业出版社
China Machine Press

本书综合讲解 Visual C# 2005 文件 IO 与数据存取的相关知识，内容全面，结构合理，论述清晰，对 Visual C# 2005 文件 IO 与数据存取技术及其实际应用都有独到见解，是一本专业性较强的计算机书籍。全书共分 14 章，包括磁盘、文件夹与文件的处理技巧、Windows Form 数据绑定、如何连接数据源、数据集模型的数据存取技巧、数据命令模型的数据存取技巧、大型对象的存取技巧、如何使用多活动结果集、执行非同步的数据存取技巧、如何使用 SqlBulkCopy 对象来执行大量复制操作、DataSet 与 XML 的数据访问技巧以及 DataGridView 控件的重要开发技巧等内容。书中包含有大量范例，可作为专业编程人员的参考书籍，也适合于对 Visual C# 有一定了解且想深入研究的读者。

本书版权登记号：图字：01-2007-0473

图书在版编目(CIP)数据

Visual C# 2005 文件 IO 与数据存取秘诀/章立民研究室著．-北京：机械工业出版社，2007.3

ISBN 978-7-111-19972-4

Ⅰ.V…　Ⅱ.章…　Ⅲ.C 语言-程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2006)第 137881 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：李南丰

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2007 年 3 月第 1 版第 1 次印刷

186mm×240mm·38.5 印张

定价：79.00 元(附赠DVD)

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

推荐序

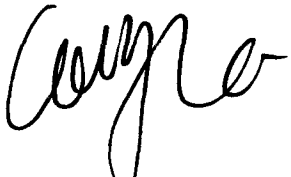
Preface

各位亲爱的读者朋友，大家好。我是微软最有价值专家暨技术社区总监——柯淑芬。非常高兴能够在这里与大家见面，并感谢大家购买本书。

本书的作者章立民先生是一位知名的技术图书作家，他与中国台湾微软公司的合作时间长达14年以上，并拥有17年以上的写作经验。他对微软开发工具与数据库管理系统等技术都有非常深入的研究。近两、三年来，他积极参与微软技术社区的相关活动，不吝将其研究的心得与大家分享。正因为他的这股热忱和奉献精神，使其当选为微软最有价值专家(MVP)。

微软最有价值专家是名副其实的专业精英，他们不仅精通某项微软产品及技术，更重要的是，他们非常乐于助人且不求回报。到目前为止，全球共有近三千名来自八十多个国家的人士当选为MVP。特别值得一提的是，中国台湾地区目前有将近100位人士获得MVP这项殊荣。时至今日，微软最有价值专家对使用微软公司产品的用户和技术社区已产生重大影响，从操作的细节，乃至于全方位的战略性规划，MVP都提供完善的协助和技术支持。

微软技术社区是一个完全免费的技术咨询园地，有关微软产品任何使用上的问题，都可以在此询问，并能与MVP进行充分的讨论和交流。而在解惑与提升技术的同时，您也许就是下一位MVP的最佳候选人。期待在微软技术社区与您见面。我们下次再见。



微软技术社区暨最有价值专家
亚太及大中华地区
区域总监
柯淑芬　Cally K

作者序

Prefate

——对自己的充分自信，对市场的完全尊重

综览人类文明的演进，大抵都建立在“今日会比昨日好、明日要比今日强”的基础上。身处于信息技术的洪流之中，每天像陀螺一样旋转，不停地追逐新技术，那种漫步在云端的不扎实感，我想是每一位IT人的苦与痛。记得我之前在台北与北京做主题为“程序人生”的演讲时，孟岩帮我写了一段宣传词，内容是这样的：“卡布奇诺有泡沫也有咖啡，如何把握住技术的本质，走稳技术人生路？也许善品咖啡的章立民先生，能对您有所启发。”

一直以来，我从不吝啬与他人分享自己的技术心得和人生经验，最近似乎又有点冲动，想找机会谈谈程序人生之类的主题。毕竟技术性的研讨会随时随处都有，但是却少有从心灵与人生规划层面来探讨的主题。其实，我也没啥成就，不过毕竟在这一行单打独斗17年，也算是小有心得啦。尤其在历经这些年来的人生沉浮之后，似乎更能看清许多人与事的本质。低潮期让我未来的处事态度多了理性、少了感性、更加务实，而当回复到上升的轨道时，我则是更加的谨慎与低调。

我热爱技术，但是我更务实，所以我常说，不要跟技术谈恋爱。技术本身是单纯的，面对技术的态度以及如何去应用技术，才是成败的关键。如果要在这个产业长久发展，你必须让自己与众不同，而不同之处不仅仅是技术。若只是一味盲目乱冲，而不适时地让自己缓下来思考，很容易被洪流吞噬。或许该找一天，左手拿着拿咖啡，右手握着麦克风，与大家谈谈面对技术洪流的生存之道，也让您思考技术的本质以及学习的方法。我们下次再见。

章立民

立民讲堂－http://liminzhang.cnblogs.com/

笔于台北中和

2006年11月3日

作者简介：

章立民

微软公司资深顾问讲师。从1992年开始于中国台湾微软主讲研讨会。

中国台湾微软最有价值专家MVP，连续四度当选MVP。

资深计算机图书作家，潜心技术创作17年，拥有60本以上著作。

专长：

关系型数据库管理系统
Visual Studio 2005开发工具
SQL Server、Access 2003等

著作：

迄今为止，章立民先生已有60余部计算机著作问世，内容涵盖SQL Sever 2000、Visual Basic.Net、ASP.NET、Crystal Report for Visual Studio.NET、Access 2002-2003、Visual FoxPro、Word VBA、Windows等。

章立民老师博客：

www.cnblogs.com/liminzhang

HZ BOOKS
专业成就人生
立体服务大众

目 录

Contents

推荐序
作者序

第 1 章 磁盘、文件夹与文件的处理技巧 …… 1

条款 1 综述 …… 1
条款 2 路径指定方式 …… 1
条款 3 使用 DriveInfo 类取得计算机的磁盘信息 …… 2
条款 4 使用 My. Computer. FileSystem 对象取得计算机的磁盘信息 …… 10
条款 5 如何列举目录 …… 11
条款 6 使用 My. Computer. FileSystem 对象来取得目录信息 …… 15
条款 7 .NET Framework 2.0 对 GetDirectories 方法的强化 …… 19
条款 8 善用 My. Computer. SpecialDirectories 对象 …… 21
条款 9 善用 Environment. GetFolderPath 方法 …… 23
条款 10 如何取得文件的相关信息 …… 25
条款 11 使用 My. Computer. FileSystem 对象来取得文件信息 …… 31
条款 12 .NET Framework 2.0 对 GetFiles 方法的强化 …… 33
条款 13 如何建立目录 …… 35
条款 14 如何移除一个目录 …… 36
条款 15 如何移动一个目录 …… 39
条款 16 如何更改目录的名称 …… 40
条款 17 如何复制文件 …… 40
条款 18 如何更改文件的名称 …… 42
条款 19 如何删除文件 …… 43
条款 20 如何使一个文件成为只读的或是可读取及写入的 …… 45
条款 21 如何从文本文件读取数据 …… 46
条款 22 使用很方便的 ReadAllText 与 ReadAllLines 方法来读取文本文件 …… 51
条款 23 如何解析文本文件 …… 53
条款 24 如何解析含有多种格式的文本文件 …… 58
条款 25 如何将文本数据写入文本文件 …… 60
条款 26 使用很方便的 AppendAllText 与 My. Computer. FileSystem. WriteAllText 方法来写入文本文件 …… 62
条款 27 如何建立一个文本文件 …… 63
条款 28 使用便利的 WriteAllText 与 WriteAllLines 方法来建立一个文本文件 …… 64
条款 29 如何读取一个二进制文件 …… 65
条款 30 使用很方便的 ReadAllBytes 方法来读取二进制文件 …… 66
条款 31 如何将数据写入一个二进制文件 …… 67
条款 32 使用很方便的 WriteAllBytes 方法来写入一个二进制文件 …… 68
条款 33 如何比较两个文件的内容是否完全相同 …… 69
条款 34 如何使用 FileSystemWatcher 组件来监视文件系统事件 …… 71
条款 35 如何建立与使用临时文件 …… 73

第 2 章　探讨 IO 的进阶议题 …… *76*

条款 36　如何压缩与解压缩文件 …… *76*
条款 37　模拟建立一个简易版的压缩软件 …… *82*
条款 38　帐户专用的文件加密与解密 …… *99*
条款 39　如何变更文件与目录的访问控制列表(ACL) …… *102*
条款 40　如何访问串行端口 …… *109*

第 3 章　探讨 Windows Forms 数据绑定 … *116*

条款 41　什么是数据绑定 …… *116*
条款 42　在列表上使用简单的数据绑定 … *119*
条款 43　类型转换 …… *120*
条款 44　格式化 …… *121*
条款 45　错误处理 …… *122*
条款 46　同步化注意事项 …… *126*
条款 47　数据变更的反应时机 …… *127*
条款 48　属性变更通知 …… *129*
条款 49　列表变更通知 …… *134*
条款 50　当使用数据绑定时为何无法移出控件与关闭窗体 …… *140*
条款 51　为什么 DataSourceUpdateMode.OnPropertyChanged 对 ADO.NET 数据源没有作用 …… *144*
条款 52　如何使用简单绑定来显示多个数据源属性 …… *148*
条款 53　探索 ComboBox 与 ListBox 控件的数据绑定 …… *151*
条款 54　如何将一个 Null 或 DBNull 项目新增至所绑定的 ComboBox 控件 …… *167*
条款 55　如何让 ComboBox 控件显示出多个数据源属性 …… *169*
条款 56　为什么我的 ComboBox 控件在数据绑定时加载速度很慢 …… *170*
条款 57　如何清除已绑定的 ListBox、ComboBox 或 CheckedListBox 控件的项目 …… *170*
条款 58　探索 BindingSource 组件 …… *171*
条款 59　探索 BindingNavigator 控件 …… *206*
条款 60　如何实现 IBindableComponent 界面 …… *216*

第 4 章　如何连接数据源 …… *219*

条款 61　如何构建连接字符串 …… *219*
条款 62　如何根据用户输入动态建立正确的连接字符串 …… *226*
条款 63　如何连接至 Access 数据库 …… *227*
条款 64　如何连接至 Excel 工作簿 …… *228*
条款 65　如何使用文件名称路径来连接 SQL Server 数据库 …… *229*
条款 66　如何将 SQL Server 数据库文件附加至 SQL Server 实例 …… *232*

第 5 章　探讨数据集模型的数据存取技巧 …… *234*

条款 67　我该使用哪一种数据存取模型 … *234*
条款 68　了解 DataSet 对象 …… *237*
条款 69　数据集的实现流程 …… *240*
条款 70　如何建立数据集 …… *241*
条款 71　如何建立一个独立存在的数据表对象 …… *241*
条款 72　善用 SqlTypes …… *245*
条款 73　如何为数据表建立自动编号字段 … *248*
条款 74　如何为数据表建立表达式字段 … *249*
条款 75　如何定义数据表的主码 …… *254*
条款 76　如何套用 UniqueConstraint 约束 … *256*
条款 77　使用 XML 架构定义数据表的架构 …… *259*
条款 78　如何新增数据行 …… *259*
条款 79　如何编辑数据行 …… *261*
条款 80　如何删除数据行 …… *262*
条款 81　探讨数据行的状态与版本 …… *264*
条款 82　完善的数据异动操作 …… *266*
条款 83　如何将后端数据加载到一个已存的数据表对象中 …… *275*
条款 84　如何将 XML 数据加载到一个既存的数据表对象中 …… *276*
条款 85　如何于数据集当中建立数据表 … *278*
条款 86　自行建立 DataTable 对象并新增至数据集 …… *278*

条款 87 通过数据适配器在数据集当中建立数据表…… 280
条款 88 使用 DataSet 的 Load 方法在数据集当中建立数据表…… 288
条款 89 关于 SQL Server 的 decimal 类型…… 291
条款 90 使用 DataTableReader 快速列举数据表的内容…… 292
条款 91 将现有的约束加入 DataSet 中 … 293
条款 92 数据表与字段的前后端对应关系 … 296
条款 93 建立数据表之间的关系…… 301
条款 94 建立外码约束…… 306
条款 95 如何使用外码约束…… 310
条款 96 取得与设置数据适配器命令的参数…… 311
条款 97 善用数据适配器全新的批处理更新功能…… 333

第 6 章 探讨数据命令模型的数据存取技巧 …… 339

条款 98 如何枚举 SQL Server 实例 …… 339
条款 99 如何枚举 SQL Server 的数据库、数据表与视图表…… 340
条款 100 如何使用数据命令执行 Transact-SQL 语句 …… 343
条款 101 如何使用数据命令来执行存储过程 …… 345
条款 102 如何执行会返回单一值的数据命令 …… 347
条款 103 如何执行会返回结果集的数据命令 …… 348
条款 104 如何执行会返回多结果集的数据命令 …… 352
条款 105 使用数据命令执行数据库操作与修改数据 …… 354
条款 106 如何取得与设置数据命令的 Transact-SQL 语句的参数 …… 361
条款 107 如何取得与设置数据命令的存储过程的参数和返回值 …… 368

第 7 章 探讨大型对象的存取技巧 …… 381

条款 108 什么是大型对象 …… 381
条款 109 以数据流形式来读取数据库中的 LOB …… 382
条款 110 以一整行加载方式读取数据库中的 LOB …… 388
条款 111 如何将 LOB 写入数据库…… 394
条款 112 如何将字段内的二进制图像数据显示在 PictureBox 控件中 …… 399
条款 113 如何将 PictureBox 控件中的二进制图像数据写回字段 …… 401
条款 114 结合使用 SQL Server 2005 的 UPDATE. WRITE 语句 …… 403

第 8 章 如何使用多活动结果集 …… 408

条款 115 什么是多活动结果集…… 408
条款 116 如何启用与禁用多活动结果集 …… 413

第 9 章 如何执行异步的数据存取操作 … 419

条款 117 什么是异步的数据存取操作 … 419
条款 118 使用轮询来实现异步数据存取 …… 419
条款 119 使用回调来实现异步数据存取——最佳做法 …… 421
条款 120 使用 WaitHandle 实现异步数据存取 …… 425

第 10 章 如何使用 SqlBulkCopy 对象执行大量复制操作…… 428

条款 121 关于大量复制操作 …… 428
条款 122 如何执行单次的大量复制操作 …… 428
条款 123 如何执行多次大量复制操作 … 436
条款 124 如何在事务中执行大量复制操作 …… 438

第 11 章 如何存取 SQL Server 2005 的 xml 数据类型数据 …… 443

条款 125 如何提取数据读取器中的 xml 数据类型字段值 …… 443
条款 126 以 XML 作为数据命令的参数 … 444

第 12 章　探讨 DataSet 与 XML 的数据访问技巧 …… 446

条款 127　前言 …… 446
条款 128　从 DiffGram 谈起 …… 446
条款 129　将 XML 的内容载入 DataSet 中 …… 451
条款 130　合并来自 XML 的数据 …… 461
条款 131　将 DataSet 的内容写成 XML 数据 …… 467
条款 132　嵌套 DataRelation …… 475
条款 133　将字段对应至 XML 元素、属性与文字 …… 481
条款 134　将 DataSet 的内容写成 XML 时如何格式化字段数据 …… 483
条款 135　将 XML 的架构加载至 DataSet …… 485
条款 136　将 DataSet 架构信息写成 XML 架构 …… 488
条款 137　使用 XmlDataDocument 来同步 DataSet …… 490
条款 138　根据 XML 架构产生 DataSet 关系型结构 …… 500
条款 139　将 XML 架构约束对应至 DataSet 约束 …… 503
条款 140　根据 XML 架构产生 DataSet 关系 …… 508
条款 141　了解约束和关联性间的交互关系 …… 515
条款 142　根据 XML 推断 DataSet 关系型结构 …… 519

第 13 章　探讨 DataGridView 控件的重要开发技巧 …… 524

条款 143　如何自选数据列类型 …… 524
条款 144　综述 DataGridView 控件 …… 540
条款 145　如何使用数据列的上下文菜单 …… 553
条款 146　如何使用 DateTimePicker 控件编辑单元格的日期数据 …… 559
条款 147　如何使一个单元格不能被编辑 …… 562
条款 148　如何禁用一个单元格 …… 562
条款 149　如何防止用户移入一个单元格 …… 564
条款 150　如何让单元格中的文本随着宽度换行 …… 564
条款 151　如何隐藏一个数据列 …… 565
条款 152　当用户尝试删除数据行时如何显示确认对话框 …… 565
条款 153　如何建立主从式窗体 …… 565
条款 154　如何在 DataGridView 控件中以跨数据列方式显示数据 … 566
条款 155　如何将 DataGridView 控件单独选取成一个位图文件 … 571

第 14 章　探讨综合性的开发议题 …… 576

条款 156　如何自动根据文件的扩展名或类型来启动相关联的应用程序并加载文件 …… 576
条款 157　如何关闭其他应用程序或其他应用程序的特定实例 … 577
条款 158　如何得知所启动的进程已经结束执行 …… 579
条款 159　如何等待所启动的进程执行完毕 …… 579
条款 160　如何传送按键给其他应用程序 …… 581
条款 161　如何以程控方式启动默认的浏览器 …… 582
条款 162　如何使用全新的 System. Net. Mail. MailMessage 类编写邮件传送程序 …… 584
条款 163　如何产生与比较哈希值 …… 591
条款 164　如何加密与解密文本文件 … 593
条款 165　善用全新的 Background Worker 组件设计异步界面 … 598

附录 A　范例安装与使用说明 …… 605

第1章

磁盘、文件夹与文件的处理技巧

条款1 综述

要在 Visual C# 2005 中进行磁盘、文件夹与文件的处理操作，可以使用下列三种方式：

- 使用 FileSystemObject。
- 使用 .NET Framework 类。
- 使用 Microsoft. VisualBasic. Devices 命名空间的 My. Computer. FileSytem 对象。

基于功能完整性与未来性的考虑，我们建议大家应该学习如何使用 .NET Framework 类来完成文件的输入输出操作。基本上，用来进行磁盘、文件夹与文件处理操作的相关类绝大多数位于 System. IO 命名空间中，为了避免在使用这些类时编写冗长的代码，建议大家在窗体或类的开头处使用 using 语句导入 System. IO 命名空间：

```
using System. IO;
```

我常说，在 Visual C# 2005 中要进行文件系统的处理操作将是前所未有的简单，这是因为 Microsoft. VisualBasic. Devices 命名空间中的 My. Computer. FileSystem 对象提供了非常直观、无界限、易于发现且易于使用的方式来执行常用的文件系统操作，诸如复制、删除、移动、重命名文件与文件夹，以及读取和写入文件，全都可以通过 My. Computer. FileSystem 对象来轻易完成。

在本章中，我们将详细讨论如何使用 System. IO 命名空间中的类与 My. Computer. FileSystem 对象来完成磁盘、目录、文件、压缩等各项处理操作。

条款2 路径指定方式

在进行磁盘、文件夹与文件的处理操作时，常常需要指定路径。相关类的许多方法(例如 Directory. Exists)也都会使用路径作为其参数，因此如何正确指定路径便成为了一项非常重要的课题。

基本上，路径可以引用文件或是只引用目录。您所指定的路径也可以是一个相对路径或由一个服务器名称与共享名所构成的通用命名规则(Universal

Naming Convention，UNC)路径，如下所示：

- C:\MyDir
- MyDir\MySubDir
- \\MyServer\MyShare

使用通用命名规则路径当作参数时必须特别注意一件事，那就是 Visual C# 无法识别分隔用的反斜杠符号“\”，因此您可以采用下列两种语法之一，将通用命名惯例路径修改为 Visual C# 允许的参数值：

- 第一种语法是使用两个反斜杠符号“\\”来取代单一个反斜杠符号“\”：

```
"C:\\MyDir"
```

- 第二种语法是在包含一对双引号的路径之前加上 @ 符号：

```
@"C:\MyDir"
```

条款 3 使用 DriveInfo 类取得计算机的磁盘信息

在 Visual Studio 2003 与 .NET Framework 1.1 中，要取得您计算机上所有的逻辑磁盘驱动器，可以通过调用 Directory.GetLogicalDrives 方法或 Environment.GetLogicalDrives 方法来返回一个包含计算机上所有逻辑磁盘名称的字符串数组，这些逻辑磁盘的名称会采用 "<drive letter>:\" 的格式。遗憾的是，.NET Framework 1.1 竟然没有提供任何方式来取得磁盘名称、磁盘类型与剩余空间等信息。因此在过去，我们必须寻求一些替代方案，包括调用 Win32 API 函数或在 Windows Scripting Host(WSH)组件中，通过接口的编组处理来使用 FileSystemObject。

不过这些困扰在 .NET Framework 2.0 中已经完全得到解决，现在只要使用 .NET Framework 2.0 所新提供的 DriveInfo 类，我们就能够轻易地取得磁盘的各项信息。接下来，我们就通过实际的程序范例来说明如何使用 DriveInfo 类。

程序范例1

图 1-1 所示是程序范例 CH1 _ DemoForm001.cs 的执行画面，它示范如何使用 DriveInfo 类来取得计算机上所安装的磁盘驱动器的相关信息。

使用 DriveInfo 类的首要操作，就是调用其 GetDrives 方法取得目前系统中所有逻辑磁盘驱动器的 DriveInfo 类型数组。通过循环处理此 DriveInfo 类型数组中代表每一个逻辑磁盘驱动器的 DriveInfo 对象并读取其属性，即可取得该磁盘的相关信息。

值得一提的是，本范例是取得所有“已就绪”的逻辑磁盘的相关信息。什么叫“已就绪”呢？举例来说，如果软驱或光驱并未插入软盘或光盘而未准备好进行读取或写入操作的话，就是“未就绪”或是“未准备好”。在这种情况下使用 DriveInfo 对象来查询其磁盘信息将会引发 IOException 异常。为了避免发生此种情况，您可以先使用 DriveInfo 对象的 IsReady 属性来判断该磁盘是否已经准备好。

图 1-1

本范例的所有程序代码编写在按钮的 Click 事件处理函数中，列示如下：

```
private void btnGetDriveInfo _ Click(object sender,EventArgs e)
{
 // 显示一个状态信息对话框来表示我们目前要尝试取得计算机的磁盘信息.
 frmStatus frmStatusMessage = new frmStatus();
 frmStatusMessage. Show("处理中,请稍后 .... ");

 StringBuilder sb = new StringBuilder();

 // 声明 DriveInfo 类对象,并使用 GetDrives 方法取得目前
 // 系统中所有逻辑磁盘驱动器的 DriveInfo 类型数组.
 DriveInfo[ ] myAllDrives = DriveInfo. GetDrives();

     try
     {
      foreach(DriveInfo myDrive in myAllDrives)
      {
       // 使用 IsReady 属性判断磁盘设备是否就绪.
       if(myDrive. IsReady)
       {
        sb. Append("磁盘驱动器盘符:");
        sb. AppendLine(myDrive. Name);
        sb. Append("磁盘卷标:");
        sb. AppendLine(myDrive. VolumeLabel);
        sb. Append("磁盘类型:");
        sb. AppendLine(myDrive. DriveType. ToString());
        sb. Append("磁盘格式:");
        sb. AppendLine(myDrive. DriveFormat);
        sb. Append("磁盘大小:");
        sb. AppendLine(myDrive. TotalSize. ToString ());
        sb. Append("剩余空间:");
        sb. AppendLine(myDrive. AvailableFreeSpace. ToString());
        sb. Append("总剩余空间(含磁盘配额):");
        sb. AppendLine(myDrive. TotalFreeSpace. ToString());
        sb. AppendLine("------------------------------------------------------------");
       }
      }
     }
     catch(Exception ex)
     {
      MessageBox. Show(ex. Message);
     }

     frmStatusMessage. Close();
     txtResult. Text = sb. ToString();
}
```

附注 您可以使用 DriveInfo 对象的 RootDirectory 属性来取得磁盘驱动器根目录的 DirectoryInfo 对象。稍后我们会说明如何使用 DirectoryInfo 对象。

程序范例2

图 1-2 所示的是程序范例 CH1 _ DemoForm002. cs 的执行画面，它示范如何使用 DriveInfo 类来取得计算机上所装设的磁盘驱动器的相关信息(包括“已就绪”与“未就绪”的磁盘)并将这些信息显示于一个 DataGridView 控件中。现将程序代码列示如下：

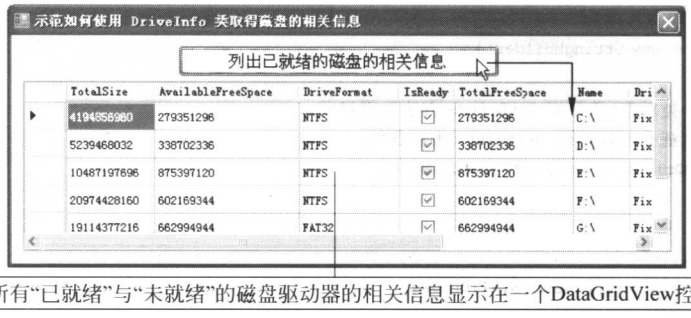


图 1-2

```
private void btnGetDriveInfo _ Click(object sender, EventArgs e)
{
 drivesDataGridView. DataError +=
   new DataGridViewDataErrorEventHandler(drivesDataGridView _ DataError);
 this. drivesDataGridView. DataSource = DriveInfo. GetDrives();
}

private void drivesDataGridView _ DataError(
System. Object sender, System. Windows. Forms. DataGridViewDataErrorEventArgs e)
{
 // 并非所有的磁盘驱动器都是就绪的. 我们忽略错误.
}
```

程序范例3

图 1-3 所示的是程序范例 CH1 _ DemoForm003. cs 的执行画面，它示范如何建立一个界面来让用户选取计算机上的某一个磁盘驱动器，并将所选取的磁盘驱动器的相关信息显示于一个 PropertyGrid 控件中。

相关设计重点说明如下：

- 我们于窗体的 Load 事件处理函数中编写下列程序代码，以便将包含所有磁盘的 DriveInfo 对象的 DriveInfo 类型数组新增至 ComboBox 控件中来成为其选项：

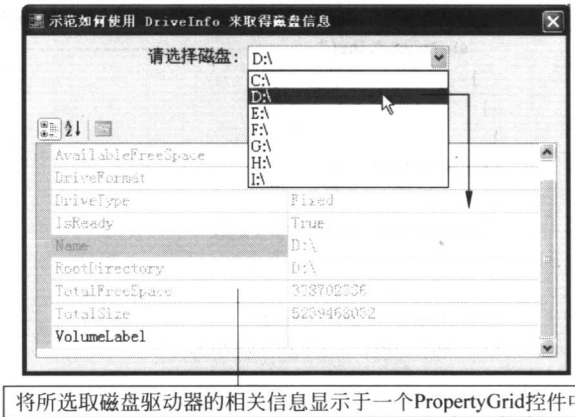


图 1-3

```
private void CH1 _ DemoForm003 _ Load(object
sender, EventArgs e)
{
```

```
// 声明 DriveInfo 类对象,并使用 GetDrives 方法取得目前
// 系统中所有逻辑磁盘驱动器的 DriveInfo 类型数组.
DriveInfo[] myAllDrives = DriveInfo.GetDrives();

// 将 myAllDrives 的内容赋给 ComboBox 控件当作选项显示.
this.cboDrives.Items.AddRange(myAllDrives);
}
```

- 我们为 ComboBox 控件的 SelectedIndexChanged 事件处理函数编写下列程序代码，以便当用户从下拉列表框中选取某一个磁盘驱动器盘符时，便会将该磁盘驱动器的相关信息显示于一个 PropertyGrid 控件中：

```
private void cboDrives_SelectedIndexChanged(object sender,EventArgs e)
{
 // 取得用户所选取磁盘的 DriveInfo 对象.
 DriveInfo theDriveInfo =
(DriveInfo)cboDrives.Items[cboDrives.SelectedIndex];

 // 检查磁盘是否已经就绪.
 if (theDriveInfo.IsReady)
 {
  this.drivesInfoPropertyGrid.SelectedObject =
    cboDrives.Items[cboDrives.SelectedIndex];
 }
 else
 {
  MessageBox.Show(
    theDriveInfo.Name + " 磁盘尚未就绪.","请注意",MessageBoxButtons.OK);
  this.drivesInfoPropertyGrid.SelectedObject = null;
 }
}
```

程序范例4

当我们在 Windows 资源管理器中使用鼠标右键单击磁盘驱动器并从快捷菜单中选取“属性”命令时，便会显示一个磁盘内容对话框，其中会以一个饼图来显示已使用空间和可用空间的比例，并且列出字节与 MB(或 GB)数值信息。程序范例 CH1_DemoForm004.cs 即是模仿此磁盘内容对话框制作出来的，图 1-4 与图 1-5 所示是其执行画面。

以下是程序范例 CH1_DemoForm004.cs 的设计重点：

- 当用户从下拉列表框中选取某一个磁盘驱动器盘符时，便会调用用户自定义函数 LoadDriveInfo() 并将用户所选取的 ComboBox 项目(亦即 DriveInfo 对象)传递给它。接下来会调用窗体的 Invalidate() 方法，以便强制重新绘制画面上的图形。程序代码如下所示：

```
private void drivesOnPc_SelectedIndexChanged(object sender,EventArgs e)
{
 LoadDriveInfo((DriveInfo)drivesOnPc.Items[drivesOnPc.SelectedIndex]);
 this.Invalidate();
}
```

- 用户自定义函数 LoadDriveInfo() 负责取得用户所选取磁盘的相关信息并将它们显示于窗体上的 Label 和 TextBox 控件中。程序代码如下所示：

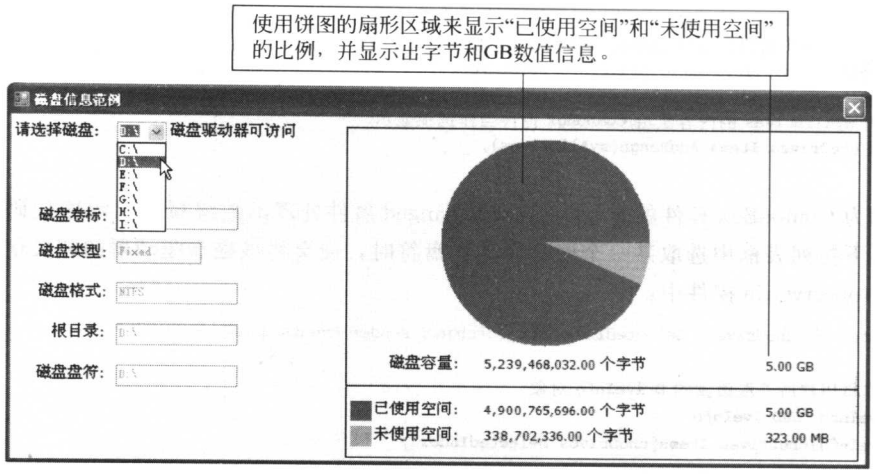


图 1-4

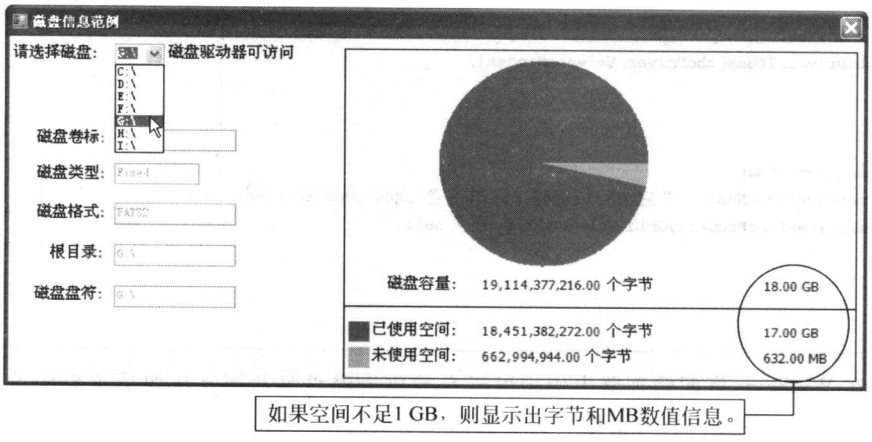


图 1-5

```
private void LoadDriveInfo(DriveInfo myDriveInfo)
{
 // 将磁盘驱动器名称显示在文本框中.
 this.driveName.Text = myDriveInfo.Name;

 try
 {
  // 判断 myDriveInfo 对象的磁盘盘符文字长度是否大于 0.
  if(myDriveInfo.VolumeLabel.Length > 0)
  {
   // 将磁盘驱动器盘符显示在文本框中.
   this.driveVolumeLabel.Text = myDriveInfo.VolumeLabel;
  }
  else
  {
   this.driveVolumeLabel.Text = "无";
```

```
}

// 将磁盘驱动器文件系统格式显示在文本框中.
this.driveFormat.Text = myDriveInfo.DriveFormat;

// 将磁盘驱动器大小赋给变量 totalSpace.
totalSpace = myDriveInfo.TotalSize;

// 将磁盘驱动器剩余空间赋给变量 freeSpace.
freeSpace = myDriveInfo.TotalFreeSpace;

// 将已使用空间赋给变量 usedSpace.
usedSpace = totalSpace-freeSpace;

// 计算剩余空间之扇形区域之第二个边的角度.
sweep = 360f * freeSpace /totalSpace;

isSpaceInfoAvailable = true;
}
catch
{
// 将文字显示在文本框中.
this.driveVolumeLabel.Text = "无法访问";

// 将文字显示在文本框中.
this.driveFormat.Text = "无法访问";

isSpaceInfoAvailable = false;
}

// 将磁盘驱动器类型显示在文本框中.
this.driveType.Text = myDriveInfo.DriveType.ToString();

// 将磁盘驱动器的根目录显示在文本框中.
this.driveRootDirectory.Text = myDriveInfo.RootDirectory.ToString();

// 取得根目录的 DirectoryInfo 对象.
dirInfo = myDriveInfo.RootDirectory;

// 判断磁盘驱动器是否已经就绪.
if (myDriveInfo.IsReady == true)
{
this.driveReadyStatus.Text = "磁盘驱动器可访问";
}
else
{
this.driveReadyStatus.Text = "磁盘驱动器无法访问";
}
}
```

➢ 由于调用窗体的 Invalidate() 方法来强制重新绘制窗体时会引发窗体的 Paint 事件，因此我们在窗体的 Paint 事件处理函数中编写下列用来绘制饼图的程序代码：

```
private void CH1_DemoForm004_Paint(object sender,PaintEventArgs e)
{
// 声明 Rectangle 结构变量,并指定 X 坐标参数、Y 坐标参数、宽度参数、高度参数,
// 用来显示圆饼图的矩形大小.
Rectangle rect = new Rectangle(410,20,200,200);
```