Microsoft

Core Reference

[美] Charles Petzold 著

章立民 译

Microsoft Windows程序设计

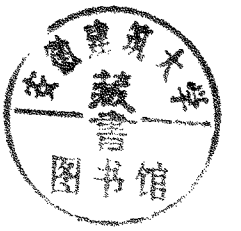
——Visual Basic .NET语言描述

Microsoft .net

华中科技大学出版社

Microsoft Windows 程序设计
——Visual Basic .NET 语言描述

Charles Petzold 著
章立民 译



Microsoft Windows 程序设计——Visual Basic.NET 语言描述
Programming Microsoft Windows with Microsoft Visual Basic.Net
Charles Petzold

图书在版编目(CIP)数据

Microsoft Windows 程序设计——Visual Basic.NET 语言描述/(美)Charles Petzold 著;章立民 译
武汉:华中科技大学出版社,2004 年 5 月
ISBN 7-5609-3142-1

Ⅰ.M…
Ⅱ.①C… ②章…
Ⅲ.BASIS 语言-程序设计
Ⅳ.TP312

责任编辑:徐 烨 曾 光　　封面设计:搜获科技
责任校对:吴 晗 封春英　　责任印制:熊庆玉
出版发行:华中科技大学出版社　(武昌喻家山 邮编:430074 电话:87557437)
http://press.hust.edu.cn
录 排:搜获科技
印 刷:湖北新华印务有限公司
开 本:787×1092 1/16　　**印张**:62.5　　**插页**:2　　**字数**:1 380 000
版 次:2004 年 5 月第 1 版　　**印次**:2004 年 5 月第 1 次印刷
ISBN 7-5609-3142-1/TP·523　　**定价**:118.00 元

(本书若有印装质量问题,请向出版社发行部调换)

前　言

本书是笔者的著作“Programming Microsoft Windows with C#”到 Microsoft Visual Basic .NET 的转换版本。这两本书除了示例编程代码之外基本上是相同的。

两本书皆在展示如何在 Microsoft Windows 之下撰写程序。有许多方法来撰写此类程序。在这两本书中，笔者使用一个称为 Windows Forms 的新类库。Windows Forms 类库是 Microsoft .NET Framework 的一部分，.NET Framework 的概念是于 2000 年夏天首次提出并于一年半之后正式问世的。

.NET Framework 是提供程序员撰写 Internet、Web、与 Windows 应用程序所需的类扩展集合。.NET 的许多媒体范围集中于 Web 编程。本书则是讨论 .NET 的其他部分。您可以使用 Windows Forms 撰写传统独立的 Windows Forms 应用程序(此类应用程序有时被称为客户端应用程序或分布式应用程序的前端)。

Windows Forms 几乎提供了撰写全方位 Windows 应用程序所需的所有功能。其最大疏忽就是多媒体支持，甚至没有任何 Windows Forms 功能能够让电脑的喇叭发生哔哔的声响。笔者曾经尝试去撰写自己的多媒体类，不过后来还是克制下来，因为笔者心理假设(希望这是合理的假设)下一个版本的 Windows Forms 会内含灵活、强大且易于使用的多媒体支持。

定义于 .NET Framework 中的类是跨语言的。在发表.NET Framework 时，Microsoft 也发布了可以使用这些类的 Visual Basic 与 C++新版本以及全新的编程语言 C#。其他程序语言的厂商正在修改它们自己的程序语言使其能够使用.NET 类。

.NET 的跨语言特性是由公共语言规范(CLS)所促成的，CLS 文档是描述使用 .NET Framework 的编程语言需要具有哪些特性。针对 .NET 所设计的编译器一般会将源代码转换成位于.exe 文件中的一个中间语言(Intermediate Language)。在运行时，中间语言会被 .NET 公共语言运行库(CLR)编译成适当的微处理器机器码。因此，NET Framework 可以说是跨平台的。

Windows 与 Basic

Microsoft 公司在 1985 年的秋天首度发表 Windows 的第一个版本。自此开始，Windows 就不断地更新与增强，在 Windows NT(1993 年)与 Windows 95(1995 年)中更是做了显著地改进，其中最重要的变革就是使 Windows 从 16 位架构迈向 32 位架构。

当 Windows 首次发布时，实际上只有一种方法来撰写 Windows 应用程序，也就是通过使用 C 编程语言来访问 Windows 应用程序接口(API)。虽然也能够使用 Microsoft Pascal 来访问 Windows API，不过很少使用此方法。

几年来，许多其他的程序设计语言也都顺势调整以便能够进行 Windows 编程。在 1991 年，Microsoft 公司发布用于 Windows 的 Visual Basic 1.0，此一革命性的产品允许程序员通过将控件拖放至窗体上来交互式地设计应用程序。Visual Basic 1.0 提供了一种比 C 程序员所使用的 API 更高级的编程界面。Visual Basic 在接下来的数年持续演进直到 1998 年推出 Visual Basic 6.0 为止。

Visual Basic .NET 代表从 Visual Basic 6.0 以来的变革(在语言本身并没有太大改变)，大部分仍然继续沿用，但是在程序与 Windows 交互的编程界面上则有长足进步。在 Visual Basic .NET 中，此界面是由实现于 .NET Framework 中的类库所提供的。

用 户 要 求

欲最有效地使用本书，您需要能够编译与运行 Visual Basic .NET 程序。欲编译这些程序，您需要一个 Visual Basic .NET 编译器。欲运行这些程序，您需要 .NET 运行库(也称为公共语言运行库，CLR)，CLR 是动态链接库的一个集合。

这两个项都已内含于称为 Microsoft Visual Basic .NET 的软件产品中，它是一个现代化的集成开发环境。另外，您可以购买更完整广泛也更昂贵的 Microsoft Visual Studio .NET，除了 Visual Basic 以外，它还能够让您使用 C++与 C#来撰写程序。

如果您喜爱较简单的做法，可以下载免费的 .NET Framework 软件开发工具包(SDK)。此下载内含一个命令行的 Visual Basic .NET 编译器与 .NET 运行库。首先，请到网站 *http://msdn.microsoft.com/downloads*，然后从左侧选取 Software Development Kits，并找到 .NET Framework SDK。您或许需要选择 .NET Framework Redistributable 页面然后在那里找到一个链接(请牢记，此 Web 站点与本书所提及的其他所有 Web 站点一样，都会经常变更、移动甚至完全消失)。

笔者撰写本书时系假设您至少了解如何在较早版本的 Visual Basic 中撰写程序，而且假定您对面向对象语言的概念非常熟悉。笔者在第 1 章花了许多功夫让您了解 Visual Basic .NET 的新特性。

笔者在本书中有时候会引用 Windows API。您可以阅读笔者的另外一本著作“Programming Windows”(微软出版社于 1998 年出版，第五版)学习更多有关 Windows API 的知识。

系 统 要 求

正如笔者在前节所提，欲最有效地使用本书，您需要会编译与运行 Visual Basic .NET 程序。系统要求列示如下：

- Microsoft .NET Framework SDK(最小需求)；Microsoft Visual Basic .NET 或 Microsoft Visual Studio .NET(建议需求)。
- Microsoft Windows NET 4.0、Windows 2000 或 Windows XP。

欲在其他电脑上运行您的 Visual Basic .NET 程序，这些电脑必须安装 .NET 运行库(即.NET Framework 可重新发布包)。该包内含于 .NET Framework SDK、Visual Basic .NET 与 Visual Studio .NET 中。可重新发布包可安装于刚刚所提过的 Windows 版本以及 Windows 98 与 Windows Me 中。

正如我们稍后所会讨论的，本书的范例文件可以自微软出版社网站下载。如果您要将它们安装于硬盘中，需要大约 3.5 MB 的额外硬盘空间(如果全部加以编译，这些范例所占用的空间将会超过 13 MB)。

本书的编排

当 Windows 1.0 首次发布时，整个 API 是实现于 3 个名称为 KERNEL(核心调用)、USER(用户)与 GDI 的动态链接库中。虽然这些与 Windows 关联的 DLL 已经变得非常庞大，但是若将 Windows 函数调用(或框架类)区分成这 3 类仍然是非常有用的。核心调用就是那些实现于操作系统的架构内部中者，而且一般会与任务、内存管理以及文件 I/O 有关。术语用户是指用户界面。这些是用来建立窗口、使用菜单与对话框以及显示按钮与滚动条等控件的函数。GDI 是图形设备接口(Graphics Device Inferface)，它是 Windows 用以负责在屏幕与打印机上显示图形输出(包括文本在内)的部分。

本书先从 4 个介绍性的章节开始。从第 5 章(说明如何绘制直线与曲线)开始并持续至第 24 章(主要介绍 Windows 剪贴板)，这些章节将会在图形主题(奇数章节)与用户界面主题(偶数章节)之间交替。

一般来说像本书这样的书籍并不会花费太多时间与篇幅来介绍非 Windows 主题，例如：文件 I/O、浮点数运算与字符串处理。然而，因为 .NET Framework 非常新，笔者发现自己期望能在这些类上有前后连贯的说明。因此笔者自己撰写了这样的说明。它们会内含于文件、数学与字符串的 3 个附录中。您可以在读完第 1 章之后，在需要了解它们的任何时候参考这些附录。

笔者试着去编排这些章节的先后顺序(以及各章中主题的顺序)以使每个主题循序渐进，而让“向前引用”的次数降至最低。笔者将本书撰写成可以很自然地一路读下去，就好比在阅读 The Stand 或 The Decline and Fall of the Roman Empire 一样。当然，如果像本书这么多篇幅的书籍能够当作参考书籍来查阅也是很不错的。出于此原因，许多使用于 Windows Forms 程序设计中的重要方法、属性与枚举皆会在讨论它们时列于表格中。然而，即使是本书这样篇幅的书籍也无法彻底讨论到 Windows Forms 中的所有技术知识。本书不能取代 Microsoft 官方的类文档。

Windows Forms 程序需要费一番功夫学习，为了帮助学习，本书内含丰富且完整的范例程序。您可以自行将这些程序中的代码片断粘贴至程序中(这也正是这些范例程序的主要目的)，但是请不要随意分布这些程序或其中的代码。

如同在 Visual Basic 的早期版本中，Visual Basic .NET 允许互动式地设计应用程序的外观。您可以将不同的控件(如按钮和滚动条等)放置在窗口的表面上，而 Visual Basic .NET 会自动生成代码。虽然这样的技术在设计对话框与前端面板类型的应用程序时

非常有用，但是本书将不采用 Visual Basic .NET 的此项特性。

在本书中，我们不会让 Visual Basic .NET 替我们生成代码，我们将学习如何自行撰写所有的代码。

支　持

本书的范例程序可以从出版社网站 *http://www.microsoft.com/ mspress/books/6259.asp* 下载(笔者个人的网站 *http://www.charlespetzold.com* 也有一个该网页的链接)。您可以将解决方案文件(.sln)或项目文件(.vbproj)下载至 Visual Basic .NET 中并重新编译这些程序。

我们已经竭尽所能来确保本书与源代码内容的正确性，微软出版社会通过如下网址的全球信息网来提供本书的勘误：

http://www.microsoft.com/mspress/support/

欲直接连接至微软出版社知识库并输入您的问题或结论，请访问：

http://www.microsoft.com/mspress/support/search.asp

如果您对本书有任何建议，问题或想法，请使用下列方式将它们传送至微软出版社：

邮寄：

Microsoft Press

Attn: Programming Microsoft Windows with Microsoft Visual Basic .NET Editor One Microsoft Way

Redmond, WA 98052-6399

电子邮件：

MSPINPUT@MICROSOFT.COM

请注意以上邮件地址并不提供产品支持。要获得关于 Visual Basic .NET，Visual Studio .NET 或 .NET Framework 的支持信息，请访问 Microsoft 产品支持 Web 站点：

http://www.microsoft.com

如何转换一本书

笔者在 2001 年的 11 月完成了 Programming Microsoft Windows with C#。在 2002 年 2 月初，微软出版社询问我是否有兴趣将本书转换成 Visual Basic .NET。我认为这是个蛮有趣的计划，因此立即同意这么做。

本项目的第一个工作是撰写一个程序(以 C# 撰写)将书中的 C#程序转换成 Visual Basic .NET。C#这本书含有 300 个以上的源代码文件，因此我知道不可能将此工作全部自动完成，Visual Basic .NET 这本书显然要花很长的时间来完成。我的第 1 个转换程序是一个片段，草率而且是特别针对我的 C#编程风格设计的，忽略了我未曾使用过的程序语言特性。一个星期后，我决定重新撰写它，但是在那之前它已成功地转换了 95%的代码，所以我决定通过“查找”与“替换”操作的协助手动转换其余的部分。

C#是一个区分大小写的编程语言，而且在范例程序中笔者也确实使用此特性来命名变量。比方说，如果我只拥有一个 Font 类型的对象，我会将它命名为 font。起初我假设 Visual Basic .NET(不会区分大小写的编程语言)不容许这样的事情，但是我错了。编译器接受一个名称为 font 的对象。然而，如果 font 在一开始没有正确定义，Visual Basic .NET 编辑器会通过将第一个字母改成大写来修正它以便使其符合 Font 类！为了避免造成不习惯程序语言会区分大小写的程序员会混淆，我决定不将类名称复制于对象名称。我的字体对象会成为 fnt 对象，而且在此过程中，我认为本书在对象命名方面已比 C#那本书具有更高程度的一致性。

我遇到的 Visual Basic .NET 是否有功能不足之处呢？其实非常少。在 C#中，算术运算符(＋、－等等)与比较运算符(＜、＞等等＝可以被结构与类重载。例如，可以使用一般的加号来相加 Size 对象。但在 Visual Basic .NET 中，必须使用 op_addition 方法。有类似的方法用于比较对象与转换它们。

Visual Basic .NET 中不带正负号整数的缺乏很少影响代码的转换。笔者在第 12 章的 HexCalc 程序中使用一个不带正负符号的长整数，但是它可以很容易就修改成带正负号的长整数。HexCalc 程序中比较严重的问题是使用 C#符号来标示计算机的按钮。

进行类似这样的转换很容易就显现出两种程序语言间非常有趣的不同点。例如，在 C#中，方法默认是专用的；而在 Visual Basic .NET 中，默认是公用的。哪一个的做法较好着实让笔者思量了好一会儿，结论是各有利弊。方法应被默认保护的。应该有一个理由(以及一个关键字)来使特定的方法是共用的或专用的。

在本书的文本部分。笔者用标签笔动手处理了 Programming Microsoft Windows Withith C#热销的副本，标出了笔者能发现的每个很小的 C#关键字、代码片段和以 C#为中心的概念。为了转换本书中许多方法的表格，笔者撰写了一个 VBScript。虽然它没有办法完完全全地正确运作，但是也让工作简化了不少。Microsoft Word 文件的其他变更则是手动完成，使用全局搜索只是为了再次检查笔者的人工努力(最常用的单一项需要改变吗？C#的 static 关键字会成为 Shared)。

本书有一些需要改变的地方起初并不明显。正如您所见到的，当翻阅本书的各页时，您会发现有许多属性的表格。在笔者最初替 C#这本书设计这些表格的格式时，我是将属性的类型摆在第一列并将属性本身摆在第二列。这是 C#程序员在源代码中所看到的顺序。但是这并不是 Visual Basic .NET 程序员所会见到的。因此这些表格的前两列必须对调以便先显示出属性名称，然后才是类型。

我的编辑与我是否已从本书去除掉所有 C#的痕迹呢？我们不敢完全保证。就在前几天(在我同意本书付梓之前最后一次校对章节与附录时)我就发现了一个分号(C#使用分号作为语句的结尾)与双斜线(C#的注解符号)。希望不会有太多类似之处逃过我的搜索。

在.NET 发表之前，将一个 C 或 C++ Windows 编程书籍转换成 Visual Basic 可说是不可思议的。这都要归功于公共语言规范(CLS)与 .NET Framework，它们确实使得此项转换工作更加顺畅。Microsoft 公司确实也创造出一个系统，那就是编程语言的选择完全只是个人的喜好而已。

目　录

第 1 章　控制台本身

在 1964 年，一项英文全名为 Beginner's All Purpose Symbolic Instruction Code(简称 BASIC)的全新程序设计语言以下列的范例程序首次问世：

```
10 LET X=(7+8)/3
20 PRINT X
30 END
```

BASIC 由 John G. Kemeny 与 Thmoas E. Kurtz 这两位数学家在新罕布什尔州的达特茅斯学院开发出来。虽然此语言设计者是数学家，然而他们却希望创建一个系统以便提供给达特茅斯主修文科专业的学生以友好且简单的访问。在当时，大多数的程序仍旧使用穿孔卡片，Kemeny 与 Kurtz 却已在连接至电传打字机的通用电子计算机上创建一个互动式且省时的系统。BASIC 编程语言拥有 ALGOL 与 FORTRAN 的元素，但是在机器效率上加强了。学生在电传打字机上输入程序并输入 RUN 来编译与运行程序(第一个 BASIC 系统并不是如大家对 BASIC 既有印象的解释器，但是许多后续用于小型计算机与个人计算机的 BASIC 实现则是编译式的)。

虽然此时回顾起来，原始的 BASIC 是运行于目前看来怪异的旧式之纯文本的计算机界面中，也即所谓的命令行或控制台。每当用户在电传打字机上的键盘上输入数据时，设备便会在滚筒纸上印出字符并将它们传送至远端计算机。计算机会以它本身的字符来响应，电传打字机会加以接收便显示在纸张上。在此输入 / 输出模式中，并没有将文本定位于页面特定位置的概念。PRINT 语句仅仅是在电传打字机打印标题之处(或是之后，命令行光标所在位置)显示出文本。

经过这么多年的演变，BASIC 已进化至今日的 Visual Basic .NET，此程序语言也已历经众多的变革，非昔日的 BASIC 所能相比。BASIC 原先并不会区分整数与浮点数，然而现今它已被视为是较严谨且实用的程序语言，并在变量被使用之前先定义其数据类型。时至今日我们也见识到它将代码与数据归类至函数、程序、模块、结构与类的智慧。

LET 关键字(早已被舍弃多年)用来指示一个赋值语句，而且会被使用于类似如下的语句：

```
X=X+1
```

显然这很容易造成初学者的混淆。PRINT 语句则仍旧保留于 Visual Basic .NET 中，然而它大多数仅使用于文件。在 BASIC 中，每一行代码开头的数值是使用于电传打字机行编辑器的一部分，并且是 GOTO 语句的目标。在程序行的开头处显示或不显示出数值可帮助操作系统来辨识程序语句与命令。

1.1 控制台的返回

Visual Basic .NET 将 Beginner's All Purpose Symbolic Instruction Code 带至一个前所未有的境界。正如您所见到的，Visual Basic .NET 是一种真正面向对象的编程语言。

在此同时，更多传统的编程元素已被重新引入语言与.NET Framework 中。尤其是，Visual Basic .NET 重新拾回控制台 I/O 的概念。控制台程序善用 Windows 中您所熟悉的命令提示符或 MS-DOS 提示符的命令行界面。虽然命令行界面已因为图形界面而被舍弃不用，但是命令行程序却比针对图形环境所撰写的程序更加简易，这是因为它们有一个好的地方开始学习一个全新的程序设计语言，或是演练某一项技巧。

1964 年所推出的第一个 BASIC 程序若转换至 Visual Basic .NET 将会如下所示：

FirstBasicProgram.vb

```
'-----------------------------------------------
' FirstBasicProgram.vb (c) 2002 by Charles Petzold
'-----------------------------------------------
Module FirstBasicProgram
     Sub Main()
        Dim X As Single = (7 + 8) / 3
        System.Console.WriteLine(X)
     End Sub
End Module
```

您可以使用许多种方式来编译此程序，而这取决于您要花多少钱以及您所要求的方便性。

最简便的方法是从 *http://msdn.microsoft.com* 下载.NET Framework 软件开发工具包(SDK)(在左侧，选取 Downloads，接着选取 Developer Downloads，然后选取 Software Development Kits，最后是 .NET Framework SDK。大约 130 MB)。安装 SDK 也会安装动态链接库(DLL)以便生成 .NET 运行时环境。.NET 技术文档可用于基于 Windows 的程序。您也可使用命令行 Visual Basic .NET 编译器来编译在这些页面中显示的程序。

如果您采用 SDK 的方法，可以使用任何的文本编辑器(例如记事本)来编写 Visual Basic .NET 程序。Visual Basic .NET 编译器的名称是 vbc.exe。您可以使用下列的命令行来编译 FirstBasicProgram.vb：

```
vbc firstbasicprogram.vb
```

如此即可。其中并不包括链接步骤(正如您在下一章所见到的，编译一个 Windows Forms 程序而不是一个控制台程序时会需要额外的编译器参数)。编译器会产生一个名为 firstbasicprogram.exe 的文件，您可以在命令行运行它。

您也可在 Visual Basic .NET 中(也即最新版本的 Microsoft Visual Studio .NET 集成开发环境中)创建、编译与运行此程序。Visual Basic .NET 对专业的 Visual Basic 开发者而言是不可或缺的。对特定类型的 Windows Forms 程序(也即将程序的窗口视为是一个包含按钮、输入字段、滚动条等控件的窗体)而言，集成开发环境显得特别有用。然而，它也并非是绝对必要的。我们已经发现使用 Windows Forms 库来进行 Windows 编程的真正乐趣

是不需要使用个别的文件。几乎所有的操作都在源代码文件中完成，而且那些文件中的所有操作都可以通过您自己的手指与大脑来完成。

接下来的段落将描述笔者使用 Visual Basic .NET 与 Visual Studio .NET 来创建本书中程序采取的相关步骤。当笔者实际谈论到两个环境时，一般指 Visual Basic .NET。然而，如果您运行 Visual Basic .NET 且与 Visual Studio .NET 相冲突，则您必须在安装中添加两个小文件才能遵照本书所述的步骤来进行。这两个文件的名称是 VisualBasicEmptyProject.vsdir 与 VisualBasicEmptyProject.vbproj，它们位于您从网络上所下载本书源代码的根目录中，请将这两个文件复制到硬盘的此目录中：

```
C:\Program Files\Microsoft Visual Studio .NET\vb7\VBProjects
```

如果您已经安装 Visual Studio .NET 而不是只安装 Visual Basic .NET，则不需要这样做。

本书中的每个示例程序皆是 Visual Basic .NET 项目，而且每一个项目都有自己的磁盘目录。在 Visual Basic .NET 中，项目通常被组织到解决方案，而笔者已替本书的每一章分别创建了一个解决方案。每一个解决方案也是一个目录，项目则位于解决方案所在目录的子目录中。

也是欲创建一个解决方案，选择【文件】|【新建】|【空解决方案】命令。在【新建项目】对话框中，选取解决方案的磁盘位置并输入解决方案的名称。而这就是笔者替本书之各章创建解决方案的方式。

当您已于 Visual Basic .NET 中加载了一个解决方案，便可以在解决方案中创建项目。选择【文件】|【添加项目】|【新建项目】命令(或是在【解决方案资源管理器】中右击解决方案的名称，并从快捷菜单中选取【添加 / 新建项目】命令)。在【添加新项目】对话框中，选取 Visual Basic 项目的项目类型。您可以选取各种模板。如果要避免 Visual Basic .NET 生成代码(笔者个人偏好自行撰写代码)，请选取【空项目】这一模板。事实上笔者正是以此方式创建本书的各个项目(空项目会内含于 Visual Studio .NET 中，但是并不会内含于 Visual Basic .NET 中。您复制的两个文件会替 Visual Basic .NET 添加空项目功能)。

在项目中，您可以使用【项目】|【添加新项】命令来创建一个新的 Visual Basic 源文件(也或是在【解决方案资源管理器】中右击项目的名称，并从快捷菜单中选择此项)。在【添加新项】对话框中，从【分类】列表中选择【本机项目项】，并于【模板】列表中选取【代码文件】。如果您选取此模板，Visual Basic .NET 并不会替您生成代码。

欲设置项目的属性，请从【项目】菜单中选取【属性】命令，或在项目名称上右击并选择【属性】命令。在左侧【通用属性】项下选择【常规】。在右侧，确认输出类型是【控制台应用程序】。如果正运行 Visual Studio .NET(而并非是 Visual Basic .NET)，将必须自行从【输出类型】列表中选取【控制台应用程序】选项。当笔者创建本书的项目时，也会选中【通用属性】项下的【生成】项，并将 Option Explicit 设置成 On。事实上 Option Explicit 的默认设置就是 On。这些选项会使 Visual Basic .NET 编译器运行一些额外且笔者个人认为非常重要的一致性检查。在命令行编译时如果需要启用这些选项，请如下所示替 vbc.exe 加上所需的选项：