高级.NET开发系列

Visual Basic是微软开发工具的基石之一。本书是Visual Basic .NET语言的权威参考指南。

——Bill Gates，微软公司董事长

The Visual Basic .NET Programming Language

Visual Basic .NET 程序设计语言

[美] Paul Vick 著
欧阳宇 译

- Visual Basic .NET之父Paul Vick最新力作，Bill Gates大力推荐
- 最深刻阐述Visual Basic .NET工作机理和强大功能
- 新程序员最佳学习工具
- 资深开发人员必备参考书

中国电力出版社
www.infopower.com.cn

高级.NET开发系列

The Visual Basic .NET Programming Language

Visual Basic .NET 程序设计语言

[美] Paul Vick 著
欧阳宇 译

中国电力出版社
www.infopower.com.cn

The Visual Basic .NET Programming Language (ISBN 0-321-16951-4)
Paul Vick

北京市版权局著作权合同登记号 图字：01-2004-5253

图书在版编目（CIP）数据
Visual Basic .NET 程序设计语言 /（美）维克（Vick, P.）著；欧阳宇译. —北京：中国电力出版社，2004
（高级.NET 开发系列）
ISBN 7-5083-2506-0
I.V... II.①维...②欧... III.BASIC 语言—程序设计 IV.TP312
中国版本图书馆 CIP 数据核字（2004）第 089511 号

丛 书 名：高级.NET 开发系列
书　　名：Visual Basic .NET 程序设计语言
编　　著：（美）Paul Vick
翻　　译：欧阳宇
责任编辑：朱恩从
出版发行：中国电力出版社
　　地址：北京市三里河路 6 号　　邮政编码：100044
　　电话：（010）88515918　　传　　真：（010）88518169
印　　刷：北京丰源印刷厂
开　　本：787×1092　1/16　　印　　张：19
书　　号：ISBN 7-5083-2506-0
版　　次：2005 年 1 月北京第 1 版　　2005 年 1 月第 1 次印刷
定　　价：35.00 元

前　言

恐怕没有哪家公司曾像 Microsoft 公司一样，将自身的命运与一种编程语言（BASIC）密切地捆绑在一起。自从 25 年前 Bill Gates 和 Paul Allen 创立 Microsoft 并出售他们的 Altair MIPS BASIC 解释程序以来，这家公司的成败就和 BASIC 编程语言联系在了一起。从 MS-DOS 到 Windows 再到 .NET Framework，BASIC 对于每个重要 Microsoft 平台都起了重要的作用，吸引了数以百万计的开发人员。

在 Microsoft 开发的所有 BASIC 产品中，最成功的要数 Visual Basic。Visual Basic 的推出对 Windows 平台和 Office 办公套件的非凡成功发挥了很大作用。这种语言在众多计算机语言中拥有最多的开发人员——总计超过 300 万，使其成为有史以来最成功的计算机编程语言。Visual Basic 之所以获得成功，很大程度上源于它的基本设计原则：简单、直接和易用。Visual Basic 的设计，使无论刚入门的程序员还是熟练的程序员都可以使用。它朴实易懂，使得学习计算机编程很容易；同时又为开发最先进的应用程序提供了很高的生产率和效能。作为一种通用的语言，它是任何程序员开发工具箱中必备的工具。

关于本书

本书详细介绍了 Visual Basic .NET 编程语言，可以作为新程序员学习 Visual Basic .NET 编程的入门读物，也可供有经验的 Visual Basic .NET 程序员参考。总的来说，本书没有涉及和语言本身无关的话题，比如如何使用 Windows Forms 库进行 GUI 编程，或者公共语言运行库的底层设计。它首先关注的是语言本身，只有在为了理解语言设计的某些部分必需的情况下才讨论语言之外的内容。

本书的安排如下：给出语言的全面概述之后，按照从简单到高级的顺序安排各个概念。本书是按照顺序阅读的方式编写的，但是有一定 Visual Basic 编程经验的读者也可以随意翻阅。本书首先从语言的基本概念如语句和表达式开始讲起，然后转到语言的面向对象编程（OOP）特性，如类。最后讨论继承和版本这类高级主题。熟悉以前版本 Visual Basic 的读者最好也逐章阅读，转移到 .NET Framework 上使该语言的很多地方发生了变化，即使熟悉的主题也可能包含新的特性。

总体而言，本书对概念的讨论都很详细。对于初学 .NET Framework 或者 Visual Basic 编程的读者，一些细节可能难以掌握。书中的高级概念通常都明显地标记了出来，在运用语言时

不需要理解这类概念——事实上读者完全可以跳过这些部分。读者无需徒劳地纠缠于一些细节上，可以先将相应的内容忽略，继续读下去，等以后再返回来研读。多数情况下，领会了后面的概念后前面的一些高级概念会更容易理解。

> **提示**
> 本书所讨论的 Visual Basic .NET 语言和 Visual Studio .NET 2003 以及 .NET Framework 1.1 对应。

非正式的 Visual Basic 简史

BASIC 编程语言最初是在 1964 年由 Dartmouth 大学的 John G. Kemeny 和 Thomas E. Kurtz 定义的。BASIC 是 Beginner's All-purpose Symbolic Instruction Code（初学者通用符号指令编码）的缩写形式，其设计目标是既便于初学者学习，又有足够的能力编写通用程序。1975 年，两个程序员（Bill Gates 和 Paul Allen）开发了 BASIC 的 Altair MIPS 版本，并通过他们创建的一家小公司销售。这个 BASIC 版本帮助 Microsoft 取得了非凡的成功，从那以后 Microsoft 开始提供各种形式的 BASIC 产品。

1991 年，随着 Visual Basic 的推出，Microsoft 的 BASIC 迈出了重要的一步。根据 Alan Cooper 最初的开发理念，Visual Basic 把 BASIC 语言的一个版本和新的 Windows 用户界面紧密地结合在一起，使其成为开发 Windows 应用程序的强大工具。这种思想获得了成功！今天，Visual Basic 是现有的应用最广泛的编程工具。第四版 Visual Basic 产品以通用对象模型（COM）的引入为标志，COM 是开发能够互相插接的组件的一个框架。COM 也获得了极大成功，称为 Windows 的基础技术，被用于构建数以百万计的组件。

虽然 COM 取得了很大成功，但其本身也存在局限性。它最大的局限是除了 Visual Basic 之外，和其他语言的协作一直不成功。比方说，C++程序员常常发现对 COM 组件进行编程非常困难而且费时，特别是那些用 VB 编写的组件。另一个问题是，COM 仅仅定义了组件间如何彼此交互，而把这些交互的细节，特别是组件生存期和内存管理这类工作，留给了组件的作者。Visual Basic 为其程序员提供这些服务，但是对其他流行语言如 C++却没有。因为每个 C++程序员都需要自己实现这种服务，因而实现不一定兼容或者完全不兼容，其结果是不仅增加了工作量，而且造成了更多的缺陷（进而给用户带来更多的烦恼）。

为了克服 COM 的缺点，Microsoft 开始开发一种替代技术——CLR（Common Language Runtime，公共语言运行库）。除了定义组件的交互之外，CLR 还代替程序员负责管理内存和组件生存期这类工作。同时，Microsoft 还开始开发 .NET Framework，它是基于 CLR 的一个类库，提供和 Win32 API 等价的功能。.NET Framework 和 CLR 共同组成了下一代 Windows 编程的基础。

虽然 CLR 的目的是取代 COM，但两者之间还有很多不同，一些明显，而另一些则很微妙。因为 Visual Basic 和 COM 的结合非常紧密，所以这种语言也需要做相应的修改，其中一些修改对 VB 的编程方式有着显著的影响。于是一种新的、不同的 Visual Basic 语言版本（Visual

Basic .NET）诞生了。尽管在这次迁移中语言的多数成分都被保留了下来，但是语言的一些细节发生了变化，并增加了许多新的概念。

在可行的情况下，那些有别于以前 Visual Basic 版本的内容放在标记为“兼容性”的方框中给出，但是 Visual Basic 6.0 和 Visual Basic .NET 的详细区别不在本书讨论之列。

风格与建议

和 40 年前刚刚出现的时候相比，编写 BASIC 代码的风格和习惯有了很大不同，并且这种改变仍在继续。和其他编程语言相比，BASIC 更适用于所能想像得到的各种用途，以及各种可能的计算环境。它可能是至今人们所开发的最不具标准性的计算机语言，因此为该语言的风格和用法提供某种建议似乎很奇怪，但无论如何这都是本书的目标之一。在上一节中我们已经阐明，Visual Basic .NET 代表了 Visual Basic 语言的一次巨大飞跃。有些成分被丢弃了，有些做了很大修改，而另一些则是全新的。本书为如何最好地应用这种语言提供了所有可能的建议，并说明了建议这样做的理由。这些建议中的一部分是从实践的角度提出的，而另一些则纯粹是风格性的，完全依赖于个人的喜好。当然，读者完全可以不管其中的一些建议——编程的乐趣之一就在于自由地编写代码，只要你认为合适就行。

鸣谢

没有那些为 Visual Basic .NET 的创建作出贡献的人们，就不会有本书。特别感谢那些和我一起参与 Visual Basic .NET 语言设计的核心小组的成员，尤其是 Alan Carter、Sam Spencer、John Hamby 和 Cameron McColl。从 Visual Basic 到 Visual Basic .NET 是一次漫长而极其困难的跃迁，他们克服了巨大的困难，坚持到底。没有他们不断的努力和对产品所抱有的激情，我们可能永远也无法完成。

在此还要感谢整个 Visual Basic .NET 团队，他们把语言的设计转化成现实，以及 C#团队——我们的同伙（就像以往那样）。特别感谢 Anders Hejlsberg、Peter Golde 和 Scott Wiltamuth，他们从截然不同的角度对语言的设计进行了阐述。CLR 和 .NET Framework 团队对 Visual Basic .NET 的成就起了无法估量的作用，诚挚感谢 Brad Abrams 和 Jim Miller 的努力，他们耐心地花费了很多时间来解释这种新平台的各个方面。

没有 Addison-Wesley 的 Stephane Thomas 和 Michael Mullen 的热情支持和长久耐心，以及 Martin Heller 的编辑技能，本书也不可能面世。感谢 Ken Getz、Amit Kalani、Rex Jaeschke、Rocky Lhotka、David Vitter、Phillip Williams、Ethan Roberts、Joe Hummel 和 Klaus Probst，他们热情地审阅了本书，提供了有帮助、有启发的意见和建议。没有 Julia Liuson 和 Paul Kuklinski 的积极努力，使我有时间撰写本书（并知道其他事情不一定要马上做），它可能永远不会完成。

最后，没有我的家人，特别是我的妻子 Andrea 的支持，这一切都是不可能的。*没有任何话语能够表达我对她的谢意。*

目　录

1

语言概述

为了在较短时间内使读者对 Visual Basic .NET 语言的特性有一个大致了解，并帮助新程序员尽快着手编写程序，本章从整体上简要地概述了 Visual Basic .NET 语言。本章涉及的所有内容在后面的各章中都分别有更详细的讨论，如果在学习 Visual Basic .NET 语言之前不需要或者不愿意阅读关于该语言的概述，则完全可以跳过这一章，从第 2 章开始阅读。

1.1 Hello,world!

按照由来已久的传统，编程语言书籍的第一个例子都是下面这样的程序。

```
Module Helloworld
  Public Sub Main()
    Console.writeLine("Hello, world!")
    Console.ReadLine()
  End Sub
End Module
```

这个 Visual Basic .NET 程序只是向输出窗口中写入文本“Hello,world!”，然后等待用户按下回车键。它声明了一个模块 HelloWorld，该模块只包含一个方法 Main，程序运行时即执行此方法。Main 方法调用了 System.Console 类上的系统定义方法 WriteLine。WriteLine 方法接收一个字符串，并把该字符串写入输出窗口。然后调用 System.Console 类中的系统定义方法 ReadLine。该方法等待用户输入一个字符串并按下回车键。ReadLine 调用保证输出窗口不会立刻消失。

虽然本书没有详细讨论 Visual Basic .NET 程序的编译、执行和调试，但简要指出其中的一些要点会有所帮助。如果在机器上作为 Visual Studio .NET 的一部分安装了 Visual Basic .NET，则可以通过以下步骤编译和运行该程序。

1. 启动 Visual Studio .NET。
2. 新建一个 Visual Basic .NET 控制台应用程序项目。
3. 用上述代码替换 Module1.vb 文件中的代码。
4. 按下 F5 键运行该程序。

如果你的机器上只安装了 .NET Framework，则可以按照下面的步骤编译并运行该程序。

1. 使用记事本之类的文本编辑器创建一个名为 HelloWorld.vb 的新文件，然后输入前述代码。
2. 找到 .NET Framework 提供的 Visual Basic .NET 编译器 vbc.exe。
3. 在命令提示符下输入 vbc.exe HelloWorld.vb，对建立的源文件运行 Visual Basic .NET 编译器。
4. 在命令提示符下输入 HelloWorld，运行得到的程序 HelloWorld.exe。

运行 Visual Basic .NET 编译器把源代码从文本编译成程序集。程序集是扩展名为 EXE 或者 DLL 的文件，.NET Framework 能够加载和运行它。程序集有一个名字（本例中是 HelloWorld），用于将其与其他程序集区分开，并帮助 .NET Framework 确定在运行某个程序时需要加载哪些程序集。编译器在程序集中创建了两方面的内容：IL（intermediate language，中间语言）指令和元数据。IL 是 .NET Framework 能够理解并且能够执行的一种机器语言。除了 IL 中存储的内容之外，元数据还提供了关于上述程序的其他信息。.NET Framework 使用元数据保证 IL 可以正确和安全地执行，.NET Framework 反射 API 也会用到元数据。通过阅读 Damien Watkins、Mark Hammond 与 Brad Abrams 合著的《Programming in the .NET Environment》（Addison-Wesley，2003）可以进一步了解 .NET Framework 的内部机理。

1.2 基本类型

Visual Basic .NET 语言预定义了可用于执行数值、Boolean 和文本操作的基本类型（参见表 1-1）。基本数值类型分为三种：不同范围的整数（Byte、Short、Integer 和 Long）、不同范围和精度的浮点数（Single、Double），以及固定范围和精度的十进制数（Decimal）。

表 1-1 基 本 类 型

类型	说明
Boolean	布尔值（true/false）
Byte	1 字节无符号整数
Short	2 字节带符号整数
Interger	4 字节带符号整数
Long	8 字节带符号整数
Decimal	16 字节十进制数
Single	2 字节单精度浮点数
Double	4 字节双精度浮点数
Date	时间/日期值
Char	单个 Unicode 字符
String	Unicode 字符串

该语言定义了可用于对数值类型执行标准算术运算的运算符（参见表 1-2）。

表 1-2 数 值 运 算 符

运算符	说明
+	加法
-	减法和负数
*	乘法
/	浮点除法
\	整数除法
Mod	整数求余
^	幂运算
Not	位反
And	位与
Or	位或
Xor	位异或

Boolean 类型代表一个 true/false 值，可用于表示条件表达式的结果。逻辑运算符在表 1-3 中列出。

表 1-3 逻 辑 运 算 符

运算符	说明
Not	逻辑非
And	逻辑与
Or	逻辑或
Xor	逻辑异或
AndAlso	短路逻辑与
OrElse	短路逻辑或

表 1-4 中的比较运算符可用于组成比较表达式。

String 和 Char 类型分别表示 Unicode 字符串和字符，字符串运算符列在了表 1-5 中。

字符串比较的特殊之处在于有两种不同的求值方式。如果文件开头指定了 Option Compare Binary 语句，则字符串按照代码点进行比较。也就是说，如果两个字符串中每个对应的 Unicode 字符的值完全匹配，则两个字符串匹配。如果在文件开头指定了 Option Compare Text 语句，则字符串按照和地区有关的方式比较。也就是说，如果每个字符在运行时的当前 Framework 文化区域中被认为是等价的，则两个字符串匹配。在一些文化区域中，两个不同的 Unicode 值可能表示同一个字符，因此文本比较的结果可能与二进制比较不同。如果在文件开始没有指定 Option Compare 语句，则默认为 Option Compare Binary。

表1-4 比较运算符

运算符	说明
=	等于
<>	不等
<	小于
>	大于
<=	小于等于
>=	大于等于

表1-5 字符串运算符

运算符	说明
&	连接
Like	模式匹配

Date 类型用于表示日期和时间值。在指定日期和时间数据时需要用#字号（#）将数值包围起来。

```
Dim x As Date
x = #8/23/1970 4:34:12 AM#
x = #4/22/1972#
x = #3:41:00 PM#
```

枚举（enumerated）数值类型也属于基本数值类型。枚举数值类型是为特定数值分配了名字的 Byte、Short、Integer 或 Long 类型。没有明确声明基础类型的枚举默认为 Integer 类型。比如，下面定义了一个称为 Color 的枚举 Integer 类型，把名字 Red 赋给值 1，名字 Blue 赋给值 2，名字 Green 赋给值 3。此外还定义了以一个名为 SwitchValue 的枚举 Byte 类型，把名字 Off 赋给值 0，On 赋给值 1。

```
Enum Color
  Red = 1
  Blue = 2
  Green = 3
End Enum

Enum SwitchValue As Byte
  Off
  [On]
End Enum
```

一旦定义了一个枚举，就可以用这个枚举值代替数字。

```
Dim Background As Color
```

```
Background = Color.Red

If Background = Color.Green Then
  Console.WriteLine("Background is green.")
End If
```

转换运算符能够在兼容的类型之间转换数据。每种基本类型都定义了转换运算符（参见表1-6），通用转换运算符 CType 可用于任何两种类型间的转换。

表 1-6 转 换 运 算 符

运算符	说明
CBool(x)	Boolean 转换
CByte(x)	Byte 转换
CShort(x)	Short 转换
CInt(x)	Integer 转换
CLng(x)	Long 转换
CSng(x)	Single 转换
CDbl(x)	Double 转换
CDec(x)	Decimal 转换
CChar(x)	Char 转换
CStr(x)	String 转换
CDate(x)	Date 转换
CObj(x)	Object 转换
CType(x,Type)	通用转换

默认情况下，兼容类型的转换不需要使用转换运算符，编译器会在运行时隐式地转换。但是，一些转换可能在运行时失败——比如，如果 Long 值超出了 Integer 的范围，把 Long 值转换成 Integer 就会失败。如果在文件开头指定 Option Strict 语句，则要求明确地声明可能失败的转换。

1.3 数组

数组类型包含一组相同类型的变量。数组中的每个元素都可以依据其在数组中的位置（称为下标，如图 1-1 所示）被访问，数组的下标总是从 0 开始。

下面的例子创建了包含 10 个 Integer 变量的数组，并用数字 1…10 填充该数组。

```
Dim x(9) As Integer

For i As Integer = 0 To 9
```

```
  x(i) = i + 1
Next i
```

数组可以是多维的，也就是说，可以按照多个维检索（如图 1-2 所示）。

下面的例子创建了包含 100 个 Integer 变量的二维数组，并使用数字 1～100 填充该数组。

```
Dim x(9, 9)As Integer

For i As Integer = 0 To 9
  For j As Integer = 0 To 9
    x(i, j) =  (i * 10) + (j + 1)
  Next j
Next i
```

可以在声明的同时指定数组变量预设的大小。大小总是用每个维的上界指定，上例中的声明 `x(9,9) As Integer` 即声明了有 100 个元素的数组，其索引范围为 0～9。

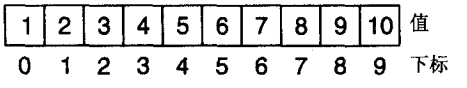

图 1-1 一维数组

下标	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	11	12	13	14	15	16	17	18	19	20
2	21	22	23	24	25	26	27	28	29	30
3	31	32	33	34	35	36	37	38	39	40
4	41	42	43	44	45	46	47	48	49	50
5	51	52	53	54	55	56	57	58	59	60
6	61	62	63	64	65	66	67	68	69	70
7	71	72	73	74	75	76	77	78	79	80
8	81	82	83	84	85	86	87	88	89	90
9	91	92	93	94	95	96	97	98	99	100

图 1-2 多维数组

省略边界的数组变量不会被初始化成任何大小，必须在对其使用前创建。数组可以使用 ReDim 语句初始化。

```
Dim x(,) As Integer

ReDim x(9, 9)
```