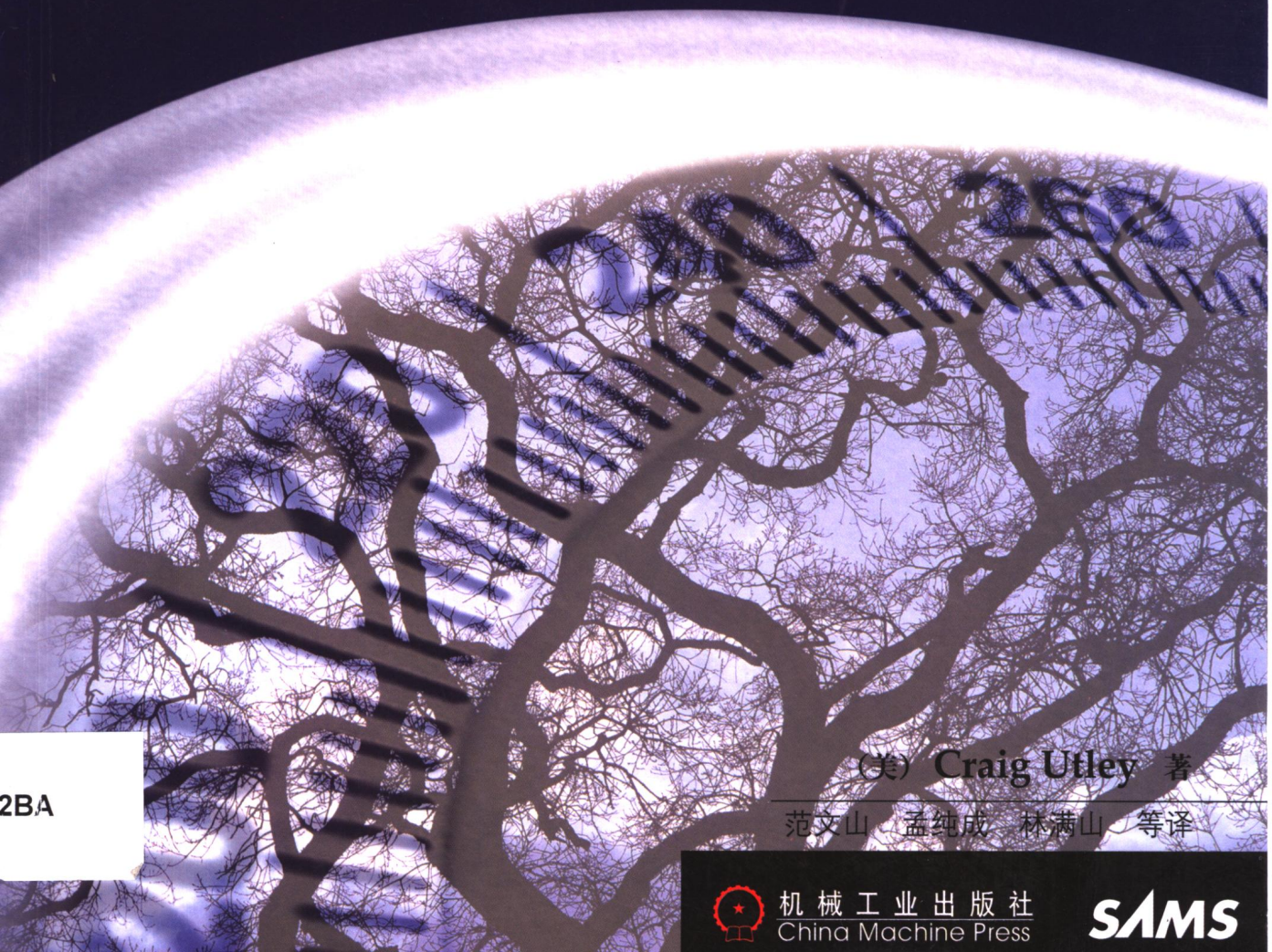
HZ BOOKS

A Programmer's Introduction to Visual Basic.NET

开发人员专业技术丛书

# Visual Basic .NET 编程指南





(美) Craig Utley 著

范文山 孟纯成 林满山 等译

机械工业出版社
China Machine Press

SAMS

开发人员专业技术丛书

# Visual Basic.NET 编程指南

(美) Craig Utley 著

范文山 孟纯成 林满山 等译

机械工业出版社
China Machine Press

本书深入浅出地介绍了Visual Basic .NET的新特性、使用和编程方法。主要内容包括：VB .NET的主要变化、使用VB .NET构建类和部件、VB .NET中的继承、ADO .NET、使用VB .NET和ASP .NET构建Web应用程序、使用VB .NET构建Web服务、使用VB .NET构建Windows服务和控制台应用程序、创建多线程VB .NET应用程序、使用VB .NET监视性能、部署和配置、.NET与COM的互操作性等。

本书通俗易懂、提供了大量中文屏幕图帮助理解。本书适合熟悉Visual Basic的开发人员阅读，可帮助读者快速从Visual Basic转到Visual Basic .NET的开发模式上来。



**本书版权登记号：图字：00-2002-2186**

**图书在版编目(CIP)数据**

Visual Basic.NET编程指南 /（美）奥特雷（Utley, C.）著；范文山等译 . -北京：机械工业出版社，2002.9

（开发人员专业技术丛书）

书名原文：A Programmer's Introduction to Visual Basic.NET

ISBN 7-111-10534-6

I. V… II. ①奥… ②范… III. BASIC语言-程序设计 IV. TP312

中国版本图书馆CIP数据核字（2002）第046246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：傅任诤 张鸿斌

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2002年9月第1版第1次印刷

787 mm × 1092 mm 1/16 · 16.25印张

印 数：0001-4000册

定 价：28.00元

# 前　言

## 为什么写本书

本书的意义在于对从Visual Basic转换到Visual Basic .NET（VB .NET）过程中出现的变化给出及时准确的介绍。书中的大部分内容都假设你已经很熟悉Visual Basic 6.0，因此，介绍VB6与新的VB .NET的主要不同是全书的重点。

从1.0版本开始，我就一直在使用Visual Basic。它最显著的变化发生在VB3到VB4的发展过程中，那时，引进了类模块，从此，VB开始了它长期而缓慢的面向对象之旅。我们第一次可以在VB中构建COM组件，从而导致了n层应用程序开发的蓬勃发展。VB4把COM开发能力带给了普通程序员，它不再是仅为少数的C++开发者所掌握的技术。

在我第一次思考VB6与VB .NET之间的不同之处时，我意识到这些改变甚至比从VB3到VB4所发生的变化要显著得多。我认为用一本书集中介绍这些变化以帮助VB6开发者转换到VB.NET上来是一件好事。为了这个目标，我把出版一本书名类似为《Migrating from VB to VB.NET》的想法推销给了两家不同的公司。Sams Publishing喜欢这个想法，一天他们打电话给我，让我在3周内提供书稿的缩略版。最终完成了一本更完整的书，就是你手中的这本书。本书讲述了更详细的内容，随着更新的、更稳定的Visual Studio .NET的推出，它将不断更新。

毫无疑问：VB .NET对所有人都是一个惊喜。有如此多的新内容需要学习，以至于一开始它令人望而生畏。但是.NET Framework的优势是显著的，最终它能大大地减少今天构建一个企业级分布式应用程序所需的努力。

## 为什么是VB .NET而不是C#

许多出版社把目标瞄准在Microsoft新创建的语言：C#（读做“C-Sharp”）上。这是一种基于C/C++的全新的语言。与VB .NET一样，C#是专门为.NET Framework构建的，而且已经有很多关于它的资料介绍。由于C#的大肆宣传，一些人可能会奇怪为什么我们选择VB .NET而不是C#。

尽管VB .NET和C#项目都是在Visual Studio .NET环境下创建的，但VB .NET是专门为VB开发者创建的，许多独一无二的属性使它成为构建.NET应用程序的非常好的选择。VB .NET还是VS .NET中包含有后台编译的惟一一种语言，这一功能意味着在键入代码的同时，它能够立即显示出其中的错误。VB .NET是支持晚绑定的惟一的.NET语言。在VS .NET IDE中，VB .NET在代码窗口的顶部提供了包括所有对象和事件的下拉列表；任何其他语言的IDE都不支持这一功能。VB .NET在为可选参数提供默认值、为开发者提供可用的控件集合等方面上也是独特的。另外，C#与它的兄弟C和C++一样，是区分大小写的，这一属性可能会使大部分习惯于用VB的开发者头疼。此外，C#对等于（=）和比较（==）操作符使用不同的符号。最后，有这样一个事实：

如果你了解VB，那么你就更贴近VB .NET的方式而不是C#。尽管许多特性变化了，但VB .NET的基本语法还是与VB类似，因此实际上如果你熟悉VB，你现在就已经知道了如何声明变量、建立循环等等。

如你所知，与其他.NET 语言相比，VB .NET具有某些优势。如果有兴趣想了解VB .NET优于传统VB的地方，那么你应该阅读本书。

## 谁应该读本书

本书面向当前的VB 开发者。如果不了解VB，那么本书的部分内容可能对你没有什么帮助。本书的目标是介绍哪些地方发生了变化。因此，如果有些特性没有变化，我将假设你已经了解它们了。如果你了解VB，并希望学习VB .NET，或者最少希望知道它能为我们做什么工作，那么本书适合于你。

如果你现在正在使用Visual InterDev创建Web应用程序，那么本书同样适用于你，因为Visual InterDev已经被集成到了Visual Studio .NET中。这意味着你可以使用VB .NET（以及C#）创建类似于Visual InterDev的Web应用程序。使用这种全新的方式，可以获得几点好处，包括编写完全的VB .NET而不是VBScript的能力。而且，与ASP模型相比，.NET 的Web应用程序体系结构（ASP .NET）优势是明显的。

无论使用VB .NET做什么，开始入手的地方都是.NET Framework。不理解.NET Framework就不能够编写好的VB .NET应用程序（包括Windows的或Web应用程序）。因此，我们将从.NET Framework的介绍开始。

# 译 者 序

随着分布式应用技术的日臻完善、电子商务的流行以及网络人口的激增，今天网络上的社会活动需要更为复杂的软件系统。微软大力推广的下一代平台技术.NET正是这种力量的代表。它给我们提供了一个未来可操作的社会活动平台。正像微软CEO鲍尔默所说，.NET代表了一个集合、一个环境、一个可以作为平台支持下一代Internet的可编程结构。

.NET技术特征包括四个特点：软件服务化、基于XML、融合多种设备和平台、新一代的人机界面。

Visual Basic .NET是微软公司推出的支持.NET 架构的新一代产品，其设计目的是为了快速而简单地开发包括Web服务和ASP .NET Web应用程序在内的.NET框架应用程序。使用Visual Basic .NET，能够在代码中方便地创建自己的组件或配置.NET框架类中的复杂组件。使用可视化组件能够在自己的应用程序中方便地使用消息对象、事件日志、性能计数等资源。

Visual Basic .NET中引入了很多更新的和改进的语言性能，例如继承、接口和重载，这使其成为一种强大的面向对象的编程语言。其他新的语言特性还包括自由线程和结构化异常处理等。Visual Basic .NET能与.NET框架及通用语言运行环境完全整合，因而能够提供多语言互用、冗码收集、增强的安全性和改进的版本支持。

全书通俗易懂、易于理解掌握，对Visual Basic .NET做了全方位的介绍。本书的读者群体定位在熟悉Visual Basic的开发人员，帮助他们快速而方便地从Visual Basic转到Visual Basic .NET的开发模式上来。本书并非只是知识点的简单罗列，而是通过对比、实例，向读者全面讲解Visual Basic .NET的语言知识点和编程技巧。

全书由范文山、孟纯成、林满山、周靖、肖林等翻译完成。尤晓东、文开棋对全书进行了审校，周迎春、徐海琳、王秀敏参加了很多必要的辅助工作，还有许多朋友同事给予了无私的支持，在此一并感谢。正是来自各方面的力量，使我们比较顺利地完成了翻译工作。同时，我们也很珍惜这次机会，能够为读者推荐这部作品是我们的荣幸。如果我们的工作能够给你带来些许助益，那是我们最大的幸福。

当然由于自身水平有限，不足和纰漏之处在所难免，恳请广大读者批评指正。

2002年2月

# 目　录

# 第1章　为什么要转移到Visual Basic .NET

本章内容包括:

- Visual Basic .NET: 一种全新的架构
- 公共语言运行环境
- 受管理执行
- 公共类型系统
- .NET Framework类库
- 自描述组件
- 跨语言互操作性
- 安全

Visual Basic .NET（VB .NET）从一出现开始，就有不少人在抱怨。主要原因是基于这样一个事实：他们已经习惯了使用当前的Visual Basic，但现在又不得不再去学习一门新的语言。的确，在这两者之间存在着明显的差异，而且现在的VB开发者在VB6.0上已经浸润了3年多。然而，现实中的确存在着需要我们转换到使用VB .NET的客观需求。尽管在使用Visual Basic .NET的起始阶段，学习曲线可能会起伏不定，但是随着对某些新概念的理解的深入，将帮助我们改变这种局面。这些概念中主要的一个就是.NET Framework。

## 1.1 Visual Basic .NET：一种全新的架构

现在最经常被问到的一个问题是：我为什么要转移到.NET上来？这个问题的产生是因为.NET刚出现不久，有关它能做些什么还存有不少疑问。从Visual Basic的观点来看，首先理解转移到Visual Basic .NET能带给我们哪些引人注目的好处是最重要的。

许多人已经接触到了VB .NET，但他们对它里面出现的变化一直存在抱怨。一本专业出版物里曾发表过一篇文章，几位长期从事Visual Basic的开发者就在文章中抱怨VB .NET中变化太多，以至于他们感到无所适从。的确，这种语言有不少显著的变化，如：新的错误处理结构（可选的）、名字空间、真正的继承、真正的多线程，以及其他的许多方面。有些人把这些改变只看作是Microsoft采取的一种手段，借此Microsoft就能在某个属性前打个对勾，然后说“yeah，我们做到了这一点”。

尽管如此，大部分开发者还是认为VB .NET中的变化有其充分的理由。过去的几年里，应用程序世界一直处在变化中，演变一直在持续地发生着。如果能够回退到20世纪90年代初，我们给使用Visual Basic1.0的开发者看一个n层应用程序，该应用程序包括一个ASP（Active Server Page）前端、一个用Microsoft Transaction Server实现的VB COM组件中间层以及一个由存储过程组成的SQL Server后端，那么毫无疑问，这个应用程序对当时的开发者来讲无疑是异类的。然而，过去的数年里，大量的开发者一直在用Visual Basic创建COM组件，同时在数据库访问领

域，他们已经非常精通ADO了。复用（reusability）和集中化（centralization，一种避免把组件分布到桌面上的方式）的需要在推动着演变向n层模型方向发展。

n层系统演变的下一步自然而然的是具有Web功能的应用程序。不幸地是，朝Web方向发展的过程中遇到了几个问题。可扩展性（Scalability）是一个问题，但更复杂的应用程序还有其他的要求，如跨多组件、跨多数据库或者两者兼具的事务（transaction）。为了解决这些问题，Microsoft创建了Microsoft 事务服务器（Microsoft Transaction Server，MTS）和COM＋组件服务（COM+ Component Service）。MTS（使用在Windows NT4系统上）和Component Service（MTS在Windows 2000系统上的升级）起着对象宿主环境的作用，使我们可以通过相对容易的方式获得可扩展性和分布的事务。然而，VB组件不能充分利用Component Service提供的全部好处，如对象池，这是因为VB不支持真正的多线程。

### 1.1.1 从Web开始

在ASP/VB6模型中，Microsoft使开发者构建一个组件，然后通过ASP调用它。Microsoft意识到使组件在HTTP上可直接调用，以便世界任何一个角落的应用程序能使用该组件是一个不错的想法。因此在简单对象访问协议（Simple Object Access Protocol，SOAP）之后推出了对此的支持。这样，开发者就能够在HTTP之上通过XML串调用组件，然后通过HTTP以XML串的形式返回数据。通过SOAP呈现出的组件支持URL，使得它们能像其他Web条目一样容易地被访问到。SOAP的一大优势是它已经成为了跨行业的标准，而不仅是Microsoft一家公司的产物。这已经赢得了许多厂商对Web 服务（Web Serviec）的赞誉，其中也包括Microsoft的主要竞争对手。

到这里，人们可能会产生一种错觉，认为SOAP就是我们需要的全部，从而我们能够继续坚持使用VB6。因此，理解VB .NET能带给我们什么，以及为什么它对我们及其他的开发者有帮助，这对升级到.NET是特别重要的。例如，我们要创建组件，希望它们通过SOAP是可调用的，但是怎么能够让别人知道这些组件的存在呢？.NET包含一种发现机制，允许他人找到可用的组件。在第9章“用VB .NET构建Web Service”中，我们将会了解这一机制的更多内容，包括所谓的“disco”文件。.NET还提供了其他许多属性，如用于资源释放的垃圾回收、首次实现的真正继承、跨语言和对在运行应用程序的调试，以及创建Window服务和控制台应用程序的能力等。

### 1.1.2 .NET是什么

在继续我们的讨论之前，多理解一点.NET的含义很重要。现在有很多.NET，如VB .NET：一种新版的Visual Basic；Visual Studio .NET（一种包含了VB .NET的集成开发环境）；C#.NET（读做“C sharp”）和C++.NET。Microsoft甚至把它的大部分服务器产品也称为“.NET服务器”。所有这些都是由.NET Framework、它的核心执行引擎以及公共语言运行环境（Common Language Runtime）组成。

在.NET模型中，我们编写针对.NET Framework的应用程序，这使得应用程序能自动得到以下这些好处，包括垃圾回收（它破坏对象，然后回收内存）、调试、安全服务、继承以及更多的功能。当编译任何一种支持.NET Framework语言的代码时，实际都是把它们编译成称为MSIL的东西，MSIL即Microsoft中间语言（Microsoft Intermediate Language）。这种MSIL文件是二进制

的，但它不是机器码，相反，它是一种平台独立的格式，能被放置到任何运行有.NET Framework的机器上。在.NET Framework内是一种被称为Just-In-Time，也称为JIT的编译器。它负责把MSIL编译成特定于硬件和操作系统的机器码。

在看过了基本变化后，需要理解的重要一点是为什么多年来Visual Basic开发者首要的需求一直是继承。从VB4以后，VB已经具有了接口继承（interface inheritance）的能力，但开发者需要实（real）继承，或称为实现继承（implementation inheritance）。原因是什么？好处又是什么？我们知道继承的主要优点是具有能够更快速地创建应用程序的能力。这是组件设计与复用性承诺的一种扩展。使用实现继承，我们可以构建一个基类，然后把它作为新类的基础，开始进行继承。例如，我们可以创建一个Vehicle类，它提供了能同时被Bicycle和Car类继承的基本功能。这里面的要点是Bicycle和Car从Vehicle类继承了功能，即实际代码。在VB4及更高版本里，我们能做到的最出色的工作是继承结构，它去除了任何实现代码。而在VB .NET里，基类里的功能对其他的类也是可用的，即必要时我们可以扩展或修改它们。

.NET为我们提供了集成的调试工具。调试过具有VB COM组件的ASP应用程序的人都知道，必须使用Visual InterDev来调试ASP，用VB调试组件。如果应用程序中还混合了C++组件，则不得不对这些组件使用C++调试器。而对于.NET，则只需要一个调试器。任何针对.NET Framework的语言都能被这个惟一的调试器进行调试，即使一个应用程序是一部分由VB .NET编写，而它又调用了由C#或者任何针对.NET Framework的其他语言编写的另一部分。

.NET 支持标准的安全机制，适用于应用程序的所有部分。.NET为所谓的DLL Hell提供了一种可能的解决方案，并省去了许多处理COM和注册表的复杂工作。.NET允许我们在本地运行组件，而不需要发起调用的应用程序到注册表中查找这些组件。

还有许多其他方面，VB .NET能为我们达成目前的VB无法实现的功能。例如，一种新的项目形式的出现：Web Application。现在Visual InterDev和它的解释性VBScript代码已经过时了。取而代之，现在我们能够用VB .NET（或C# .NET以及C++ .NET）构建自己的ASP.NET页面，为了得到更好的性能，它们被真正地编译过了。VB .NET还通过提供Windows Services 项目类型，第一次让我们可以自然地创建Windows 服务。同样，VB .NET首次允许VB开发者构建真正的多线程组件和应用程序。

需要知道的最后一点是，尽管最终的名字还没确定，但新语言将使用版本号。它很可能被称为VB .NET 2002。这暗示在某个时间，像有新的VB版本一样，VB .NET也将具有新的版本。本书中，涉及到从前的VB版本时，将使用VB或VB6，涉及到VB .NET 2002时，将简单地使用VB .NET。

现在，你已经决定了有必要从VB6转移到VB .NET，并且准备阅读本书，以了解其中的变化之处。然而，你看到的第一部分内容却是有关.NET Framework的一章。为什么用.NET Framework做为起始呢？原因在于，在没有理解.NET Framework之前，是无法理解VB .NET的。我们将看到，.NET Framework与VB .NET是紧密交织在一起的，构建到应用程序里的许多服务实际是由.NET Framework提供的，之后它们简单地被我们的应用程序调用。

.NET Framework是一个服务和类的集合。它作为一个层存在于我们编写的应用程序与底层操作系统之间。它是一个功能强大的概念：.NET Framework不是一种仅局限于Windows的解决

方案。它能被移植到任何操作系统上，即.NET 应用程序能运行于任何一种驻有.NET Framework的操作系统上。这意味着如果.NET Framework对其他平台是可用的，那么通过简单地创建VB .NET应用程序，我们即可获得真正的跨平台能力。但是，尽管这种跨平台承诺是.NET的一个主打卖点，目前却还没有关于.NET被移植到其他操作系统上正式的报告。

此外，.NET Framework令人激动之处还因为它封装了过去被构建到各种编程语言中去的许多基本功能。.NET Framework具有能使Window Form 工作的代码，因此任何语言都能使用这些内建代码来创建和使用标准Windows表单。此外，因为Web Form是Framework的一部分，因此任何.NET语言都能被用来创建Web Application。再进一步，这意味着各种程序设计元素在所有语言间将是相同的；Long数据类型在所有.NET语言中具有相同的尺寸。对于串和数组，这一点尤为重要。在把串传递到由另一种语言编写的组件中之前，我们将不再担心串是BStr的还是CStr的，对那些需要调用基于C的Windows API的VB开发者而言，这曾经是一个普遍关心的问题。

## 1.2 公共语言运行环境

.NET Framework的主要组成部分之一是公共语言运行环境（Common Language Runtime, CLR）。CLR为开发者提供了许多好处，如异常处理、安全、调试、版本管理等。并且这些好处对任何为CLR构建的语言都是可用的。这意味着CLR上能驻有多种语言，并能提供跨语言的公共工具集。Microsoft已经为它推出了VB.NET、C++.NET以及首选的C#语言，意味着这三种语言完全支持CLR。此外，其他厂商已经承诺提供其他语言的实现，如Perl、Python甚至COBOL。

当编译器对CLR编译时，生成的代码称为受管理代码（managed code）。受管理代码是一种简单代码，它利用了CLR提供的服务的好处。为了使运行环境能与这种代码一起工作，这种受管理代码必须包含元数据（metadata）。元数据在编译过程中由面向CLR的编译器生成。元数据与编译后的代码存储在一起，它里面含有关于代码里的类型、成员和引用等信息。CLR用这种元数据来：

- 定位类
- 载入类
- 生成本地代码
- 提供安全管理

如果你仔细考虑回想一下，就会知道上面的这些任务过去是由COM和注册表来处理的。.NET的目标之一是不需要注册表的使用就能够分布应用程序。事实上，.NET组件能被拷贝到一个目录中后即被使用，而不需要注册过程。.NET处理了从组件中定位和载入对象的工作，取代了过去由COM做的工作。

运行环境还处理对象的生存期。与COM/COM+为对象提供了引用计数一样，CLR管理着对对象的引用并在所有引用消失时把对象移出内存，这一过程被称为垃圾回收（garbage collection）。与VB相比，尽管垃圾回收在实际上轻微地削弱了使用者的控制能力，但却使我们获得了某些重要的好处。例如，由于.NET对那些仅包含在循环引用里的对象提供了处理逻辑，因此由于循环引用而遗留下来的对象数目会减少甚至完全消失，从而降低了错误率。此外，与VB中破坏对象的老方式相比，垃圾回收变得更加快速。那些由我们创建而由运行环境管理的对象实例称为受

管理数据（managed data）。尽管受管理数据为我们提供了运行环境的所有好处，但我们可以在同一个应用程序里与受管理数据和非受管理数据同时交互。

CLR还定义了一个标准类型系统，该系统被所有的CLR语言使用。这意味着所有的CLR语言都将具有相同长度的整数和长整数，它们也将具有相同类型的串——我们不必再担心BStrs和CStrs的问题了。这一标准类型系统为某些功能强大的语言的互操作性打开了一扇门。例如，即使组件是用不同语言编写的，我们也能够从一个组件向另一个组件传递类的引用。我们还能够在C#里继承一个由VB .NET编写的基类。同理，任何面向该运行环境的语言组合均可。请记住，COM也有一组标准类型集，但它们是二进制标准的。这意味着使用COM，我们会在运行期间具有语言的 互操作性，使用.NET的类型标准，我们则在设计期间具有语言互操作性。

在编译后，受管理代码中包含了元数据，其中含有关于组件自身的信息以及那些用来创建代码的组件。运行环境能进行检查以确定我们依赖的资源是否可用。元数据取消了需要把组件信息存储到注册表里的要求。这意味着把组件转移到新机器上不需要注册过程（除非它是全局部件，我们会在第4章“使用VB .NET构建类和部件”对此做介绍），而且撤销组件可以像删除它们一样简单。

正如你所看到的，公共语言运行环境提供了许多好处，它们不仅是全新的，而且将会增进我们构建应用程序的经验。我们将详细介绍其他的优点，其中还包括VB .NET中的一些新的面向对象属性。这些新属性中的许多只是简单地呈现给了VB .NET，与其说是对语言的增加还不如说是对运行环境的增强。

## 1.3 受管理执行

为了理解VB .NET应用程序工作方式，以及有多少代码不同于Dorothy在Kansas编写的VB代码，理解受管理代码以及它是如何工作的是非常重要的。为了使用受管理的执行以及获得CLR的好处，我们必须使用一种为运行环境构建或者说是针对运行环境的语言。幸运的是，这其中包括有VB .NET。事实上，Microsoft希望确定VB .NET成为.NET平台上的首选语言，这意味Visual Basic将不再被指责成是一种“玩具”语言。

运行环境是一种语言中立的环境，这意味着任何厂商都能创建一门利用运行环境属性优点的语言。不同的编译器可能对开发者呈现出不同数量的运行环境，因此我们所使用的工具和用于编程的语言工作起来可能会有所不同。当然，每一种语言的语法是不同的，但当编译过程发生后，所有的代码都应该能被编译成运行环境可理解的形式。

**注意** 一门针对运行环境的语言不意味着在该语言中不能增加进别的语言无法理解的任何属性。但为了确保自己的组件能够被由其他语言编写的组件使用，我们必须只使用那些被公共语言规范指定的类型。在附录A“公共语言规范”中能够看到被这些公共语言规范指定的元素。

### 1.3.1 Microsoft 中间语言

.NET非常有趣的一个方面是在编译代码时，并没有把代码编译成本地码。对此，VB开发者

可能会感到惶恐不安，以为又回到了解释码（interpreted code）的时代，事实上编译过程把我们的代码翻译成了一种称为Microsoft中间语言（Microsoft Intermediate Language）的东西，被称为MSIL，或简单地称为IL。编译器还创建了必要的元数据并把它编译到组件中。由此产生的效果是IL将独立于CPU。

当IL和元数据被放到文件中，这个编译后的文件被称为PE，即可移植的可执行文件（portable executable），或自然的可执行文件（physical executable），其提法因人而异。因为PE包含了IL和元数据，因此它是自描述的，这取消了对类型库的需要，也取消了由接口定义语言（Interface Definition Language，IDL）指定接口的需要。元数据是如此的完美以至于任何.NET语言都能够从该PE文件中的类进行继承。记住：.NET之所以被称为是设计期间兼容的（compatible），原因正在于此。当用VB .NET开发时，我们可以从由C#创建的类进行继承，同时我们也具有对IntelliSense和其他Visual Studio .NET属性的完全访问能力。

### 1.3.2 及时编译器

然而，我们的代码不会以IL的形式保持很久。因为IL是平台独立的，所以包含IL的PE文件可以与运行在.NET Framework 上的CLR一起分布和放置到.NET Framework存在的任一操作系统上。但是在运行IL时，它被编译成了针对该平台的本地码。因此，我们还是在运行本地码，我们绝对不是回到了解释码的时代。对本地码的编译是通过.NET Framework的另一个工具：及时编译器（Just-In-Time compiler，JIT）实现的。

代码被编译后，它能在Framework内部运行，并利用底层属性的好处，如内存管理和安全。编译后的代码对运行.NET Framework 的CPU而言是本地码，这意味着我们确实在运行本地码而不是解释码。JIT编译器对每一个运行有.NET Framework的平台都是可用的，因此我们总是能得到针对运行了.NET Framework的任何平台的本地码。记住：尽管目前还仅限于Windows平台上，但这种局面会在将来发生变化。

> **注意** 调用操作系统特定的API仍然是可能的，当然这会把应用程序限定在那个平台上。这意味着调用Windows API还是可能的，但那样代码将不能运行在非Windows机器上的.NET Framework中。目前，.NET Framework还仅存在于Windows平台上，但这种情况可能会在未来发生变化。

### 1.3.3 执行代码

有趣的一点是，在组件首次被调用时，JIT编译器并没有编译全部的IL。相反，它只编译了首次被调用的每一个程序。这样做能够避免去编译那些从来不会被调用的代码。在Beta2版中，这部分被编译过的代码被存储到了内存中。但是，Microsoft还提供了一个PreJIT编译器，它将一次性编译所有的代码，并把编译完成的版本保存到磁盘上，因此编译结果会随时间而持久保持。这个工具名为ngen.exe，能被用来预编译全部的IL。如果CLR无法找到一个预编译的代码版本，它会启动JIT来编译。

在代码开始执行后，就能充分利用CLR的优点了，如安全模型、内存管理、调试支持和分

析工具（profiling tool）。书中将会对这些优点中的大部分进行介绍。

### 1.3.4 部件

VB .NET创建的一种新的结构是部件（Assembly）。部件是由一个或多个物理文件组成的集合。这些文件大部分是代码，如我们构建的类，但它们也可以是图像、资源文件或与代码相联的其他二进制文件。这样的部件被称为静态部件，因为我们可以创建它们并把它们保存到磁盘上。动态部件在运行期间被创建并且通常不能被存储到磁盘上（尽管可以做到）。

部件代表了部署单元、版本管理、重用和安全。这听起来像在过去6年里我们在用Visual Basic一直创建着的DLL，它们之间确实相似。同标准的COM DLL有一个类型库一样，部件具有一个清单，其中包含了部件的元数据，如类、类型以及包含在IL之中的引用。和COM DLL一样，部件通常包含有一个或多个类。在.NET中，应用程序用部件来构建，但部件自身不是应用程序。

对部件来说，最有意义之处可能在于：所有运行期间的应用程序一定由一个或多个部件构成。

1. 部件清单

理论上清单（manifest）类似于COM DLL里的类型库。清单中包含有部件里面条目的所有相关信息，其中包括部件的什么部分被呈现给外部世界。清单还列出了该部件对其他部件的依赖性。每一个部件都要求具有一个清单。

清单可以是PE文件的一部分，但如果部件内具有一个以上的文件，则它可以是单独的文件。尽管下面不是一个详尽的列表，但清单里都应该包括：

- 部件的名字
- 版本
- 部件内的文件
- 被引用的部件

此外，开发者可以对部件设定自定义属性，如标题和描述。

2. DLL恶梦的终结

COM最大好处之一是它被看作是DLL 恶梦的终结。如果回顾一下16位编程的日子，我们一定还记得，那时不得不随Windows应用程序分发许多DLL。好像差不多每一个应用程序都必须要安装几个相同的DLL，如Ctrl3d2.dll。我们安装的每一个应用程序可能都带一个略有差异的DLL版本，最终结果是我们使用了同一DLL的多个拷贝，但其中许多是不同的版本。更糟糕的是，某一个DLL的版本被放到Windows\System目录后，可能会破坏许多已有应用程序的使用。

COM被认为解决了所有这些问题。应用程序不再需要先查找自己的目录再去查找Windows路径来得到DLL。使用COM时，对组件的请求被发送到了注册表。尽管机器上可能有同一个COM DLL的多个版本，但任何时刻在注册表里只能出现一个版本。因此，所有的客户将使用相同的版本。然而，这也意味着DLL的每一个新版本必须要保证与以前的版本兼容。这导致了在COM中，接口是不可变的。组件被生产出之后，接口被认为是永远不能改变的。在概念上，这听起来很了不起，但开发者可能并且确实会发布能破坏二进制兼容性的COM组件。换句话说，他们在组件中改变、增加或去除了属性和方法。然后这些被修改后的组件破坏了所有现存的客户。许多VB开发者都曾遭受过这类问题的折磨。