21 BEIJING HOPE

Designed for Microsoft Windows NT Windows 98

Microsoft® Programming Series

FOR ENTERPRISE DEVELOPERS

Microsoft Visual Studio 中文版系列图书
编程的利器·知识的进发

Programming Components with Microsoft Visual Basic 6.0 Second Edition

组件编程技术 第二版

- 使用配套光盘中提供的专家经验和可重用组件，能帮助您高效地创建可扩展的分布式企业解决方案，并节省编程时间
- 使用Visual Basic 和最新的COM(组件对象模型)设计技术，可达到与万维网和数据库工具的完美结合

本书配套光盘内容包括:
1. 本书的配套电子书
2. 本书相关程序的源代码及其它与组件编程有关的重要文档

[美] Guy Eddon, Henry Eddon 著
希望图书创作室 译

北京希望电脑公司
北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

Microsoft Press

美国微软出版社授权中文版系列书
开发人员的利器、知识的迸发
标准、权威、系统、完整的编程参考手册

Microsoft Visual Basic 6.0

组件编程技术

（第二版）

[美]Guy Eddon, Henry Eddon 著

希望图书创作室 译

本书配套光盘内容包括：

1. 本书的配套电子书
2. 本书中相关程序源代码及其他与组件编程有关的重要文档

北京希望电脑公司

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 提 要

本书是美国微软出版社授权中文版系列书之一，是专门为从事Microsoft visual Basic 6.0软件开发和应用的广大编程人员而编写的。全书包括三个部分：第一部分“组件开发概述”、第二部分“在Visual Basic中构造组件”和第三部分“利用数据库和Web技术”。书中的重点不是介绍面向对象编程技术的长处，而是说明它的局限性和不足，进而介绍组件编程的新技术，提高程序代码共享和重复使用的程度，减少程序员的重复劳动造成的浪费。

书中既有历史的回顾，也有现状的说明；既有概念性的介绍，也有大量的实例。其中很多实例是针对Internet环境设计的。本书适用于广大编程人员、系统分析员和VB爱好者，也是那些从事Web页面设计的人员、高等院校相关专业的师生自学、教学的重要参考书。

本书配套光盘内容包括：1. 本书的配套电子书；2. 本书中相关程序源代码及其他与组件编程有关的重要文档。

版 权 声 明

系 列 书：	美国微软出版社授权中文版系列书
书 名：	Microsoft Visual Basic 6.0组件编程技术
文本著作者：	[美] Guy Eddon，Henry Eddon 著　　希望图书创作室 译
责任编辑：	陈河南
CD制作者：	微软出版社　希望多媒体部
CD测试者：	希望多媒体测试部
出版、发行者：	北京希望电脑公司　　北京希望电子出版社
地 址：	北京海淀路82号，100080
	网址：www.bhp.com.cn
	E-mail：lwm@hope.com.cn
经 销：	电话：010-62562329,62541992,62637101,62637102 (图书发行,技术支持) 010-62633308,62633309 (多媒体发行，技术支持) 010-62613322-215 (门市)　010-62531267 (编辑部) 各地新华书店、软件连锁店
排 版：	希望图书输出中心
CD生产者：	文录激光科技有限公司
文本印刷者：	北京双青印刷厂
开本/规格：	787×1092毫米　16开本　23.375 印张　407 千字
版次/印次：	2000年1月第1版　2000年1月第1次印刷
印 数：	0001-5000册
本 版 号：	新出音管[1998]312号　ISBN 7-980026-30-6/G・04
定 价：	40.00元（1CD，含配套书）

译 者 序

本书回顾了面向对象编程技术的历史，介绍了这一技术的基本概念。但本书重点不在于介绍面向对象编程技术的长处，而在于说明它的局限性和不足。并在此基础上，介绍了一种新的、组件编程的概念。组件编程的出发点是提高程序代码共享和重复使用的程度，组件编程的目标是使编程工作变为一种“利用现成软件组件组装软件产品”的工作。这显然是广大编程人员梦昧以求的事情。

书中既有历史的回顾，也有现状的说明，既有概念性的介绍，也有大量的实例。其中很多实例是针对 Internet 环境设计的。这对于那些从事 Web 页面设计的人员尤其有参考价值。

书中的术语尽量采用已为 Microsoft 所认可的。但对少量仍需商榷的术语，我们也根据原文和所描述对象的实际含义，确定了我们自己认为最贴切的表述。

参与本书翻译的有黄建江、朱仰太、陈文泽、周景亭、曹达伟、马志力、黄娜、孟晔、高素英、程夏、黄新、毛小卫、廖江天、历复华、那小菲、施展超、方妍丽、李欣欣、林业莲。

尽管我们的确在煞费苦心追求尽善尽美，疏漏之处仍在所难免，恳请读者批评指正。

程夏执笔

1998.9

前　言

本书面向那些有兴趣学习利用 Microsoft Visual Basic 6 构造组件软件的读者 。我们竭尽全力使本书斟于完美，并希望你阅读它、欣赏它、从中最大程度地获益。通过本书，你将从 Visual Basic 的角度学习组件对象模型（Component Object Model，COM），并学会构造各种类型的基于 COM 的组件。在本书的大多数章节中，正文中穿插了大量的练习，你可通过这些 练习对正文中的概念进行实验。本书配套不光盘中有关于所有这些练习的答案。

撰写本书过程中，我们得到了很多人士的帮助。我们要感谢微软出版社的朋友们，他们当中有：Eric Stroo，他将撰写此书的计划项目交给我们实施，并总在适当的时候给我们宝贵的支持。还有 Alice Turner 和 Dail MageeJr.，他们为编辑此书花费了大量的精力。还要感谢 Eric Maffei，Joanne Steinhart，Joe Flanigen，Joshua Trupin，Laura Euler，以及 *Microsoft System Journal* 和 *Microsoft Interactive Developer* 的其他人士。

Guy 感谢 Learning Tree International 的各位人士，他们为我们创建了一个模拟工作环境。我们还要特别提到 David Collins、Eric Garen、John Moriarty、Francesco Zamboni、Mindy Robbins 和 David Barkley。Henry 感谢他在 UPS 的同事们在组件软件方面的独到见解。特别是 Laynglyn Capers、Sherri Berman、Dave Karlson 和 Rudy Vossen 都对 COM 提出了他们独特的观点。Laynglyn 确信 Interface Definition Language（IDL， 接口定义语言）是 COM 程序员的最佳朋友；Sherri 坚持认为类型库不够完美，因为它们没有包含进诸如 Size_of 之类的 IDL 属性；Dave 把 COM 视为一个更好的 C++；Rudy 认为联结点是解决编程难点的有力武器。感谢我们的家人和朋友，没有他们的支持和理解，这一规划的实施是不可能的。

Guy Eddon
Henry Eddon
guyeddon@msn.com

作 者 简 介

Guy Eddon

尽管 Guy Eddon 想成为世界闻名的大提琴演奏家的雄心不改，但他还是拿出了一个休假年（美国给予大学教授的特殊待遇——译者注）的时间涉足到软件开发这一奇妙世界中来了。他的第一个真正的工程是一个游戏“丹尼的房间”（Danny's Room），这是为他那大提琴教师的儿子创作的。1992 年，“丹尼的房间”从约翰·霍普金斯国立助残计算调查所（John Hopkins National Search for Computing to Assist Persons with Disabilities）获得一项奖励，并被国立公共广播电台的一个节目 All Things Considered（考虑周全）所采用。

Guy 的第一篇文章是关于 OS/2 的，发表在 Windows Developer' Journal。他也为 Microsoft System Journal、Microsoft Interactive Developer 和 IEEE Proceedings 撰文。Guy 的第一本书名字叫“RPC for NT”（NT 环境中的 RPC），于 1994 年由 R&D Publications 出版公司出版。此后，Guy 和 Henry 合作于 1997 年出版了《Active Visual Basic 5.0》，于 1998 年出版了《Inside Distributed COM》。两书均由 Microsoft 出版社出版。在其它时间里，Guy 喜欢驾驶小型飞机出游，酿制葡萄酒，参加潜水运动和进行证券交易。Guy 也在 Learning Tree International 讲授 Visual Basic、Java 和 Win32 编程课程。

Henry Eddon

Henry 在计算机领域的生涯可追朔到 Haifa 大学的 IBM 的 1132 系列的计算机时代。在那里，他创建了第一个计算机化的学生入学登记软件，那是用 Fortran IV 编制的。后来，他从哥伦比亚大学毕业，并获数学学位。此后，他从 Commodore SuperPET 转到 HERO 1 机器人工作，后来又到了最初的 IBM PC。1984 年，Henry 和一个眼科专家朋友开始了 AMOS 的研究。AMOS 是一个保险帐单处理和患者预约安排程序。该程序跳过 MS-DOS，直接访问视频内存，因而达到了很高的处理速度。Henry 从 National Institute for Automotive Service Excellence 取得了 Master Mechanic 许可证。他编制了一个 6800 Motorola 汇编器，使得可在 HERO 1 机器人上用汇编语言，而不用机器代码编程，Henry 因此从 Institute for Certification of Computing Professional（ICCP——计算专家认证学会）获得 Certified Computing Professionals（CCP——合格计算专家）资格。他受雇于 United Parcel Service 公司，喜欢 Dilbert 动画。

目　录

第一部分　组件开发概述

第二部分　在 Visual Basic 中构造组件

第三部分 利用数据库和 Web 技术

第一部分　组件开发概述

第一章　关于组件软件的基本知识

多年来，软件开发过程尚未发生太大的变化。大多数软件开发在解决系统资源开销和运行时间过长方面花费了很大精力，而得到的软件产品往往是隐患不断和难以维护。其间也涌现了大量的各种各样的概念和方法，从框图到面向对象，某些方法在缩短运行时间方面得到了承认，并被视为解决软件开发问题的灵丹秒药。然而，所有这些曾被寄予希望的解决方法都未达到预期目的。还从未找到一种可信赖的替代方法，能够确保多个个别人员和小组能成功地共同开发同一个项目。这并不是说框图于事无补，也不是说面向对象的方法漏洞百出，只是说它们没能把软件开发化解到能够确保软件产品获得成功的一种公式。目前，人们都认识到，软件开发天生困难，它所面临的诸多问题没有单一的解决方案。

软件开发人员和技术管理人员长时期来一直梦想有一天，软件能够用事先设计好的组件组装，就像用现有的电子组件在线路板上装配一样。由于Component Object Model（COM——组件对象模型）的出现，这一天已经到来。

基于组件的软件开发对于保证软件开发的协调性提供了很大方便。由于大多数软件共享公共的元素，因此，开发人员很容易利用经过彻底测试并业已证明有效的软件来组件组装应用程序。此外，开发人员可把针对某个特定应用程序的具体某项功能设计为一个软件组件，用以加速大型系统中的功能组装过程。

在本章，我们先探讨一下早于组件软件出现的面向对象的程序设计技术。然后我们集中精力讨论利用 Microsoft Visual Basic 6 构造组件软件的方法，用以说明它比面向对象的程序设计技术究竟有什么方便之处。

1.1　面向对象的程序设计技术——简单的历史回顾

面向对象的程序设计是近年来出现的一项技术，它已经风光了一段时期，并受到软件业界相当的好评。这种评价表现在面向对象的程序设计语言的广泛应用上。这样程序设计语言有 Ada、Smalltalk、Java 和 C++。这些编程语言都带有程度不同的面向对象的特征。面向对象的程度与设计人员的专业领域，他们所要解决的问题，以及他们所受到的客观限制都有关系。例如，C++采用一个面向对象的程序设计实用视图，这种视图很大程度上来源于

C，以及 C++与 C 的精确兼容性。Microsoft Visual Basic 不是充分面向对象的，但面向对象的思想在它的设计中得到了充分的体现，并且它继续朝着一个更完全的面向对象的语言进化。

1.1.1　什么是面向对象的程序设计

简言之，面向对象的程序设计技术的目标，在于编制能反映每个具体的人在现实世界中的思维模式的软件。这一点与过程性语言是相矛盾的。例如，在 C 这种过程性语言中，程序设计是以一种过程化的和逻辑上结构化方式，来模仿计算机的逻辑。面向对象的程序设计方法认识到，大多数软件试图将人们日常处理的事务（即对象）模型化。然而，面向对象的程序设计技术自己并不设法将这些对象化解为一组过程化的步骤，而是允许开发人员直接在程序代码中更容易、更有效地表达这些对象的存在。这样做的出发点在于使程序代码更富表现力，而又降低维护的系统资源开销。

大多数面向对象的程序设计语言都明确区分“类”（class）和“对象”（object）。类是一个模板，用于定义其自身的成员。对象则是类的一个“实例”（instance），对象完成实际操作。例如，比较旦糕切隔器和旦糕。旦糕切隔器就是类，它确定旦糕的各种属性，例如形状和大小。旦糕则是旦糕切隔器生成的一个实例，它就类似于一个对象。

面向对象的概念详细说明了面向对象的程序设计语言的三个主要特征，即：

- 封装　即隐藏对象的实现细节的能力。
- 继承性　在创建新的、更专门化的对象时，重复使用现有对象的能力。
- 多形性　依赖于所使用的具体对象，展现出多种行为的能力。

1.1.2　封装

在大多数标准的过程语言（例如 C）中，程序代码作为功能模块编写，而功能模块则对数据进行操作。利用封装功能，可将程序代码与相关数据进行组合来定义一个对象。这种程序代码与数据的组合称为“实现”（implementation）。“实现”对于所定义的对象来说是私有的，这意味着对象的内部实现是它自己的事，与任何其他人都没有关系。对象是某种意义上的一个黑匣子。

对象也展示一个公共接口，通过该接口可访问数据。该接口就构成了对象与其客户之间的契约。客户指应用程序，或需要该对象为之提供服务的其他对象。这里我们不打算讨论终端用户接口，只讨论使一个程序片断可与另一个程序片断进行信息交流的编程接口。这种封装技术的背后是这样一种思

想：它允许修改、完善和扩展一个对象的内部实现，条件是该对象所展示的公共接口不受损害。

封装技术有很多优越之处，但在软件性能方面有一点点损失，因为它增加了一个额外的公共接口层。为了提高效率，必要时，允许一个对象的某些用户跳过公共接口，而直接访问你的私人实现。这种情况下，需将这样一些特殊用户定义为“朋友”，因此可给予他们直接访问私人实现的特权。但如对这种特权使用不当，“朋友”这一概念也可能会破坏封装的思想。因此，仅在实属必要时才使用“朋友”这一概念。

1.1.3　继承性

代码重用是面向对象的程序设计技术的精髓。任何一个程序代码产品，在它被正式交付使用之前，都要经过无数次的调试和修改，所花费的时间和精力是巨大的。考虑到这种情况，你就会意识到代码重用实在是一种很诱人的前景。所谓继承性，即从另一个对象创建一个继承其功能特征的新对象。通过这种方式，继承性实现了在一个应用程序内的代码重用。如果一个已有的类具备了你所需要的基本功能，只是其行为方式不完全如你所设想，继承技术就特别有用。这种情况下，这一新的类就成为现有基础类的超集。它允许你重新使用基础类所具有的功能，与此同时又在导出类中添加了新的功能。为使用继承性，几乎总是要求开发人员具有访问源码的特权，以便他们能够将基础类成员函数的代码复制到导出类中的超越功能中，然后对它进行小规模的修改。

类库

从现有的类继承功能特征这种能力产生了类库这样一种概念。例如，与Microsoft Visual C++一起用来开发Microsoft Windows应用程序的Microsoft Foundation就是这样一种类库。这些类库提供了为创建一个标准的基于Windows的应用程序所需的大部分默认代码，使得你可创建继承那些默认功能的类。利用这种技术，很容易着手开始编写基于Windows的应用程序，因为大部分常用的繁琐工作已经为你完成了。为创建你自己的应用程序，你需要做的只是在基础类中添加你自己特别需要的代码，也许还要添加某些用于替换默认代码的超越代码。对于Windows应用程序的设计来说，实在是一种“加水搅拌即可”的方法。

多继承性

在某些面向对象的程序设计语言（例如C++）中，另一个可用的功能是“多继承性”（multiple inheritance）。利用多继承性，开发人员可以同时从

两个或多个基础类继承代码。例如，设想你有一个 TextEdit（文本编辑）类，它提供文本编辑功能。你还有一个 Window 类，它提供一个窗口系统。如果你想创建的类由一个位于某窗口的文本编辑器构成，你就可以使用多继承性。

理论上，这听起来很了不起，但实际上，事情并不总是那么尽如人意，总是那么乐观。假如很不幸，碰巧这个 TextEdit 类和 Window 类原本都是继承同一个另外的类（例如某个 Display--显示--类）而创建的，你现在就有两组相同的数据成员，它们在这个新的多继承类中会发生冲突。对这一问题，C++以虚拟基础类的方式提供了一个完满的解决办法。你很可能会问这样一个问题，对如此复杂的设计，可维护性如何保证（Java 程序设计语言为解决这样的问题采用的是绕道走的方法，即通过不支持多类继承性而避开这一问题。Java 为避开这一限制，它从接口的角度支持多继承性，这些接口不含数据成员，因此不会发生数据成员冲突）。

1.1.4　多形性

利用 C++提供的上述类型的继承性，在继承层次结构中的很多类都能实现同样的方法。各种方法都超越了它自己的基础类的实现。利用多形性，开发人员可为基础类编写相对通用的程序代码。关于这些基础类，可针对多种多样的具体情况，对它们的代码进行适应性修改。使得它们对那些与最初的设想不相符合的情况也能够适用。这就使得这些基础类能够用于更专门化的导出对象。假定你正在编写一个“交通工具”（Vehicle）类，它包含一个“前往”（Go）方法。在这一工程的有效期间内的后期某一时刻，你要添加一个“飞机”（Airplane）类和“小汽车”（Car）类，这两个类都要从“交通工具”基础类继承属性特征。在“飞机”类中，你要利用一个函数来超越“前往”方法，该函数的功能是使飞机飞往某地。而在“小汽车”类中，也要利用一个函数来超越“前往”方法，该函数的功能是使小汽车驶往某地。多形性要表达的意思是：如果你调用该“交通工具”类中某对象的“前往”方法，而该“交通工具”对象实际是一个飞机，则调用的是“飞机”对象的“前往”方法，也就是使飞机飞往某地。而如果调用的是“小汽车”对象的“前往”方法，则使小汽车驶往某地。

关于 Visual Basic，作为一个更理想的例子，我们考虑有一家包裹载运公司。一个普通包裹类大概有一个“载运”（Ship）方法，它知道怎样载运包裹。“次日航班”（NextDayAir）和其他包裹类型也有它们自己的“载运”方法。每一种包裹类型都需要有一种自己特别的“载运”方法。这些载运方法要考虑这样一些因素：包裹需要送达的时间、载运途径（空中或地面）、

特别警告信息（如保价包裹）等等。现在假定你正在编制一个供邮递用的应用程序，它要在该公司的邮递柜台运行。你想让该应用程序处理在柜台交办的任意类型的包裹，包括当前虽然还不存在，而将来可能出现的包裹类型。该应用程序了解普通包裹类的情况，但你想把所有类型的包裹同等对待，不想让这种普通类型与其他们类型有什么不同之处。那就不要让该应用程序知道任何导出包裹类型的细节。此外，当该应用程序激活任何一种已接办的包裹类型的方法时，该应用程序要使用该具体包裹类型的“载运”方法。

能处理多种形态的这种能力就是多形性。开发人员可针对基础的对象类型编写代码，但运行时，该程序代码实际上在处理一个从基础对象导出的对象。就好象向一个对象发送通用的消息，而该对象能够在其能力所及的环境中解释这条消息的含义。这种编写相对通用的程序代码，而能正确处理具体对象类型的能力就是多形性的一个明显的长处。

1.2 程序代码共享和重用

面向对象的编程技术之所以受到如此广泛的欢迎，很大程度上是因为它承诺允许开发人员在完全不同的工程（project）间共享程序代码。正如我们已经提到的，程序代码和算法的重复性开发一直不断地在进行着。这导致了大量的时间、精力和金钱的浪费。尽管一个完美实现的面向对象的设计方案的主要优点在于程序代码的共享和重用，但实际上，程序代码共享的百分比仍然很小。直到最近，即使在 Microsoft Office 这样的应用程序包中，各应用程序也在用不同的程序代码实现标准的共享功能，如工具栏、状态栏和拼写检查器。在 Windows 98 和 Microsoft Windows NT 中，很多这样的标准图形用户接口（GUI）控件已经建立在操作系统中了，因此使得所有应用程序可共享它们。考虑到这一点的话，操作系统实在是一个程序代码重用的伟大范例（如果不在乎它还不够灵活这一不足之处的话）。

程序代码重用似乎被各类人员视为一种必定会要发生的事情。这实际上已经成为人们的一种不切实际的奢望，因为程序代码重用需要进行详细的计划，这就意味着需要细致的思考。如果你编写了某个程序代码，把它交给了你的朋友使用。这是否就算程序代码重用呢？ 怎样将程序代码库链接到应用程序中呢？这可以看作是程序代码重用的例子，但它们各有自己的问题。如果你把你的程序代码交给了你的朋友，而他们不喜欢该程序代码的某些方面，他们就可能要深入到你的源码，并进行修改。这样做就与程序代码重用的思想相背离了。 修改别人的程序源码实在是一种费力不讨好的事情，大有越俎代庖，喧宾夺主之嫌。

如果你交付的程序代码经某人修改后，某些方面不再有效，按道理你不会再对该程序代码提供支持。此外，当你后来更新你自己的程序代码，并生成可供使用的新版本后，你的朋友必须仔细研读它，以便把它集成到他修改过的程序代码中。这样的程序代码就是加工过的，而不是可重用的。如果你购买了一个类库，而你不喜欢它的工作方式，这样事情就太糟糕了。除非你也购买程序源码，以便对它进行修改。这就又回到了上面提出的问题。为了更好地理解程序代码重用的概念，有必要给出一个比较确切的定义：真正的程序代码重用功能是这样一种能力，它可以把一种按相当通用的方式编制的程序代码重新用在你的软件中，用以构造更大的程序，而它仍然允许根据具体情况优化它的工作方式和工作内容。

对于大多数类型的程序代码重用来说，另一个问题是它们通常要求该软件的原开发人员和想重用该程序代码的人们使用相同的编程语言。例如，如果一个类库是用 C++编写的，基本上可以说该程序代码不可能重用在用非C++语言编制的应用程序中。由于同样的原因，一个 Java 类只能用在 Java 程序中。这样，尽管用面向对象的编程语言编制的程序代码可重用的部分要比用其他语言编制的程序代码可重用的部分多，但这也产生了很多限制。

那么怎样才能使每个人都企盼的程序代码共享和重用这一伟大思想成为实际的、现实生活中可用的编程技术呢？ 面向对象的编程技术作为一种解决现实问题的方案已经发展了一个长时期了，它已经施展了它的全部潜力。 在面向对象的编程技术中，不存在一种标准的框架，通过它让不同的经销商创建的软件对象在同一个地址空间中（极少穿越地址空间或网络，以及机器边界）彼此交互操作。按照 COM 规范的说法，面向对象的编程技术已经成为一群对象的孤岛，这些对象不能穿越应用程序的边界这一海洋，因而不能以一种有意义的方式彼此交流信息。

1.3　组件式设计思想登堂入室

作为开发人员，我们往往对高层工具有一种本能的排斥，因为它们限制我们只能使用这些工具的设计者提供的功能。其中更有甚者，简直是把你封闭在一个黑匣子中，它掩盖了在编程语言（例如 C）中可能出现的错误。这类工具中的最佳者允许你创建一些简单的应用程序或一些很诱人的原型，只要你不超越限定的一组功能即可。然而，有些事情用 C 做起来很直接了当，但用高层工具来做常常成为一场恶梦。

对于很多已经习惯于低层编程语言的开发人员来说，转向用 Microsoft Visual Basic 编程往往是一种缓慢而勉强的过程，因为他们心存疑虑。Visual

Basic 显然是一种功能优越的开发工具，对于构造原型来说更是了不起的手段。但开始时，这些功能反倒增加了他们对这种工具的排斥，他们不愿意把它用作一种实际的应用程序开发工具。 然而，随着时间的推移，开发人员发现 Visual Basic 还提供了某些更高级的功能，诸如通过 Visual Basic 的 *Declare* 语句可以很容易地调用 Windows API。这种功能起初来自一种限制，即：Visual Basic 不允许调用那些直接使用回调地址的 Windows API 函数，因为 Visual Basic 不提供函数指针。然而,开发人员很快意识到，通过用 C 或 C++ 编写动态链接库（DLL），而由动态链接库使用回调，就可以打破这种限制，因为 Visual Basic 允许调用用任何与 Declare 具有相同句法的语言编写的 DLL。如果你很喜欢用 Win32 API，你一定会很高兴地得知，Visual Basic 对于需要回调地址的 Windows API 调用已不再加以限制了。

不久，事情已经变得很明显，在 Visual Basic 中遇到的一切限制已经被 Visual Basic 的设计者预见到了，并已找到了解决办法。此外，开发人员还发现，通过用 C 编写 DLL，他们可以克服 Visual Basic 中潜在的性能方面的问题。 因为对于 GUI 程序来说，Microsoft Windows 操作系统内部的大量的闲置时间花在等待用户的操作上，所以 Visual Basic 性能往往并不是一个问题。然而，如果一个应用程序需要进行消耗大量处理器时间的操作，则要由开发人员编写 C 语言的模块，然后在 Visual Basic 中调用它们。例如，没有人想在一个 Visual Basic 程序中计算一个分数维（fractal）。

1.3.1 Visual Basic 扩展

开发人员还发现了某些比从 Visual Basic 中调用 DLL 更加令人高兴的事情，那就是 Visual Basic 扩展控件（VBX）。Microsoft 起初认为，只有 Microsoft 的开发人员要编写 VBX，而这些控件也许可以作为附加软件包与 Visual Basic 分开销售。尽管如此，Microsoft 仍然发布了 Control Developers Kit（控件开发人员工具包），其中提供了某些骨架示例程序代码，用于演示怎样用 C 创建你自己的 VBX。但 Microsoft 万万没有想到，在 Visual Basic 最初版本出现后短短几年时间，已经出现了一些小公司专门为 Visual Basic 生产附加的 VBX，使得 VBX 接口的生产成了一门小型工业。一下子，也是第一次，可以宣称程序代码重用实际上已经存在。没有人否认 VBX 是一种通用的软件组件，可由某个人编写它，而由另一个人把它用在 Visual Basic 工程中。这就诞生了组件式设计思想。

1.3.2 从 OLE 到 ActiveX

在这期间，在 Washington 州的 Redmond，在 Microsoft 公司总部，公司