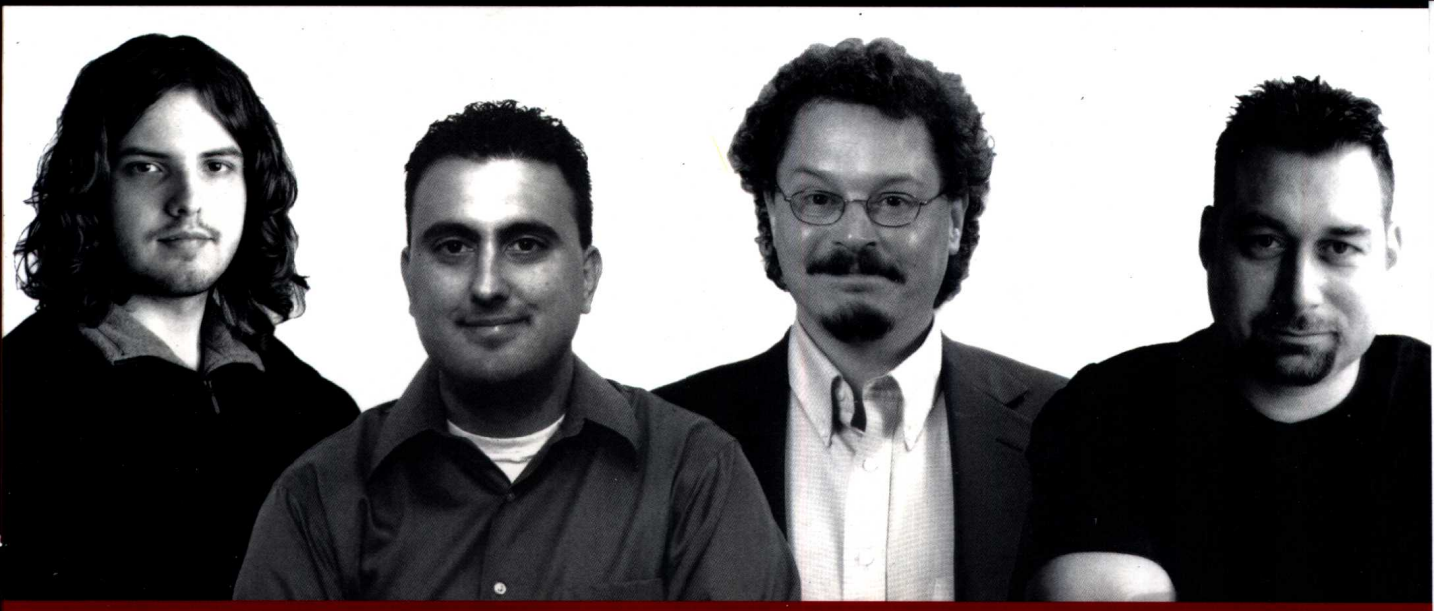
PROGRAMMER TO PROGRAMMER™



Visual C# .NET:
A Guide for VB6 Developers

Visual C# .NET 编程经典
——从 VB6 到 Visual C#.NET 快速进阶

Brad Maiani James Still 等著　康博 译

清华大学出版社
http://www.tup.com.cn

Visual C# .NET 编程经典

—— 从 VB6 到 Visual C# .NET 快速进阶

Brad Maiani
James Still
等著
康　博　译

清华大学出版社

(京) 新登字 158 号
北京市版权局著作权合同登记号：01-2002-3177

内 容 简 介

随着.NET 的日渐成熟和逐步推广，开发人员目前急需转换到新的 VS.NET 环境中来。C#是 Microsoft 专门为.NET 量身打造的一门语言，集成了 VB、VC 等众多语言的优点，是广大开发人员的首选转换语言。

本书从 VB6 开发人员的角度出发，首先概述了.NET Framework 和 Visual Studio .NET，介绍如何使用 Visual Studio .NET 和 C#开发 Windows 应用程序、C#语言和 C#的面向对象特性；接着，通过创建一个应用程序来学习如何使用控件；然后介绍类库及其用法，如何在 C#中使用现有的 VB6 代码，以及如何使用 ADO.NET 进行数据访问；最后介绍如何部署已经完成的应用程序。

本书适用于广大的 VB6 开发人员，能够在现有知识的基础上，帮助尽快地掌握使用 Visual C# .NET 语言来开发.NET Framework 应用程序。

Brad Maiani,James Still et al: Visual C# .NET: A Guide for VB6 Developers
EISBN：1-86100-717-5

书　　名：Visual C#.NET 编程经典——从 VB6 到 Visual C#.NET 快速进阶
作　　者：Brad Maiani James Still　等著　康博　译
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.com.cn
责任编辑：徐燕萍
封面设计：康博
版式设计：康博
印 刷 者：北京密云胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：28.5　**字数**：729 千字
版　　次：2002 年 12 月第 1 版　2002 年 12 月第 1 次印刷
书　　号：ISBN 7-302-06121-1/TP・3658
印　　数：0001～4000
定　　价：58.00 元

前　言

置身在 Microsoft 编程环境中的每个人都能感受到.NET Framework 所带来的冲击力。.NET 被设计成一个全新的开发环境，几乎能够开发运行在 Windows 中的所有应用程序。从而让用户能访问原来只能由 C++程序员涉及到的编程领域。

伴随着.NET Framework 诞生的有许多新的语言，但是最为引人注目的是 C#，我们将在本书中专门来学习它。C#是最适合.NET Framework 的语言，因为它是惟一一种为编写.NET 应用程序而特意打造的语言。C#是一门面向对象的编程语言，汇集了 C++、Java 和 Visual Basic 等语言的优点，不仅为语言提供了强大的功能和典雅的风格，而且易于学习。

伴随着新开发平台和新语言诞生的还有全新的 IDE—— Visual Studio .NET。Visual Studio .NET 允许开发 C#、Visual Basic .NET 和 Visual C++ .NET 应用程序，还包括 Web 应用程序和 Web 服务等。它提供了许多优于 VS6 的新特征，包括可折叠的文本编辑器、即时语法检查(就像 Word 一样，Visual Studio .NET 会在有语法错误的地方加波浪线标注)，采用自动缩进格式，并且能够非常容易地通过 Server Explorer 来访问位于本地计算机和网络上的服务。

本书读者对象

本书读者对象是想尽快将开发技巧移植到 C#中的 Visual Basic Windows 程序员。如果您想要开发 C#应用程序，想要一本实用、快速和基于解决方案的教程来学习新的语言和环境，那么本书是非常合适的。本书适合于想继续在 Visual Studio 环境中(现在是 Visual Studio .NET 或者是 Visual C# .NET 标准版)开发解决方案的广大编程人员。

本书不是一本有关 C#的百科全书——.NET Framework 的在线文档提供了.NET Framework 中所有类的概述，这自然是 C#的参考材料。在本书中，我们极力避免为您提供大量的理论资料，而是将精力集中在一些实际任务上，通过完成这些任务，您就会逐渐明白 C#和.NET Framework 的理论。

本书主要内容

本书首先讨论 C#、.NET 和 Visual Studio .NET，解释什么是.NET Framework ，C#的优越之处，C#与 Visual Basic .NET 的对比，以及 Visual Studio .NET 的概述。

在第 2 章“使用 Visual Studio .NET 和 C#开发 Windows 应用程序”中，我们会接触第一个 C#程序 SuperMind.NET。C#程序的结构不同于 Visual Basic 程序，例如在 C#中所有的代码都要位于一个类中，这就不同于 VB 的.bas 模块。Visual Studio .NET 没有将它的内部工作方式像 VB6 那样隐藏起来，所有自动生成的代码都可以被程序员查看(和编辑)。此外，C#的基本语

使用本书所需要的准备工作

因为本书并不打算成为一本深入介绍 C#和 .NET 每个知识点的综合大全，我们把精力集中放在了任务上，因此我们认为您极为可能想要执行它们。这不包括控制台应用程序，或者是手工编写代码来创建 Windows forms。本书假定您已经拥有了 Visual Studio .NET (专业版以上的版本)，或者是 Visual C# .NET 标准版。要想运行它们，必须首先安装带有 Service Pack 6 的 Windows NT 4.0、或者是 Windows 2000、Windows XP Professional、Windows .NET Server。注意对于 Visual C# .NET 标准版会有一些限制，这就是说在本书中使用的全部项目类型不是都能够在该版本中很容易地生成。特别是在第 8 章“创建自定义控件”中，需要 Visual Studio .NET 的完全版。

我们还假定您可以使用 SQL Server 或者是 MSDE 数据库服务器。如果您正在使用 Visual C# .NET 的标准版，您需要用 MSDE 来将 Server Explorer 连接到数据库。MSDE 可以和.NET Framework SDK(它会和 Visual C# .NET 一起进行安装)中的例子一同安装。

客户支持

我们一贯重视听取您的意见，特别希望听到您对本书的反馈，告诉我们您喜欢什么，不喜欢什么，您认为我们下次怎样才能做得更好。您可以将您的意见寄给我们，或者发电子邮件到 feedback@wrox.com。请您在反馈信息中务必写清楚本书的书名。

如何下载本书的范例代码

请您登录 Wrox 公司的站点(地址为 http://www.wrox.com/)，只需通过使用 Search 工具或使用书名列表，就可以方便地定位需要的书目。然后，单击 Code 列中的 Download 链接，或者单击本书具体页面中的 Download Code 链接，就可以下载相应的范例代码。

当您单击下载本书的范例代码时，会出现一个带有以下 3 个选项的页面：

- 如果您已经是 Wrox 开发社区的会员(即您已经在 ASPToday、C#Today 或 Wroxbase 上注册过)，您可以使用您的用户名和相关口令登录来接收代码。
- 如果您还不是会员，会被询问是否愿意注册以下载免费的代码。另外，您还能够下载几篇 Wrox 出版社的免费文章。注册后我们可以及时通知您本书的更新信息和新版本。
- 第三种方法是完全绕过注册，只是下载代码。

注册后再下载本书的代码并不是强制性的，但我们还是希望您在下载代码时进行注册，您的信息不会发送给任何第三方。对于更为详细的信息，您可能想要浏览一下我们的条款和条件，它们都链接在下载页面中。

在下载区域中，会发现从我们的站点所下载的文件都是使用 WinZip 压缩过的文档。当您将文件保存到磁盘上的一个文件夹后，需要使用一个解压缩程序(例如 WinZip 或 PKUnzip)来解压缩文件。在解压缩文件时，通常将代码保存到每一章所在的文件夹中。在解压缩的过程中，

应确保解压缩程序(WinZip、PKUnzip 等等)被设置为使用原有的文件夹名称。

勘误表

我们已经尽最大努力确保本书中的文本和代码没有错误，但是错误仍然在所难免。如果您发现本书存在错误，例如拼写错误或不正确的代码段，请给我们发来反馈信息，我们将不胜感激。勘误表的发送可以节约其他您学习本书的时间，而且能够帮助我们提供更高质量的信息。请将您的反馈信息 E-mail 到 support@wrox.com。我们将检查您的反馈信息，如果正确，将被粘贴到本书的勘误页面上，或者在本书的后续版本中使用。

要在我们的站点上找到勘误表，请访问 http://www.wrox.com/，并通过 Advanced Search 或者书名列表轻松定位本书页面。然后，单击 Book Errata 超链接即可，该链接位于本书的详细页面中的封面图形下面。

E-mail 支持

如果您希望直接向详细了解本书的专家咨询书中的问题，可以发送电子邮件到 support@wrox.com，要求在邮件的主题字段中带上本书的书名和 ISBN(国际标准图书编号)的后 4 位数字。一封典型的电子邮件应包括以下内容：

- 在主题部分中必须有本书的书名、ISBN 的后 4 位数字和问题所在的页数。
- 在邮件的正文部分应包括您的名字、联系信息和问题。

我们不会返回给您无用的邮件。为了节约大家的时间，我们需要了解一些具体细节。当您发送电子邮件信息后，它将经过下面一系列的支持：

- 客户支持：首先，您的信息将被递送到我们的客户支持人员手中，并由他们阅读。他们手中有最为常见的问题的文档资料，能够立即回答有关本书或者 Web 站点的任何常见问题。
- 编辑支持：接着，一些有深度的问题将被送到对本书负责的技术编辑手中，他们在程序设计语言或者特定的产品上有着丰富的经验，能够回答相关主题的详细技术问题。
- 作者支持：最后，如果编辑不能回答您的问题(这种情况很少发生)，他们将请求本书的作者。我们将尽量保护作者免受干扰，以便不影响其写作。然而，我们也非常高兴转寄给他们一些特殊的问题。所有 Wrox 公司的作者都为他们的书提供技术支持。作为回应，他们将发送电子邮件给用户和编辑，进而使所有的您受益。

Wrox 公司的支持部门仅对那些与我们出版的书目内容直接相关的问题提供支持，对于超出常规书目支持的问题，您可以从 http://p2p.wrox.com/论坛中的公众列表中获得支持信息。

p2p.wrox.com

如果想和作者及同行进行讨论，请加入到 P2P 邮件列表，而且我们独特的系统除了提供了

一对一的邮件支持系统以外，还通过邮件列表、论坛、新闻组等联系方式使您实现与程序员之间的联系。如果您向 P2P 发送一个问题，请放心它一定会得到登录邮件列表的 Wrox 公司作者和其他相关专家的帮助。无论您是在阅读本书，还是在开发自己的应用程序，都可以在 p2p.wrox.com 站点中找到许多对自己有帮助的邮件列表。

要想订阅一个邮件列表，只需遵守以下的步骤：

(1) 登录 http://p2p.wrox.com/站点。

(2) 从左边的菜单栏选择适当的类别。

(3) 单击希望加入的邮件列表。

(4) 按照订阅说明填写自己的邮件地址和密码。

(5) 回复您收到的确认邮件。

(6) 使用订阅管理工具加入更多的邮件列表并设置自己的邮件选项。

本系统为什么能够提供最好的支持

您可以加入到整个邮件列表，也可以只接收每周的邮件摘要。如果您没有时间和工具来接收邮件列表，可以直接查找我们的在线文档。独特的 Lyris 系统可以将一些没有用的垃圾邮件删除，并保护您的电子邮件地址不被侵扰。存在加入或离开列表、以及任何有关列表的其他常见问题时，请将邮件发送到 mailto:listsupport@p2p.wrox.com。

目　　录

第1章 .NET、C#和Visual Studio .NET

本章将从VB6开发人员的角度，介绍C# 语言和.NET Framework。主要介绍如下内容：

- .NET Framework，C#和其他.NET 语言的支持基础
- C#和 VB6 的不同之处
- C#和 VB.NET 的不同之处
- Visual Studio .NET，为构建.NET 应用程序而特意创建的 IDE

在我们开始介绍这些内容之前，首先来明确一下C#为什么相对于VB6来说是一个质的飞跃。

1.1 VB6中存在的问题

作为一名 VB 开发人员，您可能必须要使用完全不同的一些工具、技术和语言。进行前端开发时，您要使用 VB 窗体，标准的和第三方的ActiveX控件来开发桌面应用程序；而(D)HTML、JavaScript 和 ASP 代码是开发 Web 应用程序时的传统方法。Windows 应用程序和动态 Web 页的编程逻辑是完全不同的：VB 窗体是事件驱动的，而 ASP 页的服务器脚本是按照从头到尾的顺序进行处理的。

创建自定义组件的时候，您还会使用各种 COM 组件，例如 ADO、FileSystemObject 和 CDONTS 等等。在某些情况下，尤其是处理图形控件时，您可能还会使用 Windows API、最初为了 C++语言而使用 C++语言编写的底层函数。从 VB 调用这些函数总是会有问题，主要是因为函数需要的类型不同，VB 中缺少一些功能(例如指针)，并且需要使用句柄和内存地址——这些都不是典型的 VB 开发人员所习惯使用的东西。在某些时候，例如使用子类的时候，使用无效的参数调用函数，或者是在出错的时候调用函数，都会使得应用程序和 VB IDE 崩溃。

我们需要掌握很多知识，因为不同的语言不仅是语法不同，还有功能、约定和数据类型上的许多不同。这些就使得不同的语言之间即使是在 COM 的巨大帮助(COM 提供了可复用的编译代码的概念)下相互协作起来也很困难。

应用程序开发完成之后，还需要做很多事情。解决方案和所有的 COM(+)组件必须进行部署、注册。开发人员还必须留心系统和自定义组件的不同版本。如果您曾经做过这项工作，就会知道有时候这简直就是一种梦魇。这不是偶然的，它就是大家通常所说的 DLL Hell!

另外，还有非常重要的一点是。不管对 VB6 做多么大的改进，VB6 也始终会受到自身面向对象功能的限制。例如，它不能重载和实现继承，这就使得人们在将 VB 作为一门正式的语言方面打了折扣。

上述这些都是 VB6 中存在的问题，但是从另一方面来讲，像 C++这样的语言，它要比 VB 复杂得多，需要花费更多的时间来掌握。C++中的指针功能能够对内存提供很强的控制能力，允许开发人员做几乎所有想做的事情。但是它们也是后来在开发环境中突然出现的错误的最普

通的来源，导致开发人员需要花费大量的时间来发现和处理错误。C++比起 VB 来说功能更强大，性能更好，但是因为这个原因使用量反而要比 VB 少。

Microsoft 始终注意着这些(和其他)问题，并且在近几年，它的技术人员开发了一个更强大的开发平台.NET Framework，它能够成功地解决这些问题中的绝大部分。

1.2 .NET Framework

.NET Framework 是一个广泛而强大的开发平台，旨在帮助程序员花费较少的时间编写出安全、健壮的代码，允许不同的语言使用相同的公共类和服务，使得相互间的交互更加有效，并且使得组件的部署更加容易。Framework SDK 可以从 http://www.microsoft.com/net 上免费下载，可以安装在 Windows 2000 和 Windows XP 上。而它的可重新发布版本，只带有运行库而不带有所有的文档和例子，可以在 Windows NT 4.0 以后的各个版本中使用。在 Windows 未来的版本中，.NET Framework 将会嵌入在操作系统中。Microsoft 还开发了 Windows CE .NET，它是一个适用于像 PDA 和便携电话这种移动设备的支持.NET 的操作系统(它实际上嵌入了压缩的.NET Framework)。很可能在将来，我们会看到其他操作系统(例如 UNIX/Linux)支持的.NET Framework。

Framework 是由两个主要部分组成的：公共语言运行库(CLR)和 Framework 类库(FCL)，我们会在下面的小节中分别进行介绍。

1.2.1 公共语言运行库

VB6 有一个特定的运行库(MSVBVM60.DLL)来执行代码，并为所有的应用程序提供公共函数和功能。.NET Framework 有一个由所有.NET 语言共享的公共运行库。公共语言运行库(CLR)是负责执行托管代码的引擎，使用.NET 语言和编译器来编写和编译代码(在下一节我们会介绍有关这方面的更多内容)。下面是 CLR 提供的一些最重要的特性：

- 跨语言集成—— 不同的语言都是以 .NET Framework 为目标的，例如 Visual Basic .NET 和 C#，通过创建和使用以其他语言编写的类实例，甚至通过继承和扩展用不同语言编写的类，可以很容易地实现语言之间的互操作。
- 跨语言异常处理—— 和第一点类似，支持.NET 的语言可以处理使用其他语言编写的对象所抛出的异常。
- 自动内存管理—— 垃圾收集的功能能够使您自由地释放不再被使用的对象的内存空间，并且能够解决在设计糟糕的 VB6 对象中所存在的循环引用问题，该问题常常会引起内存泄漏。
- 代码访问安全—— 开发人员可以指定运行应用程序的用户必须有运行特定过程的某些权限。
- 更加易于部署和版本更新—— 组件不再需要在 Windows 注册表中进行注册，它们只需被复制到客户端即可。允许在同一个客户端有相同组件的多个版本，因此在不同的位置安装组件的新版本不会中断使用旧组件版本的现有应用程序。

要想提供这些优点，CLR 必须知道它所要执行代码的详细资料，例如在代码中定义的类型

(类)，创建的对象以及与其他对象间的关系。所有必需的详细资料都是由编译器产生的，并且会以元数据的形式装载在生成的编译代码中，我们将在下一节中介绍有关这方面的更多内容。

1. 中间语言和 JIT 编译

要想编写一个使用.NET Framework 运行的应用程序，可以从许多支持.NET 的语言中任选一种，例如 C#、VB.NET、Fortran.NET、Eiffel.NET、COBOL.NET 等等。然后可以使用命令行编译器来编译源文件，例如 CSC 或 VBC 是分别对应于 C#和 VB.NET 的(不使用 Visual Studio .NET 时)。虽然所生成的编译代码不是本机代码，而是中间语言(IL)，更具体地说，是 Microsoft 中间语言(MSIL)，是一种不带有特定 CPU 体系结构中的函数和调用的汇编语言。

当执行应用程序的时候，会按照它的需要，由 CLR 载入其 IL 代码并编译为机器代码，这就是通常所说的实时(JIT)编译。当首次调用方法的时候，每个方法的 IL 代码会被编译为机器代码，然后会为了以后的调用将这些机器代码缓存起来。这就意味着如果一个方法从来没有被调用过，它的 IL 代码就永远不会由 JIT 编译器转换为机器代码。在 JIT 编译期间，还会检查代码的类型安全性：编译器确保代码不会试图访问无效的内存地址，返回和传输的是正确类型的变量等。

使用 IL 和.NET CLR 的实时编译有一个好处：能够生成独立于 CPU 和平台的代码。事实上，在 IL 中不包含对 Windows API 和 Intel CPU 基本函数的调用，只包含对定义在.NET 核心类型和基类中函数的调用。这意味着如果有一个适用于 Linux 的.NET Framework 的完整实现，我们就可以在 Linux 中运行在 Windows 中开发的 IL 代码，CLR 就会根据这些 IL 代码来编译本机代码。这对于 CPU 的独立性来说也是如此：我们使用 Intel 计算机开发和编译为 IL 的代码，可以在支持.NET Framework 的非 Intel 的计算机上运行。CLR 会检测 CPU 的类型，从而生成相应的机器代码。这会提供更好的性能，因为即使我们编写和编译了一次代码，CLR 会在第二次编译过程(将 IL 生成本机代码)中针对目标 CPU 对代码进行优化，使用特定的 CPU 指令(例如 Pentium4 指令)可以提高应用程序的性能。

2. 元数据和程序集

我们在前面已经提到过，编译器在生成 IL 代码的同时还会生成元数据。元数据是一组数据表，保存的是编译后的 IL 代码所使用的和引用的类型信息。其中，包括类型(类)和它们成员的名称、参数、对其他类型(一些成员可以作为返回或输入类型使用，或者可以在过程中进行实例化)的引用，以及其他细节。在实践过程中，您可以将它看作是 COM 环境中使用的 IDL 文件和类型库的一个极度扩展的版本(虽然元数据比它们要大得多)，但是元数据装载到 IL 代码上以后，就不再需要在其他任何地方再进行任何注册。元数据的独特优点如下所示：

- 安全的代码和验证—— 当 CLR 执行 IL 代码并将其转换为机器代码的时候，它会检验代码会不会做一些危险的事情，例如读取还没有写入内容的内存，将一个错误类型的参数传递给一个函数，或者是参数的数目错误。
- 智能感知—— 代码编辑器可以实现智能感知，当开发人员输入对象的名称或使用特定工具的时候，可以查看类型中所包含成员的详细信息，类似于 VB6 的对象浏览器。智能感知现在由 Visual Studio .NET(我们将在本章的后面讨论这个产品)实现了，但是类似的功能也可以在任何其他的第三方编辑器中实现。

元数据还具有前面介绍过的 CLR 一些重要的特性，例如更加易于开发和版本更新。因为所有的类型都嵌入在相同的文件中，不再需要载入和注册其他文件，因此减少了与同一组件中原有版本的其他文件的混淆性和不兼容性。

C#编译器的输出是通常所说的托管模块文件，包含了 IL 代码和元数据，但是 CLR 不直接使用托管模块；它使用的是程序集。程序集是代码可以被执行和复用的最小单元，由一个或多个带有像图片这样可选的资源文件托管模块组成。当资源文件在命令行中编译的时候，它们默认产生带有单个托管模块的程序集，但是这种行为可以被改变为只构建托管模块，然后根据多个模块创建程序集。(如果您从 Visual Studio .NET 环境中编译源程序，就不必采用这种方法)。程序集还有一个清单数据表，列出了数据表中包含的所有文件。

在这里概括一下我们在本节中介绍的内容。图 1-1 所示的是程序的编译流程图，展示了一段.NET 应用程序代码从最初的编译生成中间语言和元数据，到程序集和它加载的中间语言代码，然后是验证、JIT 编译，最后生成用于执行的本地代码。

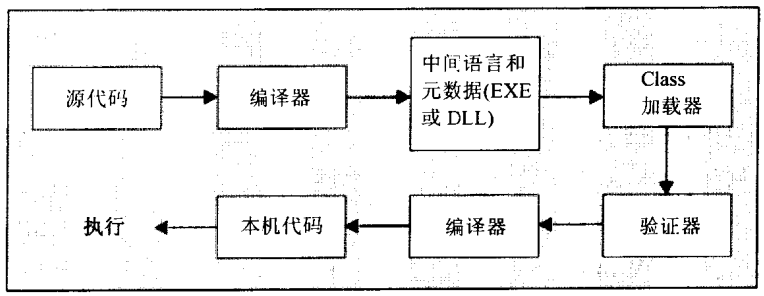


图 1-1

3. 不同的语言

要想编写一个程序，很明显需要一门程序语言。但是到目前为止我们所讲述的只是对.NET Framework 的总体介绍，并没有具体介绍一种语言。在本书前面提到过，有多种支持.NET 的语言。Microsoft 提供了 C#、Visual Basic .NET、J#(Java 的.NET 实现)、JScript.NET 和托管的 C++ 语言，除此之外许多其他的软件零售商也都开发了他们语言的.NET 版本。在这些语言之中，其中的任何一门语言都可以互操作、继承和扩展其他的语言，就像它们都是用一种语言编写的那样。当然不同的语言还是有不同的特征和语法，但是如果它们想要将 CLR 设定为它们的目标编译器，就必须遵守通用语言规范(CLS)。CLS 定义了一个任何支持.NET 的语言都必须支持的公共类型系统子集，例如产生元数据，具有允许定义和继承类的面向对象特征等。一门语言可以添加比 CLS 定义的更多特征，但是您使用这些额外的特征会损害与其他语言的互操作性，因为其他的语言并不支持这些额外的特征。

如果一门语言符合 CLS，它的编译代码就可以很容易地由其他的语言进行重用，因为它的编译器会像其他语言的编译器一样生成 IL 代码和元数据。这就意味着一个程序用一种语言编写，可能不如分别用不同的语言编写更快，因为最终生成的编译代码是相同的。事实上，用不同的语言编写相同的程序，最终生成的 IL 代码会有一些不同，但是这是由编辑器转换具体语言指令和语句的方法来决定的。一般而言，输出只有微小的不同，并且性能对于所有的语言来说通常都是相同的。