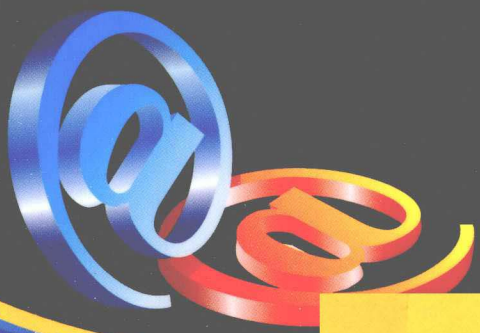
在线服务：视频库、源代码库、专业论坛、专家实时支持

C#编程网络大讲堂

郑千忠 邓德华 等编著



DVD 257段全程配音语音教学视频
全书实例源代码，使学习、分析、调试程序更方便

在线服务方式 在线服务网站：www.itzcn.com
QQ群在线服务：45368980、33925615、107423140

清华大学出版社

在线服务：视频库、源代码库、专业论坛、专家实时支持

C#编程网络大讲堂

郑千忠 邓德华 等编著

DVD 257段全程配音语音教学视频
全书实例源代码，使学习、分析、调试程序更方便

在线服务网站：www.itzcn.com
QQ群在线服务：45368980、33925615、107423140

清华大学出版社
北　京

内 容 简 介

本书全面介绍C#编程知识，全书共分4篇18章，内容包括：C#基础入门篇（第1～9章），介绍C#的开发环境和基础知识；C#实际应用篇（第10～12章），介绍创建Windows窗体应用程序和各类Windows控件的使用，MDI程序设计，ADO.NET数据库访问技术等内容；C#高级编程篇（第13～17章），本篇是本书的重点之一，介绍GDI+绘图，文件和注册表操作，以及XML编程、LINQ查询、Windows高级操作等知识；C#实例开发篇（第18章），介绍影碟出租系统综合案例。本书配套网站www.itzcn.com提供了配套学习资源和在线互动学习平台，帮助读者实现交互式学习模式。

本书可以作为C#的基础入门学习书籍，也可以帮助中级读者提高编程技能，掌握面向实践的应用技能。

图书在版编目（CIP）数据

C#编程网络大讲堂 / 郑千忠，邓德华等编著. —北京：清华大学出版社，2011.1
ISBN 978-7-302-23973-4

Ⅰ. ①C…　Ⅱ. ①郑…　②邓…　Ⅲ. ①C语言－程序设计　Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2010）第205082号

责任编辑：夏兆彦
责任校对：徐俊伟
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京密云胶印厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：190×260　**印　张**：43.25　**字　数**：1080千字
（附光盘1张）
版　　次：2011年1月第1版　　**印　　次**：2011年1月第1次印刷
印　　数：1～4000
定　　价：79.00元

产品编号：035206-01

前　言

自 Microsoft 公司推出.NET Framework 和 C#语言以来，C#语言的快速流行，使之成为使用.NET Framework 的 Windows 和 Web 开发人员无可争议的首选语言。C#编程语言是.NET 语言的主打产品，Microsoft 提供该语言来开发各种各样的应用程序和组件。同时，C#语言也是真正的面向对象的编程语言，在.NET Framework 强有力的支持下，开发人员能够使用 C#语言便捷地编写各类应用程序及组件。从最简单的“Hello World”程序，到最复杂的企业应用程序，都可以用 C#编程语言实现。

1．本书内容

第一篇：C#基础入门篇（第 1～9 章）。本篇首先介绍 C#的开发环境，如安装配置 Visual Studio 2008；接下来详细介绍 C#的基础知识，包括常量和变量、数据类型、控制语句以及数组、结构等；然后讲解 C#面向对象的基础知识和关键技术，如类、方法、构造函数等；最后介绍字符串操作、正则表达式、委托与事件以及异常处理。

第二篇：C#实际应用篇（第 10～12 章）。本篇首先介绍创建 Windows 窗体应用程序；包含各类 Windows 控件的使用，如文本、按钮、标签、表单、列表、容器、树视图和计时器等；然后介绍 MDI 程序设计，包含如何创建 MDI 父窗体、子窗体，管理子窗体等；最后讲解使用 ADO.NET 数据库访问技术，包含操作数据库的 ADO.NET 核心对象，使用 ADO.NET 的数据读取、显示、编辑和保存技术等。

第三篇：C#高级编程篇（第 13～17 章）。本篇是本书的重点之一，首先介绍 GDI+绘图，如使用画笔、使用画刷、绘制文本、绘制图像等各种知识；接下来介绍文件和注册表操作，包含创建文件、读取文件、写入文件和保存文件等，以及操作注册表信息；最后介绍 XML 编程、LINQ 查询、Windows 高级操作，如 XML 读取与写入、使用 LINQ 查询、线程、内存管理等内容。

第四篇：C#实例开发篇（第 18 章）。本篇主要介绍一个大型案例：影碟出租系统，通过该案例，读者可以了解 C#项目的开发流程。

2．本书特色

本书引用大量来自一线论坛的问题来进行讲解，力求通过读者实际操作时的问题方法，使读者更容易地掌握 C#的管理操作。本书难度适中，内容由浅入深，实用性强，覆盖面广，条理清晰。

- ❑ **结构独特**　通过“问题描述→解决方法→知识扩展→触类旁通”形式将每个知识与实际应用中的问题相结合。
- ❑ **形式新颖**　用准确的语言总结概念、用直观的图示演示过程、用详细的注释解释代码、用形象的比喻帮助记忆。
- ❑ **技术文档**　将一些非常简单的知识点或者理论性的内容安排在这里。通常这些文档没

有具体的实际问题，但是读者又必须要了解，如一些概念和术语。

- **内容丰富** 涵盖了实际开发中 C#技术所遇到语法基础、控制语句、数组、字符串、ADO.NET、WinForm、MDI、绘图以及文件和 XML 等方面的热点问题。
- **随书光盘** 本书为实例配备了视频教学文件，读者可以通过视频文件更加直观地学习 C#的使用知识。
- **网站技术支持** 读者在学习或者工作的过程中，如果遇到实际问题，可以直接登录 www.itzcn.com 与我们取得联系，作者会在第一时间内给予帮助。
- **贴心的提示** 为了便于读者阅读，全书还穿插着一些技巧、提示等小贴士，体例约定如下。

  提示 通常是一些贴心的提醒，让读者加深印象或提供建议，或者解决问题的方法。

  注意 提出学习过程中需要特别注意的一些知识点和内容，或者相关信息。

  技巧 通过简短的文字，指出知识点在应用时的一些小窍门。

3. 读者对象

本书具有知识全面、实例精彩、指导性强的特点，力求以全面的知识性及丰富的实例来指导读者透彻地学习 C#各方面的知识。本书可以作为 C#的基础入门学习书籍，也可以帮助中级读者提高编程技能。

本书适合以下人员阅读学习。

- C#初学者以及在校学生。
- C#专业开发人员。
- 各大中专院校的在校学生和相关授课老师。
- 其他 C#从业人员。

除了封面署名人员之外，参与本书编写的还有于永军、张秋香、李乃文、张仕禹、夏小军、赵振江、李振山、李文才、吴越胜、李海庆、何永国、李海峰、陶丽、吴俊海、安征、张巍屹、崔群法、王咏梅、康显丽、辛爱军、牛小平、贾栓稳、王立新、苏静、赵元庆、郭磊、徐铭、李大庆、王蕾、张勇、郝安林、郭新志、牛丽平、唐守国等。在编写过程中难免会有疏漏，欢迎读者与我们联系，帮助我们改正提高。

编著

2010 年 8 月

目　录

第一篇　C#基础入门篇

第二篇 C#实际应用篇

第三篇　C#高级编程篇

第四篇 C#实例开发篇

绪　论

0.1 .NET 与 C#

.NET（全称为.NET Framework）提供了一种环境。在这个环境中可以开发运行在 Windows 上的几乎所有应用程序，而 C#是专门用于.NET 的一种新编程语言。例如，使用 C#可以编写出动态 Web 网页、数据库访问组件以及 Windows 桌面应用程序等。

1．什么是.NET Framework

Microsoft 发布的.NET Framework 简称为.NET，是支持生成和运行下一代应用程序和 Web 服务的内部 Windows 组件，它提供了托管执行环境、简化的开发和部署以及与各种编程语言的集成。简而言之，如果想要开发和运行.NET 应用程序，就必须首先安装.NET Framework。

（1）.NET Framework 组件

.NET Framework 主要有两个组件：公共语言运行库和.NET Framework 类库。公共语言运行库是.NET Framework 的基础。读者可以将运行库看作一个在执行时管理代码的代理，它提供内存管理、线程管理和远程处理等核心服务，并且还强制实施严格的类型安全以及可提高安全性和可靠性的其他形式的代码准确性。事实上，代码管理的概念是运行库的基本原则。以运行库为目标的代码称为托管代码，而不以运行库为目标的代码称为非托管代码。

.NET Framework 的另一个主要组件是类库，它是一个综合性的面向对象的可重用类型集合，读者可以使用它开发多种应用程序，这些应用程序包括传统的命令行或图形用户界面（GUI）应用程序，也包括基于 ASP.NET 所提供的最新创新的应用程序（如 Web 窗体和 XML Web Services）。

在图 1 所示的.NET Framework 平台上显示了公共运行时和类库与应用程序以及与整个系统之间的关系。

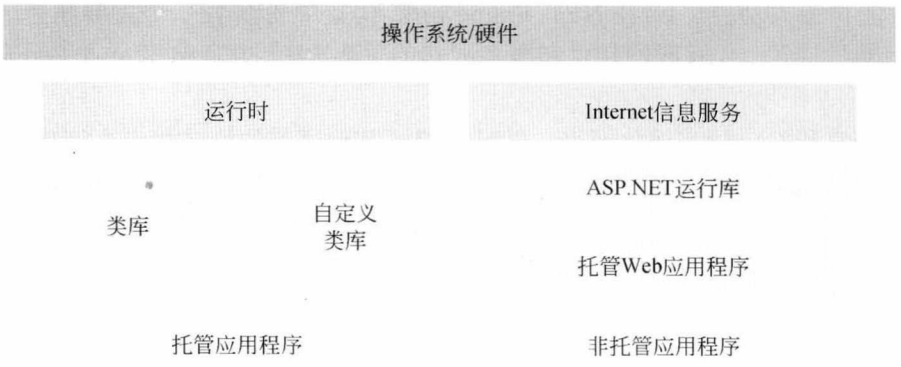


图 1　.NET Framework 平台

（2）.NET Framework 3.5 简介

从微软公司发布的第一个.NET Framework 以来，已经陆续发布了 1.0 版、1.1 版、2.0 版和 3.0 版。.NET Framework 3.5 版是目前最新的版本，也是功能最强大和完美的一个版本。

.NET Framework 3.5 版以.NET Framework 2.0 版和.NET Framework 3.0 版为基础，包括.NET Framework 2.0 和 3.0 版的 Service Pack，主要包括如下的组件。

- .NET Framework 2.0。
- .NET Framework 2.0 Service Pack 1，它更新包含在.NET Framework 2.0 中的程序集。
- .NET Framework 3.0，它使用.NET Framework 2.0 或.NET Framework 2.0 SP1（如果已安装）中存在的程序集，并且包含.NET Framework 3.0 中引入的技术所必需的程序集。例如，Windows Presentation Foundation(WPF)所必需的 PresentationFramework.dll 和 PresentationCore.dll 就随.NET Framework 3.0 一起安装。
- .NET Framework 3.0 Service Pack 1，它更新在.NET Framework 3.0 中引入的程序集。
- 一些新程序集，它们为.NET Framework 2.0 和 3.0 提供附加功能，同时还提供.NET Framework 3.5 中新采用的技术。

如果在计算机上安装 .NET Framework 3.5 时缺少上述任何组件，则会自动将安装它们。应用程序无论针对的是.NET Framework 2.0、3.0 还是 3.5 版，都使用相同的程序集。例如，对于使用 WPF 并针对.NET Framework 3.0 的应用程序，其所使用的 mscorlib 程序集实例与使用 Windows 窗体并针对.NET Framework 2.0 的应用程序是相同的。如果.NET Framework 2.0 SP1 已安装在计算机上，则 mscorlib.dll 已更新，并且两个应用程序将都使用 mscorlib.dll 的更新版本。

.NET Framework 2.0、3.0 和 3.5 版之间的关系不同于 1.0、1.1 和 2.0 版之间的关系。.NET Framework 1.0、1.1 和 2.0 版是彼此完全独立的，对于其中任何一个版本来说，无论计算机上是否存在其他版本，自己都可以存在于该计算机上。当 1.0、1.1 和 2.0 版位于同一台计算机上时，每个版本都有自己的公共语言运行库、类库和编译器等。应用程序可以选择是针对 1.0、1.1 还是 2.0 版。

2. 什么是 C#

C#是随.NET Framework 一起发布的一种新语言，是一种崭新的面向对象的编程语言，强调以组件基础的软件开发。不但结合了 Visual Basic 的简单易用性，同时也提供了 Java 和 C++ 语言的灵活性和强大功能。C#在.NET Framework 构架中扮演着一个重要角色。可以说是 Microsoft 公司面向下一代互联网软件和服务战略的重要内容，也是编写.NET Framework 应用程序的首选。

C#从 C 语言和 C++语言演化而来，并且考虑了其他语言的许多优点，例如 Visual Basic 的易用性。C#本身是面向对象的语言，然而 C#进一步提供了对面向组件（Component Oriented）编程的支持。现代软件设计日益依赖于包含和描述功能包形式的软件组件。这种组件关键在于，通过属性（Property）、方法（Method）和事件（Event）来提供编程模型；提供了关于组件的声明信息的属性（Attribute）；同时，还编入了自己的文档。C#提供的语言构造直接支持这些概述，这使得 C#语言自然而然成为创建和使用软件组件的首选。

C#具有几个非常优秀的用于构造健壮和持久应用程序的特性，如下所示。

- 垃圾回收将自动回收不再使用的对象所占用的内存。
- 异常处理提供了结构化的错误检测和恢复方法。
- 类型安全的语言设计则避免了读取未初始化的变量、数组索引超出边界或执行未经检查的类型强制转换等情形。

此外，C#还具有一个统一的类型系统，所有 C#类型（包括如 int 和 string 之类的基础数据类型）都继承于一个唯一的基类型：Object。因此，所有类型都共享一组通用操作，并且任何类型的值都能够以一致的方式进行存储、传递和操作。另外，C#同时支持用户定义的引用类型和值类型，既允许对象的动态分析，也允许轻量结构的内联存储。

为了确保 C#程序和库能够以兼容的方式逐步演进，C#的设计中充分强调了版本控制。许多语言都不太重视这一点，导致采用那些语言编写的程序，常常因为其所依赖的库的更新而无法正常工作。C#的设计在某些方面直接考虑到了版本控制的需要，其中包括单独使用的 virtual 和 override 修改、方法重载决定规则以及对显式接口成员声明的支持。

总之，C#是一个易于使用的、能够开发出功能强大、安全、稳定应用程序的语言，在本书的后面内容中将详细描述 C#的这些特性。

0.2 公共语言运行时简介

.NET Framework 提供了一个称为公共语言运行时（Common Language Runtime，CLR）的运行环境，它运行代码并提供使开发过程更轻松的服务。作为.NET Framework 的核心组件，它是执行时管理代码的代理，它提供内存管理、线程管理和远程处理等核心服务。图 2 所示是公共语言运行时环境中的各个部分及其提供的重要服务。

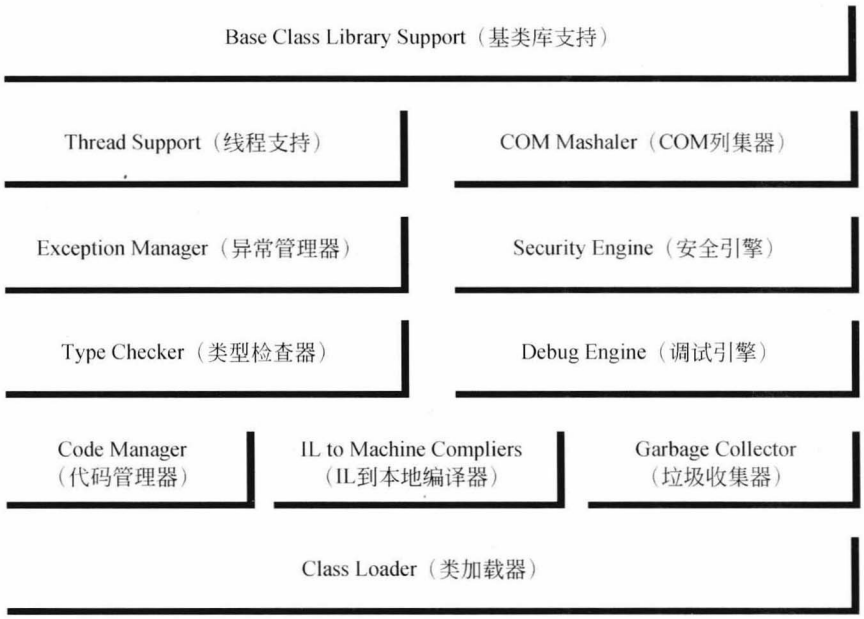


图 2　公共语言运行时环境

公共语言运行时通过公共类型系统（Common Type System，CTS）和公共语言规范（Common Language Specification，CLS）定义了标准数据类型和语言间互操作性的规则。Just-In-Time 编辑器在运行应用程序之前把中间语言（Intermediate Language，IL）代码转换为可执行代码。CLR 还管理应用程序，在应用程序运行时为其分配内存和解除分配内存。

1．公共类型系统

公共类型系统（Common Type System，CTS）定义了如何在运行库中声明、使用和管理类型，同时也是运行库支持跨语言集成的一个重要组成部分。公共类型系统执行以下功能。

- 建立一个支持跨语言集成、类型安全和高性能代码执行的框架。
- 提供一个支持完整实现多种编程语言的面向对象的模型。
- 定义各语言必须遵守的规则，有助于确保用不同语言编写的对象能够交互作用。

公共类型系统支持.NET Framework 提供的常用两种类型，其中每一类又可细分成子类型。

（1）值类型

值类型直接包含它们的数据，值类型的实例要么在堆栈上，要么内联在结构中。值类型可以是内联的（由运行库实现）、用户定义的或枚举的。

（2）引用类型

引用类型存储对值的内存地址的引用，位于堆上。引用类型可以是自描述类型、指针类型或接口类型。引用类型的类型可以由自描述类型的值来确定。自描述类型进一步细分成数组和类类型。类类型是用户定义的类、装箱的值类型和委托。

作为值类型的变量，每个都有自己的数据副本，因此对一个变量的操作不会影响其他变量。作为引用类型的变量可以引用同一对象；因此对一个变量的操作会影响另一个变量所引用的同一对象。

在图 3 所示的公共类型系统基本结构中给出了这两种类型及其可能出现的子类型。所有的这些类型都派生于一个基类型 System.Object，将会在后续章节对这些类型进行详细介绍。

2．公共语言规范

公共语言规范（Common Language Specification，CLS）是一组结构和限制条件，用作库开发者和编译器编写者的指南。它使用任何支持 CLS 的语言都完全使用库，并且使用这些语言相互集成。公共语言规范是公共类型系统的子集，它们一起定义了允许不同编程语言的标准集，由这些编程语言编写的应用程序可以互操作。

CLS 和.NET 自身都依赖于 Windows API（Application Programming Interface，应用程序编程接口）提供的低级服务。Windows API 提供了像菜单、按钮、列表框和标签等这些基本控件的类，提供了基本的 Windows 服务来管理文件、进程和内存。

CLS 定义了所有基于.NET Framework 的语言都必须支持的最小功能集。CLS 定义的规则可以概括如下。

- CLS 定义了命名变量的标准规则。例如，与 CLS 兼容的变量名都必须以字母开始，并且不能包含空格。变量名之间必须有所区别，除了变量名之间的大小写之外。
- CLS 定义了原语数据类型，如 Int32、Int64、Single、Double 和 Boolean。
- CLS 禁止无符号数值数据类型。有符号数值数据类型的一个数据位被保留来指示数值的正负。无符号数据类型没有保留这个数据位。

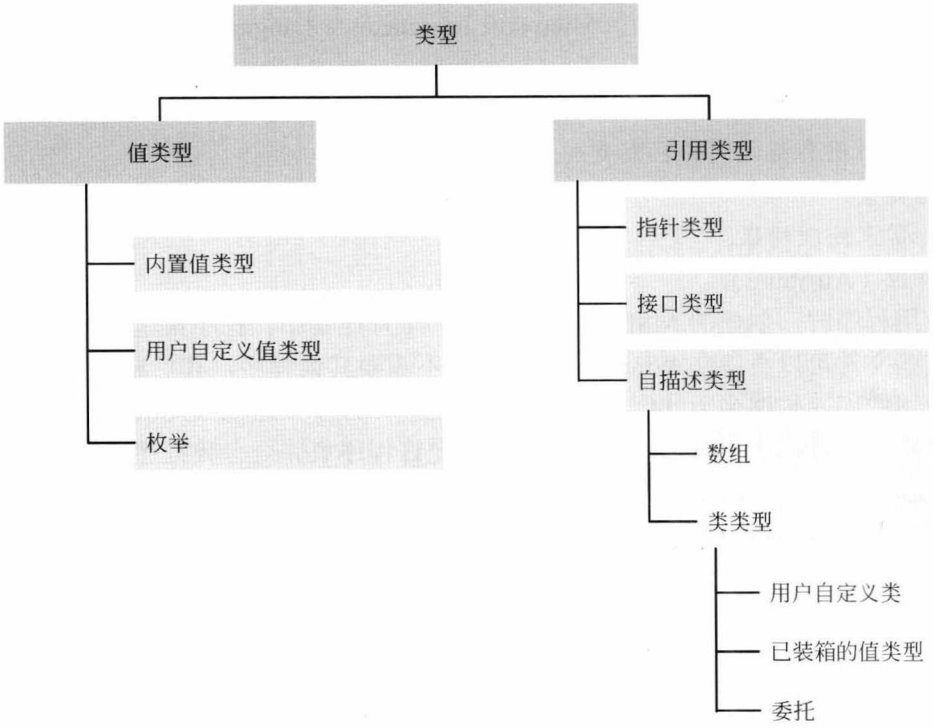


图 3　公共类型系统基本结构

- CLS 定义了对支持基于 0 的数组的支持。
- CLS 指定了函数参数列表的规则，以及参数传递给函数的方式。例如，CLS 禁止使用可选的参数。
- CLS 定义了事件名和参数传递给事件的规则。
- CLS 禁止内存指针和函数指针，但是可以通过委托提供类型安全的指针。

一般将完全兼容 CLS 的语言称为兼容 CLS 语言，除此之外任何语言都可以扩展基本的 CLS 需求。例如，有些语言支持无符号整型。但是，不鼓励使用非标准的功能，因为这样做就妨碍了语言之间的互操作性。

3．中间语言

使用.NET 语言开发的任何应用程序都必须在执行之前编译为可执行文件。传统的可执行文件包含了允许在特定 CPU 体系结构上执行的本机指令。不同厂商生产的 CPU 体系结构都不同，它们具有不同的寄存器，并且执行一套独有的指令。除了不同的 CPU 厂商会制作各种不兼容的硬件之外，即使是同一个厂商，如 Intel，也常常会制造出带有特殊指令附加功能的 CPU。通过把应用程序编译为独立于 CPU 的中间语言，当用户执行应用程序时，就会把中间语言优化为特定 CPU 的可执行文件。因此，Microsoft 公司为每一种目标 CPU 版本提供了不同版本的 JIT（Just-In-Time）编译器，以便利用各种 CPU 的独特功能，进而提高性能。

使用.NET 语言开发的任何应用程序在执行之前都会编译为目标计算机能够理解的语言，即本机代码，在.NET Framework 下这个过程可分为两个阶段。由于 CPU 体系的不同，首先

把应用程序编译成一种称为中间语言（Microsoft Intermediate Language，MSIL）的独立于硬件的格式，中间语言的主要特征如下。

- 面向对象和使用接口。
- 值类型和引用类型之间的巨大差别。
- 强数据类型。
- 使用异常来处理错误。
- 使用特性（Attribute）。

在编译应用程序时，创建的 MSIL 代码存储在一个程序集中，程序集包括可执行的应用程序文件（这些文件可以直接在 Windows 上运行，不需要其他程序，其扩展名是.exe）和其他应用程序使用的库（扩展名为.dll）。除了 MSIL 之外，程序集还包括元信息（程序集中包含的数据，也称为元数据）和可选的资源。元信息允许程序集完全自我描述，不需要其他信息就可以使用程序集。这样部署应用程序就非常简单了，只需要把文件复制到远程计算机上的目录下即可，因为不需要目标计算机上的其他信息，所以只在安装了.NET CLR 的计算机中执行。

第二个阶段就是使用 JIT 的阶段，它把 MSIL 编译为专用于目标操作系统和目标机器结构的本机代码，只有这样目标操作系统才能执行应用程序。使用 JIT 编译器把中间语言转换为可执行文件的过程通常称为 JITting。

4．托管执行过程

CLR 执行的代码称为托管代码（Managed Code），它的作用之一就是防止一个应用程序干扰另一个应用程序的运行。这个过程称为类型安全性（Type Safety）。使用类型安全的托管代码，一个应用程序就不会覆盖另一个应用程序分配的内存。C#开发的应用程序的执行由 CLR 控制，可以被视为托管代码。创建托管代码的方法如下。

（1）选择一个合适的编译器，它能够生成适合 CLR 执行的代码，并且使用.NET Framework 提供的资源。微软公司目前提供了 4 种兼容.NET Framework 的语言。

（2）把应用程序编译为独立于机器的中间语言。

（3）在执行时，必须把中间语言代码转换位本机可执行文件。本机可执行文件可以在目标 CPU 上执行。这个过程称为 Just-In-Time（JIT）编译。

（4）应用程序执行时，会调用.NET Framework 和 CLR 提供的资源。

上述托管代码的创建过程如图 4 所示。

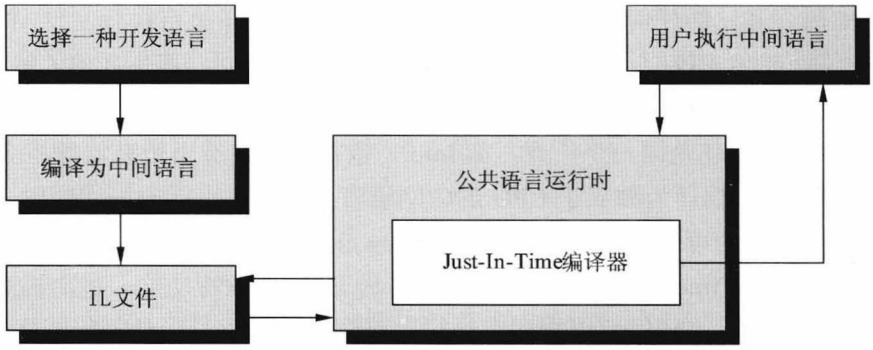


图 4　托管代码的创建过程

有些 Visual Studio.NET 生成的可执行文件不依赖于 CLR 提供的服务，这种类型的可执行文件称为未托管的可执行文件（Unmanaged Executable）或者未托管代码。

如图 4 所示，.NET Framework 使用托管代码执行过程主要有如下优点。

❑ 平台无关性

这意味着包含字节代码的同一文件放在任一平台中，运行时编译过程的最后阶段可以很容易地完成，这样代码运行在该特定的平台上。编译为中间语言就可以获取.NET 平台无关性，这与编译 Java 字节码就会得到 Java 平台无关性是一样的。

❑ 提高性能

实际上，MSIL 比 Java 字节码的作用还要大。因为 MSIL 总是即时编译的，而 Java 字节码常常是解释性的。JET 编译器并不是把整个应用程序一次编译完（这样会有很长的启动时间），而是只编译调用的那部分代码。代码编译过一次后，得到内部可执行代码就存储起来，直到退出该应用程序为止，这样在下次运行这部分代码时，就不再需要重新编译了。

❑ 语言的互操作性

使用 MSIL 不仅支持平台无关性，还支持语言的互操作性。简言之，就是能将任何一种语言编译为中间代码，编译好的代码可以与其他语言编译过来的代码进行交互操作。

5．自动内存管理

自动内存管理是公共语言运行时在托管执行过程中提供的服务之一。公共语言运行时的垃圾回收器为应用程序管理内存的分配和释放。对开发人员而言，这就意味着在开发托管应用程序时不必编写执行内存管理任务的代码。自动内存管理可解决常见问题，例如，忘记释放对象并导致内存泄漏，或尝试访问已释放对象的内存。本节会简单描述垃圾回收器如何分配和释放内存。

（1）分配内存

初始化新进程时，运行时会为进程保留一个连续的地址空间区域。这个保留的地址空间被称为托管堆（Manage Heap）。托管堆维护着一个指针，用它指向将在堆中分配的下一个对象的地址。最初，该指针设置为指向托管堆的基址。托管堆上部署了所有引用类型。应用程序创建第一个引用类型时，将为托管堆的基址中的类型分配内存。应用程序创建下一个对象时，垃圾回收器（Garbage Collector，GC）在紧接第一个对象后面的地址空间内为它分配内存。只要地址空间可用，垃圾回收器就会继续以这种方式为新对象分配空间。

从托管堆中分配内存要比非托管内存分配速度快。由于运行时通过为指针添加值来为对象分配内存，所以这几乎和从堆栈中分配内存一样快。另外，由于连续分配的新对象在托管堆中是连续存储的，所以应用程序可以快速访问这些对象。

（2）释放内存

垃圾回收器的优化引擎会根据所执行的分配决定执行回收的最佳时间。垃圾回收器在执行回收时，会释放应用程序不再使用的对象的内存。它通过检查应用程序的根来确定不再使用的对象。每个应用程序都有一组根。每个根或者引用托管堆中的对象，或者设置为空。应用程序的根包含全局对象指针、静态对象指针、线程堆栈中的局部变量和引用对象参数以及 CPU 寄存器。垃圾回收器可以访问由实时 JIT 编译器和运行时维护的活动根的列表。垃圾回收器对照此列表检查应用程序的根，并在此过程中创建一个图表，在其中包含所有可从这些