开发人员专业技术丛书

C#.NET Web开发指南

（美） Adrian Turtschi 等著

王海峰 冯义 郭卫平 等译

机械工业出版社
China Machine Press

本书介绍了.NET框架类库的功能。主要内容包括：Microsoft .NET平台简介、C#编程介绍、Visual Studio.NET集成开发环境、Windows窗体、使用TCP和UDP协议进行网络编程、远程连接、消息队列、ADO.NET、XML、ASP.NET、Web服务等。

本书编排独特、针对性极强。通过阅读本书，读者将学会如何使用C#创建新应用程序。本书适合从事网络开发人员，从其他语言转为C#语言的开发人员阅读。随书光盘中包含了本书使用的代码文件。

本书版权登记号：图字：01-2002-2180

图书在版编目（CIP）数据

C#.NET Web开发指南/（美）托特斯基（Turtschi, A.）等著；王海峰等译. -北京：机械工业出版社，2002.7

（开发人员专业技术丛书）

书名原文：C#.NET Web Developer's Guide

ISBN 7-111-10486-2

Ⅰ.C… Ⅱ.①托… ②王… Ⅲ.C语言－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第042865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：于杰琼 张鸿斌

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

2003年1月第1版第1次印刷

787mm×1092mm 1/16·34.75印张

印数：0 001 - 4 000 册

定价：59.00元（附光盘）

前　言

在电脑软件的历史上，很少有一种技术能够得到开发者和业界如此强烈的正面响应。全球已经有数百万的开发者下载了Microsoft的.NET软件开发工具包，已经出现了很多有关.NET平台及其相关技术和语言的教材、网站和新闻团体。

Microsoft在创建.NET上已经投入了数十亿美元进行了多年的研究。.NET是一种全面的策略，它由操作系统、数据库服务器、应用程序服务器和.NET运行时库组成，还包括运行于.NET平台之上的操纵语言。

很多人把.NET平台看作先前所说的Windows DNA的实际实现，也有人把它看作是改进先前技术和语言的结果。然而，这些仅仅说明了.NET是对Microsoft以前技术的重大改进。其实.NET平台是从头开始设计的，包括许多内在目标，如安全性、可升级性、可靠性、灵活性和互操作性。为了使.NET平台适合于企业和开发者，所有这些目标从一开始就被考虑到了。

.NET平台呈现了Microsoft思想的重大转变。建立.NET平台时，Microsoft表现出对开放标准极大的支持，如XML、SOAP和UDDI，而不是建立自己的标准和技术。而且.NET平台的核心部分（Common Language Infrastructure, CLI）和C#规范都已提交给ECMA，并通过了标准化。

C#来源于C和C++，是一种简单、现代、面向对象和类型安全的编程语言，由Microsoft的Anders Hejlsberg专门为.NET平台开发的语言，继承了许多语言的特征，如C、C++和Java。C#是为了综合Visual Basic的简单和C++作为面向对象语言的功能强大的优点而写的，对于开发者来说，C#使企业应用程序的创建、调试和配置变得很容易。有人预言，C#将成为在.NET平台上开发应用程序最受欢迎的语言。

作为Visual Studio IDE的下一个版本，Visual Studio.NET也是.NET战略的关键组成部分。Visual Studio.NET IDE也已经被整合并装入了大量的新功能。位图编辑器、调试器、Web窗体设计器、Windows窗体设计器、Web服务设计器、XML编辑器、HTML编辑器、Web浏览器、服务器资源管理器和多语言支持，所有这些都被集成在IDE中。

C#.NET Web开发指导的重点不是教你C#语言的核心内容，而是提供代码实例来帮助你使用.NET框架类库的功能。基础类库的.NET框架集合覆盖了大量API。尽管不可能在一本书中包含所有的特征，但是本书中包含了其中的关键思想、类库和.NET框架的API，希望本书能够帮助你容易地使用C#创建新的应用程序。

既然有这么多特征要学习和掌握，那就不要再等待了，咱们现在就开始吧！

Saurabh Nandu

www.MasterCSharp. com创建者

作者简介

Todd Carrico（MCDBA，MCSE）是Match.com的高级数据库工程师。Match.com是数字时代的门户，除它的主网站之外，它还为AOL、MSN和许多其他网站提供后端服务。Todd尤其擅长使用Microsoft技术设计和开发高性能、高可靠性的数据体系，曾经为许多公司做过设计、开发、咨询和工程管理，包括富士通、埃森哲、International Paper和GroceryWorks.com。Todd除了参与本书的写作之外，还负责Syngress的.NET系列其他图书的部分章节，包括《ASP.NET Web Developer's Guide》和《VB.NET Developer's Guide》。

Mark Tutt是MICROS Systems的高级软件工程师。MICROS为企业提供完全的信息管理方案，包括软件、硬件、企业系统集成、咨询和支持。Mark是许多软件包的主要设计者，包括为MICROS饭店业系列平台开发的客户服务方案、客户关系管理系统。除了产品开发之外，Mark在以下方面做了很大的贡献：系统集成软件的设计和开发，消费者指定产品的扩展，MICROS客户的整个技术方案完全集成MICROS的产品。

Jason Werry（MCSD）在澳大利亚经营一家咨询公司Synergy Data Solution，该公司为客户提供策略和技术咨询，他专长于基于Windows的企业系统开发。Jason具有使用Microsoft技术的深厚背景，现正在.NET平台上开发基于Web的先进应用程序。他的客户从一家台湾多媒体公司到各种政府部门。Jason是一个天生的程序员，13岁时就用汇编语言为Z80处理器写代码，从那时起，他使用过许多流行的编程语言，现在喜欢使用SQL Server、MTS、IIS、Visual Basic和C#。Jason具有数学和计算机科学学士学位，毕业于昆士兰州大学，他把作品献给他的爱妻，LiHsing。

Patrick Coelho（MCP）是University of Washington Extension、North Seattle Community College、Puget Sound Center和Seattle Vocational Institute的讲师，他讲授Web开发课程（DHTML、ASP、XML、XSLT、C#和ASP.NET）。Patrick是DotThatCom.com公司的创始人之一，这个公司提供咨询、在线资源开发和学生实习。现在他与David Jorgensen和nLogix一起从事.NET方案工作。Patrick具有华盛顿大学的理学学士学位，现在与他的妻子Angela居住在华盛顿州Puyallup。

David Jorgensen（MCP）是一名North Seattle Community College、University of Washington Extension、Puget Sound Center的讲师，也在Seattle Vocational Institute授课，还在Seattle为贫困学生讲授.NET和Web开发。David也通过他的公司DotThatCom.com提供实习机会，在线提供课程。David从St. Martin's College获得了计算机学士学位，与他的妻子Lisa和两个儿子Scott和Jacob居住在华盛顿州Puyallup。

Greg Hack是Allscripts Healthcare Solutions高级软件工程师，具有15年多的平台软件开发经验，范围从主框架到桌面，使用许多语言和技术。最近工作包括：通过一个基于Web的应用程序让病人能够查看自己的病历，以及一个Pocket PC应用程序，为护理内科医生传递临床信息。

Axel Goldbach是modulo3 GmbH的一个高级顾问。该公司是一个主要为德国服务的咨询公

司，专长为整个欧洲做工程管理咨询。modulo3是一个主要网络框架的过程实现专家，包括XP、MSF和V Modell。Axel现在为所有德国和中欧的modulo3客户提供高水平策略和技术咨询，他的职责包括不同环境下的多层应用程序的分析和开发，还进行modulo3技术侦查和培训，他的培训专长包括编程语言、网络和学术领域，如开发方法论、分析和解释技术、复杂性理论和可证明正确软件。

Joseph Albahari是一个自由专业顾问，具有10年设计网络系统的开发经验。他领导了一系列成功工程，从为始创公司定制应用程序框架，到为电信巨人开发的高性能OLAP和数据仓库系统。他在面向对象用户界面设计方面的知识已经在许多大而复杂系统的规划和生产中得到很好的体现，高度的抽象思维是至关重要的。Joseph也在SQL Server数据库管理方面经验丰富，已经具有为特殊需要的客户开发高性能方案的能力，如为一个复制系统提供字段级同步，以及大量成批拷贝代理。Joseph具有计算机和物理学的学士学位。

Adrian Turtschi（MCSE，MCSD）是Avanade公司的首席架构方案开发者，他负责提供移动计算空间方案，他从2000年秋使用Microsoft .NET平台，擅长用Web服务开发企业系统。他对使用Web服务来链接平台和系统边界尤其感兴趣。在加入Avanade之前，他在Boston的毕马威的Global Knowledge Exchange工作，帮助设计和开发毕马威的全球知识管理和协作方案，世界范围内已经有100 000专业人员使用。Adrian具有在瑞士、荷兰和美国的工作经验，具有数学和计算机学位，现在居住在德国的柏林。

关于随书光盘

随书光盘包含了本书每一章中涉及的代码文件。每一章的代码文件位于chXX目录下，如第8章的文件在ch08目录中。目录的结构取决于这一章中出现的代码的具体情况。

为了使用所提供的实例工作，需要操作系统至少是Windows 2000或Windows XP专业版，且有最新的服务包IIS 5.x和IE 6.0，因为ASP.NET和Web Service（ASP.NET的一部分）在早期的操作系统（如Windows 9.x/WindowsME/Windows NT）都得不到支持。还需要.NET SDK测试版2，在写这本书时它是最新的公共版本，以及Visual Studio.NET测试版2 IDE。

本书提供了大量实例，将有助于解决在开发.NET平台应用程序中遇到的问题，而不是只讲述C#理论和.NET程序。因此代码丰富也是本书的主要特点。

每章都包含说明原理的代码段和实例程序。第2章通过一系列实例程序介绍C#的概念，这不同于以往的面向对象语言。第4章帮助理解创建图形用户界面（GUI）风格的Windows窗体应用程序，该章介绍了Windows窗体应用程序，这些应用程序将在其他章中用到。第8章介绍的代码有助于使用ADO.NET在不同数据库间的相互操作，这一章也作为其他章涉及到的数据库的基础。第9章将学习使用.NET类库来与XML交互。

第5、6和9章讨论的是技术和应用程序编程接口（API），两个应用程序可以使用它们进行通信和相互联系。第5章集中讨论在TCP和UDP协议上应用程序间的通信，并对Web页中用到的技术进行概括说明。第6章和第11章中的代码实例是使用简单对象访问协议（SOAP）、对象串行化和解串化。

第7章中的实例使用Microsoft消息队列（MSMQ）在分布式应用程序中进行消息传送。第10章概述了ASP.NET，这有助于创建不同的应用程序，以增加复杂功能。

第12章创建一个Jokes Web Service应用程序。这一章的代码有助于创建Jokes Web Service和Windows Forms Client。

目　录

第1章 Microsoft.NET平台简介

本章要点：

- .NET平台简介
- .NET平台的特点
- .NET结构组件
- 探讨代码运行过程
- 追求标准化

1.1 简介

.NET平台是下一代软件开发的基础。Microsoft公司已经在它的开发中投入了大量资金，并且正在致力于使其成为一种新的标准。Microsoft的许多合作伙伴也宣布支持.NET工具和组件——读者可以登录以下网站查看提供.NET产品的发行商的最新名单。网站地址为：http://msdn.microsoft.com/vstudio/partners。

.NET平台不单纯是一种新的程序语言、软件开发工具包（SDK）或者一种操作系统。它为用户提供了很多新的强有力的服务，其中包括新的处理器独立的二进制格式、新的管理语言、对现有语言的管理性语言的扩展，下面还有很多很多。用户如果没有扎实的平台背景知识，就不可能有效地利用这些可以增强应用的新工具。

在本章中，将讲述.NET平台的各种组件。不仅介绍它们的基本概念和技术，而且还对用于描述它们的术语进行阐述。这样有助于读者更好地理解.NET平台的内部工作机制，同时还有助于全面吸收后面章节提供的信息。

1.2 .NET平台简介

.NET平台背后的思想是：计算机世界正在从一台PC机通过Internet之类的网络连接到服务器的时代转变为所有方式的智能设备、计算机和服务协同工作，以提供一种更丰富的用户经历。Microsoft正是用.NET平台迎接这种转变向软件开发者提出的挑战的。

.NET平台有几个组件——然而，不同的学者对其组成有不同的看法。有些学者把服务器（比如BizTalk和SQL Server）、各种服务（比如.NET My Services）以及它最初的可视组件和.NET Passport作为.NET框架完整的组成部分。然而，对多数人来说，在提到.NET时所能想到的就是.NET框架。它包括Visual Studio.NET（VS.NET）、.NET公共语言运行时（Common Language Runtime，CLR）和.NET基础类库（Base Class Library，BCL）。其他组件可能用于专门应用，但它们并不是所有.NET应用的必需组成部分。

浏览整个结构，.NET含有三个基本组件，分别是：

- **.NET 框架** 一种全新的应用开发平台。
- **几种.NET产品** Microsoft的多种应用是基于.NET 框架的，其中包括新版本的Exchange 和 SQL Server。现在，它们都支持可扩展标记语言（Extensible Markup Language，XML），并被集成到了.NET平台中。
- **几种.NET服务** Microsoft提供的，用于开发运行于.NET 框架上的应用程序。Microsoft提出的My Services计划实际上就是想把最关键的Web服务中的大多数用Microsoft商标进行包装。

.NET 框架本身可以分为以下三个部分：

- **CLR** 一种托管的执行环境，用来处理内存分配、查错以及与操作系统服务进行交互。
- **基础类库** 收集了大量编程组件和应用程序界面（API）。
- **两个顶级开发目标** 一个是为Web应用（ASP.NET）制订的，另一个是为常规Windows应用（Windows窗体）而制订的。

.NET 框架的优点包括：可以缩短开发周期（代码重复利用、更少的程序故障和支持多种编程语言）、易于部署、减少了整体安全性带来的与数据类型有关的错误，同时垃圾回收器减小了内存泄漏。总的来说，在.NET框架下开发的应用具有更强的可伸缩性和可靠性。

1.2.1 Microsoft .NET 和 Windows DNA

如果在一些推销场合，介绍.NET的内容听起来比较熟悉，其原因可能是因为.NET平台是所谓的Windows DNA的下一代产品。然而，虽然Windows DNA的确提供了创建实时的、可伸缩的分布式系统所需的一些创建模块，但总的来说，它并没有什么实物。

Windows DNA是一种技术规范，它致力于创建基于Microsoft的服务器产品、利用大量技术和语言（如ASP、HTML、JavaScript、MTS和COM等）的软件。而从开发者的角度看，其中很多都是互不相连的。所涉及到的所有服务器和语言都具有不同的API和类型系统，这使得实现互用性成为最大的挑战。下面指出了最大的不同点：.NET不仅仅是一种技术规范。从产品本身来讲，它将所包含的为了使这些n层应用的开发更简单些所需要的工具和语言巧妙地封装成一个相关的、全面的API。

1.2.2 Microsoft .NET体系结构

图1-1展示了.NET平台的体系结构。从本质上讲，.NET语言家族中的每个成员都根据公共语言规范（Common Language Specification），被编译为Microsoft中间语言（Microsoft Intermediate Language，MSIL或简单称为IL）输出。应用开发的主要类型是开发Web Form、Web Service以及Windows Form应用。这些应用通过XML和简单对象访问协议（Simple Object Access Protocol，SOAP）进行通信，从基础类库获得功能，然后在公共语言运行时环境中运行。在开发.NET 框架的应用时，Visual Studio.NET不是必须的，但它的确提供了一种可扩展的体系结构，使之成为开发.NET软件的理想选择。

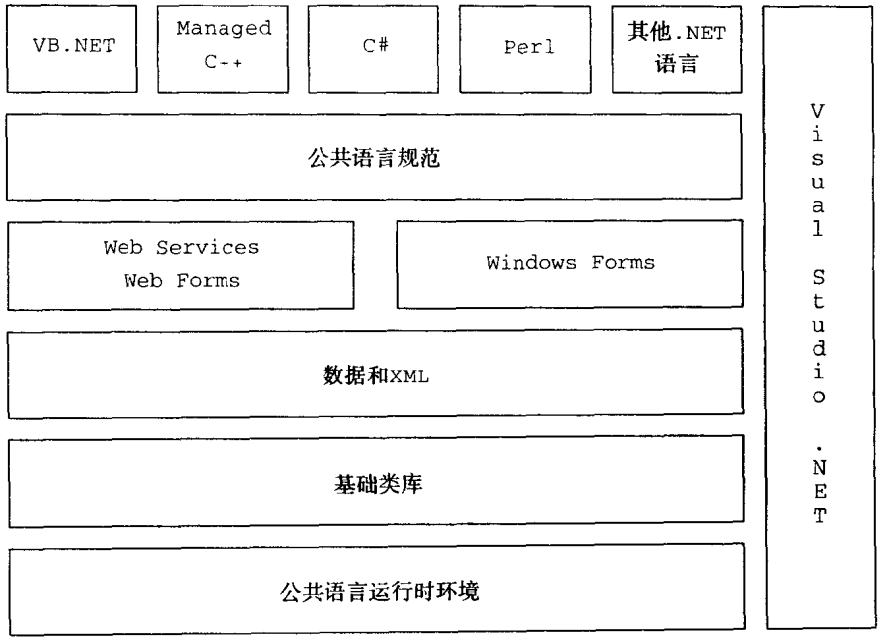

图1-1 .NET平台的体系结构

1.3 .NET平台的特点

.NET平台的核心是公共语言运行时环境、基础类库（Base Class Library）和公共语言规范。.NET基础类库展示公共语言运行时环境的特性与Windows API允许用户使用Windows操作系统特性的方式相同，并且它还提供了很多便于代码重用的更高级特性。

这种结构还有大量优点，而不仅仅是保证一致的API。通过把代码写入公共语言运行时环境和使用.NET基础类库，借助通用的面向对象的编程模型，可以获得所有应用服务。现在，还可以通过基于C语言的API调用DLL（动态连接库）来访问操作系统的一些函数，同时可以借助COM对象访问其他工具，使得开发者只需做少量的工作就可以使所有部分平稳地协调工作。有些特性只有那些使用低层语言，被迫作出设计决策的开发者才能使用。

这种新的编程模型大大简化了开发者编写Windows DNA应用软件，或者说开发几乎所有的Win32和COM工程所需的工作。开发者不再需要是一位对GUID、IUnknown、AddRef、Release、HRESULTS等有深入了解的Windows或COM体系结构的高手。.NET不是简单地把这些对开发者隐藏起来，而是在新的.NET平台中，这些思想根本就不存在。

对.NET开发者来说，另一个大的优点是它的错误处理模型是通过异常来完成的。为Windows平台开发软件的同时还意味着引入了不一致的因素，尤其是对于错误返回方式来说更是如此。有些函数返回的可能是Win32错误代码，有些返回HRESULTS，也有些会抛出异常。所有这些都要求程序员编写不同类型的错误处理代码。在.NET中，所有错误都借助异常记录，这极大地简化了代码的写、读和维护。由于公共语言规范和公共类型系统（Common Type System）

的存在，.NET异常还可以跨模板和语言进行工作。

1.3.1 多语言开发

由于很多语言都是面向.NET公共语言运行时环境的，所以现在可以简单地使用适当的语言来实现应用程序的某一部分。过去允许不同语言互用的方式，比如COM或CORBA，它们都是通过使用接口定义语言（Interface Definition Language，IDL）来实现的。而.NET平台是使用MSIL把不同语言集成来工作的。尽管它含有看起来像配件代码似的指令，比如对数值进行压栈和出栈操作和把变量移入或移出寄存器，它还含有一些用来管理对象和激活其方法、操纵数组以及抛出并捕捉异常的指令。

Microsoft公共语言规范描述了使用其他开发工具的开发者为了使自己的编译器能够输出IL代码以允许它们相互之间集成所必须完成的工作。Microsoft最新推出了几种可以产生面向.NET公共语言运行时环境的IL代码编译器，其中包括带控制扩展的C++、C#、Jscript和Visual Basic。此外，除Microsoft，还有其他几家公司也生产面向.NET公共语言运行时环境的语言的编译器。现在常见的有COBOL、Eiffle、Fortran、Perl、Python、Scheme，还有很多已经被不同卖主公布了。读者可以到http://msdn.microsoft.com/vstudio/partners/language/default.asp网站查看最新的产品清单。

为什么要关注IL的细节呢？因为这是.NET管理其众多跨语言（Cross-language）特性的关键所在。IL元数据在顶层就已经完成整个转换，所以无需使用接口定义语言（Interface Definintion Language）来启用跨语言功能。例如，对于IL定义的异常对象，不管.NET使用的是何种语言，这个异常对象将可以被捕捉。用C#编写的组件抛出的异常可以被使用它的Fortran应用捕获。此时不用考虑不同的调用规则或数据类型，这是一种无缝互用性（Seamless Interoperability）。

IL的使用使得跨语言继承成为可能。现在可以创建一个基于其他语言编写的组件的新类，而不需要这些组件的源代码。例如，可以用C++创建一个类，使之派生于用Visual Basic 实现的类。.NET使这种技术成为可能，这是因为它定义并提供了所有.NET语言共用的类型系统。

使用Windows DNA规范，开发应用软件的一个挑战是如何调试使用不同语言编写的应用。这里幸亏有统一的Visual Studio.NET 开发环境和以IL作为所有.NET语言的输出，才使得跨语言调试不必求助于汇编语言。.NET公共语言运行时环境完全支持对跨语言的应用程序进行调试。这种运行时环境同时还提供了内置的浏览堆栈的工具，便于查找断点和错误。

1.3.2 独立于平台和处理器

中间语言是独立于CPU的，并且位于大多数机器语言的上层 。.NET应用一经编写和创建，就可以在任何支持.NET公共语言运行时（CLR）环境的平台上执行。由于.NET公共类型系统（Common Type System）定义了在.NET应用程序中可以使用的基本数据类型的大小，同时应用是在公共语言运行时环境中运行的，所以应用软件开发者根本就无须考虑硬件和支持.NET平台的操作系统的任何细节。

虽然在开始编写本书时的.NET应用只能运行在Windows平台上，但在2001年6月27日，Microsoft宣布已经和Corel公司达成协议共同开发资源共享的C#编译器，同时还为Uinx的FreeBSD版本开发.NET 框架底层结构。希望在2002年的上半年可以推出第二个版本。

几周后，也就是2001年7月10日，Microsoft提供了由Ximain设计的.NET开放资源版本。Ximain是Linux公司中从事流行的GNOME用户界面的开发者。读者可以从www.go-mono.net站点看到这个项目，项目名称是Mono。这个开发组现在正在开发C#语言编译器和.NET 公共语言运行时环境。开发基础类库的工作也已经开始了。计划在2001年底推出第一个实用的Mono工程代码。

1.3.3 自动内存管理

对于使用不提供自动内存管理的开发环境的开发者来说，单是提到内存泄漏问题，就需要他们进行无休止的调试。即使对那些有幸使用某种形式的这种功能的开发者来说，他们也很可能需要花费一些时间去查找那些涉及到资源管理方法的代码所带来的不是很明显的错误（bug）。

具有Visual Basic或COM背景知识的开发者应该熟悉引用计数技术。这项技术在没有其他对象引用该对象，也就是说不再需要这个对象时，就释放该对象所占用的内存。虽然这从理论上听起来非常完美，但在实际应用中还存在一些问题。其中最常见的问题是循环引用问题，即一个对象拥有另一个对象的引用，而另一个对象本身又含有对第一个对象的引用。当内存管理器查找不被使用的对象时，这些对象的引用计数总会大于零。因此，除非这些对象被显式地删除，否则它们所占用的内存永远不会释放。

对于C或C++程序员——他们习惯于保证对象已被正确删除，即程序员自己管理内存——这听起来很正常，也是不信任其他任何人来管理资源的非常好的理由。然而，在.NET环境中，Microsoft致力于使软件的开发更简单些。在本章后面，将讨论.NET垃圾回收的工作原理，以及这种内存管理机制对严格的引用计数和手工内存管理方法的改进。

1.3.4 版本支持

开发（或至少是支持）Windows软件的时间不长的开发者是无法理解“DLL Hell”含义的。对于那些还不具有初步知识的人来说，当一个用户安装了和你的应用使用具有相同DLL的软件包时，就会发现自己已经处于DLL Hell中了。然而，你的应用中使用的是该DLL的1.0版本，而新的软件中是版本1.1。作为开发者，总是确保自己开发的所有代码都是100%向后兼容的，对吧？新的DLL可能会使你的应用出现某些奇怪的问题，或者干脆不能协调工作。经过大量检查后，你找出了出问题的DLL，并让客户用能够和你的软件协调工作的版本替换这个新版本。现在他们的新软件不能工作了……欢迎来到DLL Hell。很多开发者只是简单地安装应用目录中列出来所需的所有DLL，从而可以在应用程序加载库的时候第一个发现存在的问题。这样做虽然没有达到共享库的目的，但也不失为解决问题的一种方法。

COM将改变这种现象：其中一个基本功能是用户不能改变方法接口，用户只是简单地增加新方法。不幸的是，软件开发者通常都是完美主义者，他们不会留下一个“破碎”的功能来惹恼某些人。问题在于改变已经投入使用的组件接口，会对那些希望使用组件旧行为的客户软件产生负面影响。因为COM对象在安装时用到注册表中的信息，所以，简单地替换应用目录中的DLL或控制是不起作用的。

现在.NET结构分开了应用组件，从而使得应用总是加载那些它在创建和检测时用到的组件。

如果应用程序在安装后运行，那么这个应用程序应该一直处于运行状态。这些是通过配件（assembly）完成的，配件是一个.NET打包（.NET-packaged）组件。尽管当前的DLL和COM对象仍然含有版本信息，但是OS不会把它们用于任何实际应用中。配件包含版本信息，.NET公共语言运行时环境使用这些信息保证应用程序加载其赖以建立的组件。本章后面将详细地介绍配件和版本是如何工作的。

1.3.5 支持开放标准

现在，并不是所有想要使用的工具都运行Microsoft OS或使用Intel CPU。基于这一点，.NET结构依赖于XML和其最可见的下一代——SOAP。SOAP是新出现的一种标准，保证通过Internet发送信息，并且无论它下层结构如何，都可激活程序和应用。SOAP提供的方法使分离系统之间可以更容易的进行信息交换。更为重要的是，它允许用户从远程系统上激活方法，并返回结果。由于SOAP是与HTTP类似的一种简单的基于文本的协议，它可以很容易地通过防火墙。这一点与DCOM和CORBA对象不同。

.NET平台使用的其他标准包括统一描述、发现和集成协议（Universal Description，Discovery，and Integration，UDDI）、一个公司目录以及它们的XML界面和Web服务描述语言（Web Services Description Language，WSDL），这种WSDL描述了各段应用代码的功能。通过把.NET结构的大部分置于开放标准之上，同时通过提交为C#提议的设计标准并把.NET通用语言底层结构提交给ECMA，Microsoft希望能够看到自己领域之外的用户都接受的未来软件的新版本。

1.3.6 配置简单

现在，为基于Windows的应用软件开发安装程序的困难度是难以想象的，以致于大多数公司都使用第三方开发的工具来开发他们的安装程序。但即使这样也不是令人满意的。这通常涉及要在几个目录中安装大量文件、各种各样的注册表设置、安装需要的COM组件以及所需创建的各种快捷方式等等。彻底卸载一个应用软件几乎是不可能的，即使软件本身提供了卸载程序，在卸载过程中仍然会到处留下碎片。随着Windows 2000的发布，Microsoft引入了一种新安装引擎来解决这些问题。但是Microsoft安装包（Microsoft Installer Package）的设计者仍然可能会在某些问题上出现漏洞。即使使用那些专门为简化安装程序开发的第三方工具，正确安装可卸载的应用仍然意味着大量的工作。

.NET设计组一定也遇到了同样的问题，因为.NET的目标是希望能够一劳永逸地解决这个问题。注册表不引用.NET组件，同时幸亏使用元数据和映射（reflection），才使得这些组件是自我描述的。实际上，安装.NET应用程序只不过是把文件拷贝到一个目录中而已。类似地，卸载应用软件也只不过和删除这些文件一样简单。

> **开发和配置：使用Visual Studio .NET 设置工具**
>
> 当意识到配置应用软件和编写安装包通常需要大量工作时，Visual Studio.NET开发组在Visual Studio.NET环境中集成了大量设置工具。