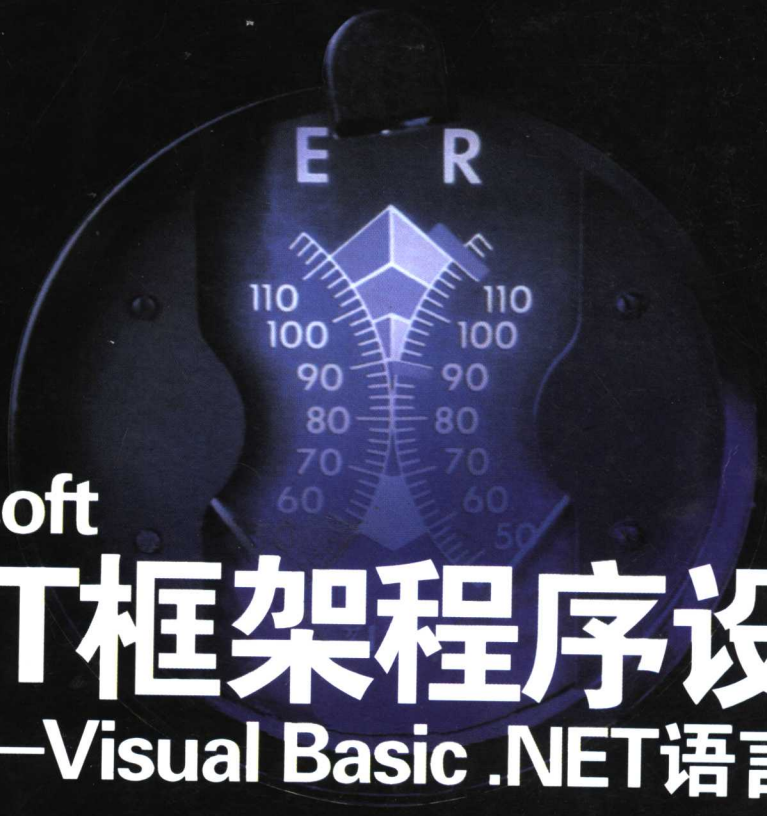
Microsoft

[美] Jeffrey Richter
Francesco Balena 著

李建忠 译



Microsoft .NET框架程序设计

——Visual Basic .NET语言描述

Microsoft .net

华中科技大学出版社

Microsoft .NET 框架程序设计
——Visual Basic .NET 语言描述

Jeffrey Richter
Francesco Balena 著
李建忠 译

华中科技大学出版社

Microsoft. NET 框架程序设计——Visual Basic. NET 语言描述
Applied Microsoft. NET Framework Programming in Microsoft Visual Basic. NET
Jeffrey Richter, Francesco Balena

图书在版编目(CIP)数据

Microsoft . NET 框架程序设计——Visual Basic . NET 语言描述/(美) Jeffrey Richter, Francesco Balena 著;李建忠 译
武汉:华中科技大学出版社,2004 年 6 月
ISBN 7-5609-3145-6

Ⅰ. M…
Ⅱ. ①J… ②F… ③李…
Ⅲ. Basic 语言-程序设计
Ⅳ. TP312

责任编辑:徐 烨 曾 光 封面设计:搜获科技
责任校对:陈 骏 责任印制:张正林
出版发行:华中科技大学出版社 (武昌喻家山 邮编:430074 电话:87557437)
http://press. hust. edu. cn
录排:搜获科技
印刷:湖北新华印务有限公司
开本:787×1092 1/16 印张:28. 25 插页:2 字数:658 000
版次:2004 年 6 月第 1 版 印次:2004 年 6 月第 1 次印刷
ISBN 7-5609-3145-6/TP · 525 定价:54. 00 元

(本书若有印装质量问题,请向出版社发行部调换)

前　言

随着时间的推移，我们以计算机为主导的生活方式不断地发生着变化。如今，每个人都注意到了互联网的价值，并开始越来越依赖基于 Web 的服务。就个人而言，我喜欢通过互联网购物、买票、比较产品、获取交通状况、阅读产品评价等。

然而，我发现仍然有许多事情在目前用互联网是无法完成的。比如，在周围寻找一家有特色风味的餐馆。而且，我还希望能够查询一下这个餐馆在晚上七点钟是否有座位。或者，假如我经营着一家商店，我可能想知道哪家厂商库存中有某种商品的现货。如果有多家厂商可以供货，那我希望能找出价格最低，或者是发货最快的那一家。

诸如此类的服务在今天还不存在，有两个主要的原因。首先，目前还没有一个标准来集成这些信息。各个厂商都有自己描述产品的方式。旨在描述各类信息的可扩展标记语言(XML)刚刚成为一门新兴标准。其次，开发集成这些服务的复杂性也是一个巨大的挑战。

微软认为销售服务是未来的发展方向。换句话说，就是企业提供服务，而感兴趣的用户消费这些服务。这其中，许多服务将会是免费的，有一些可以通过订购计划获得，同时还将有一些是按使用情况来付费的。我们可以将这些服务视作某种商业逻辑的执行。下面是一些服务的例子：

- 验证信用卡交易
- 获取从甲地到乙地的方位
- 查看餐馆的菜单
- 预定航班、客房或者出租车
- 更新在线相册里的照片
- 合并家长和孩子的日程表来安排一次家庭度假
- 从支票账户中支付账单
- 跟踪发送给我们的包裹

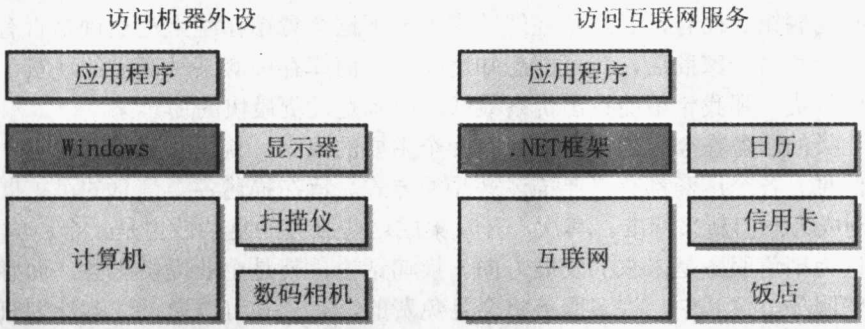
还可以列出很多类似的服务，任何一家企业都可以实现这些服务。毫无疑问，微软在不远的将来会创建并提供其中一些服务。其他一些公司也将会提供这样的服务，其中一些还可能在市场中与微软形成竞争。

那么，怎样才能从我们今天所处的世界中便捷地获得所有这些服务呢？我们又怎样利用并组合这些服务来为用户提供更丰富的特色应用(基于 HTML 或者其他技术)呢？举个例子，如果一些餐馆提供了他们的菜单访问服务，我们就应该能够写出一个应用程序来查询每个餐馆的菜单，搜索某一特定的口味或菜肴，最后向用户推荐那些离他们最近的餐馆。

> **注意**　为了创建这样一些丰富的应用，企业必须提供他们商业逻辑服务的编程接口。这样的编程接口必须可以通过远程网络(比如互联网)进行调用。这也是整个 Microsoft .NET 平台创新的主旨。简单地讲，.NET 平台创新之处就是关于人、信息和设备之间的互联。

让我用下面的方式对此做一个解释：我们知道，计算机都有外设(例如鼠标、显示器、键盘、数码相机、扫描仪等)与它相连。而操作系统，例如微软的 Windows，则将应用程序对这些外设的访问抽象化，以为我们提供了一个开发平台。我们完全可以将这些外设看作.NET 平台中的服务。

在.NET 平台所描述的这个崭新世界里，服务(或者外设)将与互联网络相连接。开发人员需要一种便捷的方式来访问这些服务，而 Microsoft .NET 创新的一部分便是提供一个这样的开发平台。下图展示了它们之间的一个类比。左边，从开发人员来看，Windows 是一个抽象了硬件外设的开发平台。右边，从开发人员来看，Microsoft .NET 框架则是一个抽象了 XML Web 服务通信的开发平台。



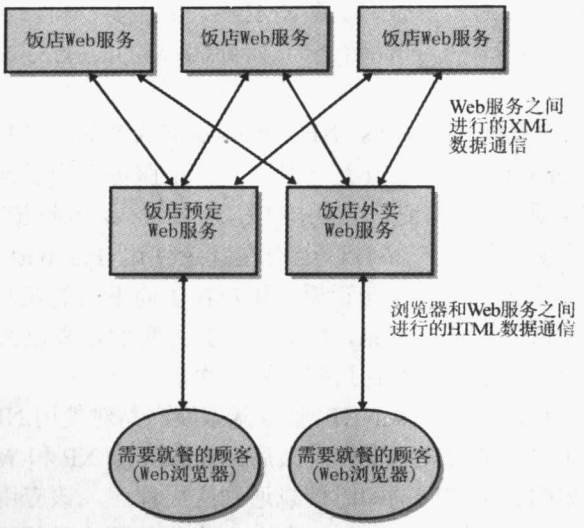
虽然是在相关标准的开发和定义方面的一个领导者——正是这些标准使这样的新世界成为可能，但微软并不拥有其中的任何标准。客户机通过创建特殊格式的 XML 来描述一个服务器请求，然后通过企业内部网或者互联网来发送它(通常使用 HTTP 协议)。服务器知道怎样分析这些 XML 数据，处理客户的请求，然后再以 XML 作为响应传回客户机。其中 SOAP(简单对象访问协议)在这里被用来描述通过 HTTP 协议发送的特殊格式的 XML。

下图描述了一组 XML Web 服务相互之间通过带 XML 负载的 SOAP 进行通信的情形。另外，图中还向我们描述了客户端应用程序可能与 Web 服务、甚至其他客户端(通过 SOAP 或者 XML)进行通信的情形。此图向我们展示的是客户通过 HTML 从一个 Web 服务器获得请求结果的情形。这时，用户可能会首先在一个 Web 窗体上填写自己的请求，然后将这个 Web 窗体发送回 Web 服务器。接着，Web 服务器处理用户的请求(包括与一些 Web 服务进行通信)，最后将结果以 HTML 页面的形式返回给终端用户。

提供 Web 服务的计算机必须运行在一个能够侦听 SOAP 请求的操作系统上。微软希望这样的操作系统是 Windows，但这也并非必须。任何能够侦听 TCP/IP 套接字(socket)端口，并能对该端口进行读/写字节的操作系统都可以胜任这样的工作。在不远的将来，移动电话、汽车、微波炉、电冰箱、手表、音响设备、游戏控制台，以及各种其他设备都将能够参与到 Web 服务的新世界中。

在客户端，操作系统必须能够从套接字端口读取字节以发送服务请求。客户端的计算机当然要能满足用户应用程序的需求。如果用户想创建窗口或者菜单，那么客户端的操作系统必须能够提供这样的功能，否则应用程序开发人员必须自己手动来实现。当然，微软希望人们在构建 Web 服务客户端应用程序时也能够使用 Windows。但同样，只是推荐

Windows，而非必须。



不管微软是否参与其中，充满 Web 服务的全新世界都将到来。整个微软.NET 平台创新的目的就是帮助开发人员创建和访问这些服务。

当然，如果愿意，我们也可以编写自己的操作系统，创建自己的 Web 服务器来侦听和处理 SOAP 请求。但是这通常很困难，并且要花费很长的时间。微软已经替我们完成了这些复杂的工作。我们可以直接把注意力放在真正的商业逻辑和服务上，而把底层的通信协议和基础构造交给微软来做——微软有众多热衷于此的开发人员。

Microsoft .NET 平台构成

我和微软及其各种技术打交道已经有很多年了。多年来，我有幸目睹了微软公司的各种技术创新：MS-DOS、Windows、Windows CE、OLE、COM、ActiveX、COM+、Windows DNA 等等。当我第一次听到 Microsoft .NET 平台时，我就知道它将续写微软不败的神话。它使我由衷地感到，微软的确是一个非常有远见卓识的公司。而且，看得出来，整个微软军团在实现他们这一远景规划方面是有雄心壮志的。

我们不妨将 Microsoft .NET 平台和下面几项微软的技术做一个比较。首先是 ActiveX，这只不过是 COM 的另一个好听的名字罢了。ActiveX 本身所涵盖的东西有限，而且和这个名字扯在一起的 ActiveX 控件从来就没有获得真正意义上的成功。另一个是 Windows DNA(Distributed InterNet Architecture)，这也是一个漂亮的市场标签而已，其实质内容仍是一大堆现存的技术。与前面两项技术相比，我对 Microsoft .NET 最有信心。写作本书也是对我这种信心的一个佐证。那么，到底整个 Microsoft .NET 平台创新包含有哪些内容呢？在下面的各个小节中我将向大家一一道来。

底层操作系统：Windows

由于 Web 服务和使用 Web 服务的应用程序仍然运行在计算机上，而且既然是计算机

就都要有外设，所以我们仍然需要一个操作系统。当然，微软建议人们选用 Windows。实际上，微软为整个 Windows 产品线添加了 XML Web 服务支持。在它们中间，Windows XP 和 Windows .NET 服务器家族产品将会为这种服务驱动(service-driven)的世界提供最好的支持。

特别地，Windows XP 和 Windows .NET 服务器家族产品已经集成了 Microsoft .NET Passport XML Web 服务支持。Passport 是一种用户认证服务。为了保证信息访问的安全，许多 Web 服务都需要用户认证。当用户登录到一台运行有 Windows XP 或者 Windows .NET 服务器家族中的产品时，用户在登录使用 Passport 认证的 Web 站点和 Web 服务时的效率将会大大提升。换句话说，用户在访问不同的互联网站点时，将不再需要每次都输入用户名和密码。可以想见，Passport 会为我们带来巨大的便利。因为我们只需保留一个身份标识和一个密码，而且只需输入一次！

另外，Windows XP 和 Windows .NET 服务器家族产品对使用.NET 框架技术实现的应用程序的加载和执行还提供了内置的支持。最后，Windows XP 和 Windows .NET 服务器家族产品还有一个新型的、可扩展的即时信息通知应用程序。该应用程序允许第三方厂商(如 Expedia、United States Postal Service，等等)和它们的用户进行无缝地通信。例如，当用户的航班(Expedia)延迟，或者邮包(U.S. Postal Service)送达时，他们可以收到自动通知。

不知道大家对这些怎么看，就个人而言，我对它们的期望已经有很多年了——确切地说，是有点迫不及待！

辅助产品：.NET 企业服务器

作为.NET 平台创新的一部分，微软提供了一些有价值的服务器产品供各种公司选择。下面是其中的一些企业服务器产品：

- Microsoft Application Center 2000
- Microsoft BizTalk Server 2000
- Microsoft Commerce Server 2000
- Microsoft Exchange 2000
- Microsoft Host Integration Server 2000
- Microsoft Internet Security and Acceleration (ISA) Server 2000
- Microsoft Mobile Information Server 2002
- Microsoft SQL Server 2000

这些产品刚开始很有可能仅仅出于市场目的，而被贴上.NET 标签。但是随着时间的推移和.NET 战略的推进，我相信微软会将很多.NET 特性集成到这些产品中。

开发平台：.NET 框架

大家目前可能已经看到.NET My Services 中的一些服务了，比如 Passport。这些服务目前都运行在 Windows 上，实现技术也还是如 C/C++、ATL、Win32、COM 等一些传统技术。随着时间的推移，这些服务以及许多新的服务最终必将会采用一些更新的技术来构建，比如 C#(念作“C Sharp”)和.NET 框架。

要点 虽然这里介绍的都是创建基于互联网的应用程序和 Web 服务，但是.NET 框架的能力远不至此。实际上，.NET 框架开发平台允许我们创建各种各样的应用程序：XML Web 服务、Web 窗体、Win32 GUI 应用程序、Win32 CUI(控制台 UI)应用程序、Windows 服务(由服务控制管理器控制)、实用程序和独立的组件模块。而本书所述的内容适用于上述任何类型的应用程序。

.NET 框架包含公共语言运行库(CLR)和.NET 框架类库(FCL)两个部分。它本身又是.NET 平台创新中一个关键的组成部分。实际上，它也是本书将要谈论的一切：开发面向.NET 框架的应用程序和 XML Web 服务。

刚开始的时候，CLR 和 FCL 还只能运行在各种版本的 Windows 平台上，包括 Windows 98、Window 98 第二版、Windows Me、Windows NT 4、Windows 2000，以及 32 位和 64 位的 Windows XP 和 Windows .NET 服务器家族产品。另外还有用于 PDA(如 Windows CE 和 Palm)和家电产品(小型设备)的.NET 微缩框架(.NET Compact Framework)。但是，在 2001 年 12 月 13 日，欧洲计算机制造商协会(European Computer Manufacturers Association，简称 ECMA)已经接受了 C#编程语言、部分的 CLR 以及部分的 FCL 作为标准。可以预见，在不远的将来，与 ECMA 标准兼容的这些技术将出现在各种操作系统和 CPU 上。

注意 .NET 框架并没有和 Windows XP(包括家庭版和专业版)打包在一起。然而，Windows .NET 服务器家族(包括 Windows .NET Web 服务器、 Windows .NET 标准服务器、Windows .NET 企业服务器以及 Windows .NET 数据中心服务器)将会包括.NET 框架。实际上，这也是 Windows .NET 服务器家族的名称由来。下一个版本的 Windows(代码名为“长角牛”，即 Longhorn)将会在所有版本中包括.NET 框架。就目前而言，我们还必须将.NET 框架和我们的应用程序一起分发给客户，也就是说安装程序需要安装一个.NET 框架分发包。微软创建了一个免费的.NET 框架分发包，大家可以到下面的地址获取它：*http://go.microsoft.com/fwlink/?LinkId=5584*。

绝大多数程序员都对运行时和类库比较熟悉。相信大家很多人对 C 运行时库、标准模板库(STL)、微软基础类库(MFC)、活动模板库(ATL)、Visual Basic 运行时库或者 Java 虚拟机等都有所涉猎。实际上，Windows 操作系统本身也可被看作一个运行时引擎和库。运行时和库为应用程序提供着各项服务，也是很多开发人员的最爱，因为我们不必一次次地重新设计同样的算法。

Microsoft .NET 框架为开发人员提供的技术比以前的任何微软开发平台提供的技术都要多，比如代码重用、代码专业化(code specialization)、资源管理、多语言开发、安全、部署、管理等。在设计.NET 框架时，微软还感到有必要改进目前 Windows 平台的某些缺陷。下面的列表向大家描述了 CLR 和 FCL 提供的一部分服务：

- 一致的编程模型

我们知道对于当前的 Windows 操作系统而言，某些功能需要通过动态链接库(DLL)来访问，而某些功能则需要通过 COM 对象来访问。然而，在.NET 框架下，所有的应用程序服务都将以一种一致的、面向对象的编程模型提供给开发人员。

- 简化的编程方式

CLR 的一个目的就是简化 Win32 和 COM 环境下所需要的各种繁杂基础构造。在 CLR 下，我们可以远离如下这些概念：注册表、全局惟一标识符(GUID)、IUnknown、AddRef、Release、HRESULT 等。注意，CLR 并不是简单地对开发人员抽象这些概念；相反，CLR 完全抛弃了这些概念。当然，如果我们编写的.NET 框架应用程序要与现存的一些非.NET 代码进行互操作，我们还必须对这些概念有足够的了解。

- 可靠的版本机制

相信所有的 Windows 开发人员都对“DLL hell”版本问题比较熟悉。出现这个问题的根本原因在于为一个新应用程序所安装的组件覆盖了一个现有应用程序正在使用的组件，而其结果往往会导致现有的应用程序出现一些奇怪的行为，甚至不能正常工作。.NET 框架采用了一种新型的版本机制来隔离应用程序组件，这种隔离策略可以保证一个应用程序总能加载它当初生成和测试时所使用的组件。这使得应用程序在安装之后的任何时候，都能按期望的行为运行。新的版本机制彻底关上了“DLL hell”的大门。

- 轻便的部署管理

当前的 Windows 应用程序都非常难以安装和部署。因为安装一个应用程序要照顾到许多事情：各种文件、注册表设置和快捷链接。另外，要完全卸载一个应用程序更是一种几乎不可能完成的任务。虽然 Windows 2000 引入了一种新的安装引擎来帮助解决这些问题，但是发布软件安装包的公司仍然免不了在一些事情上出错。.NET 框架希望将这些问题彻底变成历史。在.NET 框架下，组件(或者说类型)将不再受注册表的任何引用。实际上，大多数.NET 框架应用程序安装工作所需要的只不过是将文件拷贝到一个目录中，然后添加一个快捷链接到【开始】菜单、桌面和快速启动栏上而已。卸载应用程序也相当简单：直接删除它们就可以了。

- 广泛的平台支持

当编译器编译面向.NET 框架的源代码时，它实际上产生的是通用中间语言(Common Intermediate Language，简称 CIL)。只有在运行的时候，CLR 才会将这些 CIL 翻译为 CPU 指令。由于这个过程发生在运行时，所以它是面向特定的宿主 CPU 的。这意味着只要一台机器中包含有与 ECMA 兼容的 CLR 和 FCL，我们便可以将.NET 框架应用程序部署在该机器上。这样的机器可以是 x86、IA64、Alpha，或 PowerPC 等。当用户改变他们的硬件或者操作系统时，他们便会看到这种技术的价值。

- 无缝的语言集成

COM 允许不同的语言之间进行互操作。而.NET 框架则允许不同的语言之间进行无缝集成。在.NET 框架下，当我们使用一个类型时，不管它是用何种语言开发的，我们都可以像使用自己语言开发的类型一样来使用它。例如，我们可以在 Visual Basic 中创建一个类，然后在 C++中继承它。CLR 允许我们这样做是因为 CLR 要求所有面向它的语言都要遵循一种称作通用类型系统(Common Type System，简称 CTS)的规范。而通用语言规范(Common Language Specification，简称 CLS)则描述了一个语言与其他语言很好地集成在一起所必须遵循的规范。微软已经提供了几个面向 CLR 的语言编译器：托管扩展 Visual Basic .NET(包括 Visual Basic 脚本，即 VBScript 以及 Visual Basic for Applications，即 VBA)、C++、C#和 JScript。另外，其他一些公司和学术组织也正在开发面向 CLR 的语

言编译器。

- 简便的代码重用

使用上面所述的机制，我们可以很容易地创建一些类型，来为第三方应用程序提供服务。.NET 框架使得代码的重用变得非常简单，同时也为组件(类型)厂商创造了一个巨大的市场。

- 自动化的内存管理(垃圾收集)

程序设计是一项需要大量技巧和规则的活动，尤其在处理诸如文件、内存、屏幕空间、网络连接、数据库等资源的情况下更是如此。在这中间，最常见的一个 bug 就是因忘记释放某些资源而导致不正常的运行行为。为此，CLR 会为我们自动追踪资源的使用情况，从而确保应用程序不致泄漏资源。实际上，在.NET 框架中，我们甚至没有办法显式“释放”内存。本书第 19 章将对 CLR 的垃圾收集原理有详细的解释。

- 坚实的类型安全

CLR 可以确保所有的代码都是类型安全的。类型安全确保了系统所分配的对象总能够以正确的方式被访问。例如，假设一个方法声明的输入参数接受一个 4 字节的数值，那么 CLR 将阻止我们向其传递一个 8 字节的数值。同样，如果一个对象在内存中占用 10 个字节，那么应用程序将不可能把它当成多于 10 个字节的对象来读取。类型安全还意味着应用程序的执行流程只能向已经确知的位置传递(也就是真正的方法入口点)。换句话说，我们不可能构造一个指向某个内存位置的任意引用，然后让应用程序从那个地址开始执行。总而言之，这些确保类型安全的措施减少了很多常见的编程错误和一些典型的系统攻击(例如，利用缓冲区溢出进行的攻击)。

- 丰富的调试支持

许多编程语言都支持 CLR，这使我们可以很容易地采用最合适的语言来做它最擅长的工作。但是调试该怎么办呢？答案是 CLR 完全支持跨语言调试。

- 统一的错误报告

Windows 程序设计中最令人烦恼的一个问题就是报告错误的各种不同方式。例如，一些函数通过返回 Win32 状态码来报告错误，一些函数通过返回 HRESULT 报告错误，而另一些函数则通过抛出异常来报告错误。在 CLR 中，所有失败的调用都是通过异常来报告的。异常使我们能够将恢复代码和真正的应用程序逻辑代码分离开来实现。这种分离可以极大地简化代码的编写、阅读和维护。另外，CLR 中的异常还具有跨模块和跨语言的特性。而且与状态码和 HRESULT 不同的是，异常不能够被忽略。最后，CLR 还提供了内置的栈遍历机制，这使得我们可以很容易地定位任何的 bug 和调用失败。

- 全新的安全策略

传统操作系统的安全机制都是基于用户账号来提供隔离和访问控制的。这种机制虽然很有效，但从其本质上来讲，它假设的是所有代码都具有相同的信任度。当所有的代码都从物理介质(例如 CD-ROM)，或者可信赖的公司网络上安装时，这种假设是正确的。但是随着当今计算平台对可移动代码(如 Web 脚本、从互联网上下载的应用程序以及电子邮件附件)依赖的增加，我们就需要一种以代码为中心的控制方式。CLR 中的代码访问安全(CAS)为我们提供了这种方式的实现机制。

- 强大的互操作能力

微软很清楚，许多开发人员有着无数现存的代码和组件。要重写所有这些代码来利用.NET 框架平台无疑将是一项艰巨的工作，其结果往往会延缓开发人员对.NET 框架平台的接受速度。为此，.NET 框架从一开始就对访问现有 COM 组件，以及调用传统 DLL 中的 Win32 函数提供了完全的支持。

软件用户一般不会直接去欣赏 CLR 及其能力，但是他们很快会注意到采用 CLR 应用程序所具有的品质和特性。另外，采用 CLR 应用程序的开发时间和部署时间都要比原来的 Windows 应用程序快许多，这一点也可以从用户的反馈和公司的营收报表中得出。

集成开发环境：Visual Studio .NET

整个.NET 平台创新的最后一个组成部分就是 Visual Studio .NET。Visual Studio 是微软耕耘多年的开发工具，并且随着.NET 平台的发布，它又引入了许多专门针对.NET 框架而设计的特性。支持 Visual Studio .NET 的操作系统包括 Windows NT 4、Windows 2000、Windows XP、Windows .NET 服务器家族产品，以及 Windows 的后续版本。而 Visual Studio .NET 产生的代码除了可以运行在上述操作系统上之外，还可以运行在 Windows 98、Windows 98 第二版和 Windows Me 上。

与任何一种优秀的开发工具一样，Visual Studio .NET 也包括一个项目管理器、一个源代码编辑器、一个 UI 设计器、许多的向导、编译器、链接器、工具、实用程序、文档，以及调试器。它既支持 32 位和 64 位的 Windows 应用程序，也支持.NET 框架平台的应用程序。Visual Studio .NET 的另一个重要改进是对于所有的编程语言，它现在只有一个集成开发环境。

另外，微软还提供了一个.NET 框架 SDK。该 SDK 是免费的，它包括所有的语言编译器，以及很多工具和大量的文档。我们也可以不使用 Visual Studio .NET，而直接使用该 SDK 开发出面向.NET 框架的应用程序。当然，这时候我们必须使用自己的代码编辑器和项目管理工具。我们也可能会因此而失去 Visual Studio .NET 为 Web 窗体和 Windows 窗体提供的便捷的拖拉式设计支持。就个人而言，我经常使用 Visual Studio .NET，因此我在本书中多次引用到了 Visual Studio .NET。但 Visual Studio .NET 对于学习、使用和理解本书的所有概念并非必须，因为本书的主旨在于程序设计，而非开发工具。

本书目标

本书的目标是解释如何开发面向.NET 框架的应用程序。具体而言，本书将解释 CLR 的工作机制，以及它提供的各种构造。本书还会讨论到 FCL 中的各个部分。当然，没有哪本书能够完全解释 FCL——它包括的类型有数千种，并且这个数目还在以惊人的速度增长。因此，本书仅将重点放在每个.NET 开发人员都需要理解的 FCL 核心类型上。需要提醒大家的是本书不会讲述 Windows 窗体、XML Web 服务、Web 窗体等这些具体的应用程序模型，本书内容适合于所有这些应用程序模型。

本书不会向大家讲述 Visual Basic .NET 编程语言，也不会假定读者是一位 Visual Basic 6 的开发老手，虽然某些情况下本书会对 Visual Basic .NET 和 Visual Basic 的早期版

本进行一些比较。但是本书假定大家熟悉面向对象的一些概念，比如数据抽象、继承和多态。对这些概念的理解对于掌握.NET 框架程序设计来说是至关重要的，因为所有.NET 框架的特性都是通过面向对象的范式来提供的。如果对这些概念还比较陌生，笔者建议大家先找一本这方面的书来看一看。

虽然本书的目的不是讲述基本的编程技巧，但是我仍会在那些与.NET 框架密切相关的主题有所着墨。所有的.NET 框架开发人员都要理解这些主题，书中不仅会讲解它们，还会在很多地方用到它们。

为了演示.NET 框架的工作原理，本书提供了许多代码示例。显然，最佳的语言选择应该是 IL 汇编语言。因为 IL 是 CLR 惟一理解的编程语言。所有其他的语言编译器都是先将源代码转换为 IL，然后再交由 CLR 处理。使用 IL，我们可以访问 CLR 提供的所有特性。

然而，IL 汇编语言是一种非常低级的语言，用来演示程序设计的概念有时候并不合适。在 Applied Microsoft .NET Framework Programming(微软出版社，2002)一书中，我选择了 C#语言来演示所有的例子。选择 C#是因为它是微软专门设计用来开发.NET 框架应用程序的语言。但是在 Applied Microsoft .NET Framework Programming 出版之后，很多 Visual Basic 开发人员来信告诉我，他们非常希望从 Visual Basic 的角度来理解 CLR 及其能力。由于 CLR 的行为是独立于编程语言的，所以我觉得将 Applied Microsoft .NET Framework Programming 采用 Visual Basic .NET 语言来描述应该是一件简单的事。但问题是我并不熟悉 Visual Basic .NET，并且对 Visual Basic 6 以及 Visual Basic 开发人员转型到.NET 框架开发平台上遇到的问题也知之甚少。于是我做了一个聪明的决定，我联系了我的朋友，同时也是 Wintellectual 的同事 Francesco Balena。Francesco 有着多年的 Visual Basic 经验，并且他在 Visual Basic .NET 上也颇有建树。他愉快地接受了我的邀请，对 Applied Microsoft .NET Framework Programming 的每一章按照 Visual Basic .NET 开发人员的角度进行了重新编辑。同时，他还添加了一些与 Visual Basic 6 的比较，以及 Visual Basic .NET 提供的一些独有特性。在复审 Francesco 工作的过程中，笔者学到了很多有关 Visual Basic .NET 的知识，对 Visual Basic .NET 也有了新的认识。

许多开发人员都认为 Visual Basic .NET 在语言功能上只不过是 C#的一个子集。事实上，这种看法并不正确。比如，Visual Basic .NET 提供有含参属性，异常筛选器，更灵活的接口方法实现，方法静态变量，可选参数，默认参数，以及更方便的事件创建与处理，等等。所有这些主题在本书中都有详细的讨论。

另外，书中偶尔会谈到一些 Visual Basic .NET 没有提供的 CLR 特性。对于这些特性(比如运算符重载，非安全代码)，Francesco 和我都同意继续保留在本书中，即使 Visual Basic .NET 程序员不能使用它们。为了解释或者演示这些特性，本书使用了 C#或者托管扩展 C++来描述它们。这样，大家将能够全面地理解 CLR 的能力。如果今后碰到某些情况必须使用 CLR 的某个特性，而 Visual Basic .NET 又没有提供，那么大家就可以用另外一种不同的语言来实现。而且 Visual Basic .NET 也在不断的发展过程中，将来的版本也许会支持更多的特性。总之，无论使用何种语言，理解 CLR 的能力对于开发面向.NET 框架的应用程序都将是至关重要的。

系 统 要 求

.NET 框架(译注：指仅支持.NET 应用程序运行的.NET 框架分发包)可以安装在 Windows 98、Windows 98 第二版、Windows Me、Windows NT 4(所有版本)、Windows 2000(所有版本)、Windows XP(所有版本)，以及 Windows .NET 服务器家族产品上。大家可以到 *http://go.microsoft.com/fwlink/?LinkId=5584* 上下载它。

.NET 框架 SDK 和 Visual Studio .NET 可以安装在 Windows NT 4(所有版本)、Windows 2000(所有版本)、Windows XP(所有版本)，以及 Windows .NET 服务器家族产品上。大家可以到 *http://go.microsoft.com/fwlink/?LinkId=77* 上下载.NET 框架 SDK。当然，Visual Studio .NET 必须购买。

另外，大家还可以到 http://www.Wintellect.com 上下载本书的源代码。

完 美 无 瑕

这个标题清楚地表达了我对本书的期望。但是大家都知道这是个真实的谎言。尽管如此，我和我的编辑仍然将“深度及时、通俗易懂和准确无误”确定为本书目标，并为此投入了很多工作。然而即便有一流的团队，事情总难免会出现纰漏。如果大家发现了书中任何的错误(尤其是 bug)，并能通过 *http://www.Wintellect.com* 发送给我，我将非常感谢。

相 关 支 持

为确保本书的准确无误，各方都付出了很多努力。微软出版社在下面的 Web 站点提供有图书的勘误服务。

http://www.microsoft.com/mspress/support/

大家也可以直接到微软出版社的知识库上查询相关问题，知识库的地址为：

http://www.microsoft.com/mspress/support/search.asp

如果大家对本书有任何评论、问题，或者建议，也可以通过下面的方法联系微软出版社：

邮政信箱：

Microsoft Press

Attn: Applied Microsoft .NET Framework Programming Editor

One Microsoft Way

Redmond, WA 98052-6399

电子信箱：

MSPINPUT@MICROSOFT.COM

另外，请注意上述邮件地址并不提供微软的产品支持。如果要获得 C#，Visual Studio，或者.NET 框架的支持信息，可以访问微软的产品标准支持站点：

http://support.microsoft.com

目　录