微软.NET程序员系列

Wintellect
Raise Your Windows IQ

- 汇集.NET平台思想之大成
- 雄踞亚马逊销售榜首14个月
- .NET核心开发组成员鼎力推荐
- .NET领域当之无愧的圣经教本

Microsoft® .NET 框架程序设计(修订版)

Applied Microsoft .NET Framework Programming

(美) Jeffrey Richter 著
李建忠 译

Microsoft .net

清华大学出版社

微软.NET 程序员系列

Microsoft .NET 框架程序设计(修订版)

(美) Jeffrey Richter 著
李建忠 译

清华大学出版社
北 京

内 容 简 介

本书是《微软.NET 程序员系列》丛书之一，主要介绍如何开发面向 Microsoft .NET 框架的各种应用程序。Microsoft .NET 框架是微软公司推出的新平台，包含通用语言运行时(CLR)和.NET 框架类库(FCL)。本书将深入解释 CLR 的工作机制及其提供的各种构造，同时还将讨论 FCL 中一些重要的类型。全书共分为五个部分，包括：.NET 框架基本原理、类型和通用语言运行时、类型设计、基本类型，以及类型管理。

本书适用于要了解、掌握.NET 平台的读者，尤其适合广大编程爱好者、软件工程师、系统架构师阅读。

Microsoft .NET 框架程序设计(修订版)
Applied Microsoft .NET Framework Programming
Jeffrey Richter

图书在版编目(CIP)数据

Microsoft .NET 框架程序设计(修订版)/(美)里克特(Richter,J.)著；李建忠译.—北京：清华大学出版社，2003
(微软.NET 程序员系列)

书名原文：Applied Microsoft .NET Framework Programming

ISBN 7-302-07509-3

Ⅰ.M… Ⅱ.①里…②李… Ⅲ.计算机网络—程序设计 Ⅳ.TP393

中国版本图书馆 CIP 数据核字(2003)第 097582 号

出 版 者：清华大学出版社 **地 址**：北京清华大学学研大厦
http://www.tup.com.cn **邮 编**：100084
社 总 机：010-62770175 **客户服务**：010-62776969

译 者：李建忠
文稿编辑：杨志娟
封面设计：陈刘源
印 刷 者：北京市世界知识印刷厂
装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开 本：185×230 **印张**：38.75 **插页**：5 **字数**：933 千字
版 次：2003 年 11 月第 1 版 2004 年 2 月第 2 次印刷
书 号：ISBN 7-302-07509-3/TP·5532
印 数：4001～6000
定 价：68.00 元

探 微 知 著

——译序

.NET 平台推出至今已经整整 3 年了，3 年来.NET 从最初的市场噱头走进了绝大多数开发人员的视野，并成为很大一部分开发人员真实生活的一部分。我相信读者在阅读这篇译序时已经不再有“.NET 技术有什么优势”或者“我是否需要学习.NET 技术”之类的疑问了。然而.NET 技术浩如烟海，从底层的托管运行时，到五花八门的高级编程语言，从巨量的类库 API，到丰富多彩的应用程序模型，各路技术尽数囊括，多种思想和盘托出。从哪里入手学，怎样学？

是的，Jeffrey Richter 先生在本书中给出了这一问题的答案，这就是.NET 底层框架技术。

我一直认为从微观入手、从底层入手是掌握软件技术的不二法门，我发现我的这种心得和 Jeffrey 先生在本书中的技术思路不谋而合。这也是我在翻译这部名著的高强度劳动中得以保持快乐的一个重要原因。在.NET 平台中，.NET 框架占据着核心的位置，它是整个.NET 平台的关键支撑，是为众多高级语言(如 C#、Managed C++、Visual Basic .NET 等)和应用程序模型(如 Windows 窗体、ASP.NET Web 窗体、XML Web 服务等)提供各种服务的重要基石。脱离.NET 框架来谈.NET 平台，难免不陷入空中楼阁的尴尬境地。实际上，在.NET 平台中，任何一门编程语言提供的功能都只是.NET 框架下一个子集的映射，具体的语言已经退居为一个语法表达的层次了。所以虽然本书的描述语言为 C#，但是任何.NET 语言下的开发人员都可以通过阅读本书来获得教益。除了编程语言外，各种类型的.NET 应用程序在设计、开发、测试、部署、运行等诸多环节也和.NET 框架密切相关。虽然本书并不涉及这些具体.NET 应用程序模型的细节，但是如果没有对.NET 框架的深刻把握，学习再多的.NET 应用程序模型“开发技巧”都将只是徒劳——皮之不存，毛将焉附？因此，不管是学习 Windows 窗体、还是 ASP.NET Web 窗体、抑或是 XML Web 服务，我建议大家首先从.NET 框架开始迈出坚实的一步——探微而知著，这也是 Jeffrey 先生写作本书的初衷。

本书融合了 Jeffrey Richter 先生十几年的开发、顾问经验，对整个.NET 框架技术进行了一次全面的检阅和酣畅淋漓的剖析。全书采用 Jeffrey 先生惯用的“于细微处入手，在平实中参透”的技术文笔，将.NET 框架中的各个技术要点一一呈现给读者。尤其是书中对程序集、元数据、值类型/引用类型(装箱/拆箱)、异常处理、垃圾收集等这些.NET 核心技术的讲解，运思精深，鞭辟入里。对这些主题的掌握是理解整个.NET 框架的关键，也是构建各种高效应用程序的必经路径。在对这些大部头的技术主题进行深入剖析的同时，书中还对读者在.NET 应用程序开发实践中遇到的各种问题给予了很

多有益的忠告和指导。此外，书中也不乏大量经典的可重用代码范例，如 Dispose 资源管理模式、事件设计模式、Equals 的重写实现、对象池的设计等，这些代码像珍珠一样遍布在本书的各个角落。实际上，笔者在日常的开发实践中就重用了其中许多代码。虽然本书是对.NET 框架技术的一次全面的剖析，但它并不是对.NET 框架技术事实的简单堆砌，而是带领读者去探索、去领悟、去构筑一个关于.NET 平台核心技术的思想体系。书中每一章内容都值得读者反复阅读、细细品味，相信每一次重新阅读都会带给读者对.NET 平台更为深刻的理解。

虽然本书涵盖了大量艰深的技术，但是阅读起来却并不困难，这中间除了 Jeffrey 先生优美的技术文笔外，大量短小精悍的示例代码也功不可没。除非读者是久经沙场的软件开发老手，阅读本书只是出于消遣或者对 Jeffrey 先生的一份尊敬，否则笔者强烈建议大家在阅读的同时多通过动手演练这些代码来领悟这本名著的精髓。在阅读本书和进行代码演练时，建议大家到微软 MSDN 网站上下载一份.NET 框架 SDK，同时准备一个源代码编辑器(这方面有 UltraEdit、SharpDevelop、Visual Studio .NET，以及 C# Builder 等)。在.NET 框架 SDK 中，除了最常用的工具 C#编译器(CSC.exe)外，建议读者把 IL 反汇编工具(ildasm.exe)也放在手边——实际上 Jeffrey 先生在本书中就有一个论断“如果你对 IL 反汇编工具显示的内容了解得越多，那么你对.NET 框架的理解也会越深”。另外，读者也要多参阅.NET 框架 SDK 中附带的类库文档。在源代码编辑器方面，如果读者使用的是比较高级的 IDE(比如 Visual Studio .NET)，建议读者刚开始学习的时候避免使用其中的向导、自动代码生成等功能，而应该只把它们作为源代码编辑器来使用。因为不只我一个人发现刚开始学习程序设计就使用大型豪华 IDE 的高级功能往往会把学习者引入一个“神秘的、混沌的、并且不知所措的”状态中，除了打击读者的信心和积极性外，对学习掌握程序设计没有任何帮助。实际上从最简单的源代码编辑开始的学习者到后来往往最容易驾驭这些高级 IDE。工具的学习也有一个探微知著的过程。

再来简单介绍一下本书的作者 Jeffrey Richter 先生。我相信阅读这篇译序的很多读者都比我更了解 Jeffrey 先生，以及他在.NET 技术上所做的令人尊敬的贡献。Jeffrey Richter 先生早在 1999 年.NET 平台的整个开发工作正在进行时便受邀和微软一线的研发人员在一起讨论各种.NET 技术问题，开展.NET 技术的咨询工作。本书就诞生于这个过程。因此如果读者在本书中看到 Jeffrey 先生对.NET 框架中大量技术要点来龙去脉的精辟讲解，各个优缺点的犀利分析，以及各种开发实践的忠告指导时，请不要感到任何的惊奇和突然——Jeffrey 先生的确具有.NET 技术的半官方背景。

我相信对于.NET 领域应该很快会有这样的说法——.NET 程序员将会因为此书而分为两类，一类是读过《Applied Microsoft .NET Framework Programming》的，一类是没有读过《Applied Microsoft .NET Framework Programming》的。

下面简单交代一下译本的几个问题。大家知道.NET 中引入了许多新的术语，很多术语至今在开发界还不能做到耳熟能详，甚至时有混乱。因此，除了译本中附上的术语表外，书中正文在刚开始遇到这些术语时，一般都采用了中英并陈的方式，对于有些读者可能感到比较生疏的术语，还在多个地

方进行了中英并陈，目的无非只有一个——方便读者阅读和理解。再一个问题是勘误。Jeffrey 先生为本书的英文原版在微软出版社的网站上维护有一个勘误表，本书定稿时的最新勘误日期为 2003 年 5 月 30 日，这些勘误直接在译本中进行了修改。所以如果读者进行中英文对照阅读，会发现个别地方讲的不一致。这时读者首先应该去查阅英文版的勘误表。对于那些笔者发现的、但未在原书勘误中列出的错误(其中绝大多数都和 Jeffrey 先生进行了讨论)，笔者采用添加译注的方式进行了说明。译本在出版之后也会长期维护一个勘误表。最后，本书采用了页页对译，书后的索引完全来自本书英文原版。但是由于有些地方添加了 Jeffrey 先生的勘误和笔者的译注，这使得页页对译很难在每一页中都得到严格的保证。因此读者偶尔会发现索引位置不匹配的问题，这时读者应该在索引页码的前后页查找。希望译本的这些做法能够提升读者的阅读体验。

最后，我要感谢所有对本书的翻译给予过帮助和支持的朋友。我首先要感谢本书的原作者 Jeffrey Richter 先生，除了为本书带来大量的精彩华章外，Jeffrey 先生对这本中译本也贡献颇多。在本书的翻译过程中，Jeffrey 先生对我提出的大量不管是琐碎的、还是艰深的问题都给予了耐心的解答，这是这本书的翻译能够成功完成的重要保证。书中很多重要的译注都是我和 Jeffrey 先生交流切磋的结果，如果它们能够给读者的阅读和对.NET 框架的理解带来帮助的话，请把这些功劳记在 Jeffrey 先生的名下。我还要特别感谢清华大学出版社的章忆文女士，是她促成了这次合作，读者才有机会看到这个译本。还有清华大学出版社的杨志娟女士等各位编辑，她们在这部名著的翻译、校对、排版、制图等各个环节上做了大量的工作，读者阅读本书时得到的各种美好体验与享受一定有她们的很多功劳。还有在本书翻译过程中给过我帮助、鼓励与支持的很多朋友和网友，请原谅我不能在这篇短序中一一列出他们的名字，我相信一个好的译本是对大家最好的答谢。

我真诚地希望为这本书付出辛勤劳动的各方没有辜负大家的期望，希望各位读者朋友在美好的阅读享受之中能对.NET 框架技术有一个彻底的、通透的理解。如果你阅读完本书后告诉我你对整个.NET 框架技术有了一种豁然开朗的感觉，并大大提高了你的.NET 应用程序设计和开发能力，甚至因此改变了你的程序生涯——这一点儿都不奇怪，这也是 Jeffrey 先生所有著作的特点——我将会感到十分的荣幸和愉快！如果你发现译本中有任何问题或不满意的地方，请不要吝啬你的 CPU 和 Memory 给我发信，我会非常感谢读者任何的批评与反馈，并及时做出更正与改进。谢谢！

李建忠 2003/07 于上海浦东

jzli@china.com(个人电子邮箱，欢迎读者的批评等阅读反馈)

www.lijianzhong.com(个人网站，提供本书勘误、代码、FAQ 等支持)

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，我喜欢通过互联网来购物、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，我想在我的周围找一家有着特色风味的餐馆。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。在这之中，许多服务将会是免费的，有一些是可以通过订购计划获得的，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、客房或者一辆出租车
- 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(例如我们自己的)也将会提供这样的服务，其中一些还可能在自由市场中与微软发生竞争。

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丰富的特色应用(基于 HTML 或者其他技术)呢？举个例子，如果餐馆提供了他

们的菜单访问服务，我们就应该能够写出一个应用程序，它可以查询每个餐馆的菜单，搜索某一特定的口味或菜肴，然后向用户推荐那些离他们最近的餐馆来。

> **注意**　为了创建这样一些丰富的应用程序，企业必须提供针对他们的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。这就是整个 Microsoft .NET 平台创新的主旨。简单地讲，.NET 平台创新就是关于人、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、数字相机以及扫描仪等)和它们相连。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后，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的世界里，服务(或者外设)将和互联网络相连接。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。图 0.1 展示了它们之间的一个类比。在左边，从开发人员来看，Windows 是一个抽象了硬件外设的开发平台。在右边，从开发人员来看，Microsoft .NET 框架则是一个抽象了 XML Web 服务的开发平台。

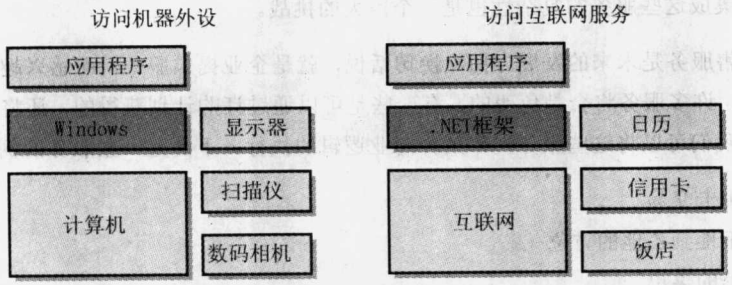

图 0.1

虽然是一个在相关标准的开发和定义方面的领导者——正是这些标准使得这样的新世界成为可能，但微软并不拥有其中任何一个标准。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(典型地利用 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)在这里用来描述通过 HTTP 协议发送的特殊格式的 XML。

图 0.2 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和 Web 服务，甚至其他客户端(通过 SOAP 或者 XML)进行通信的情形。图中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(包括和一些 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。

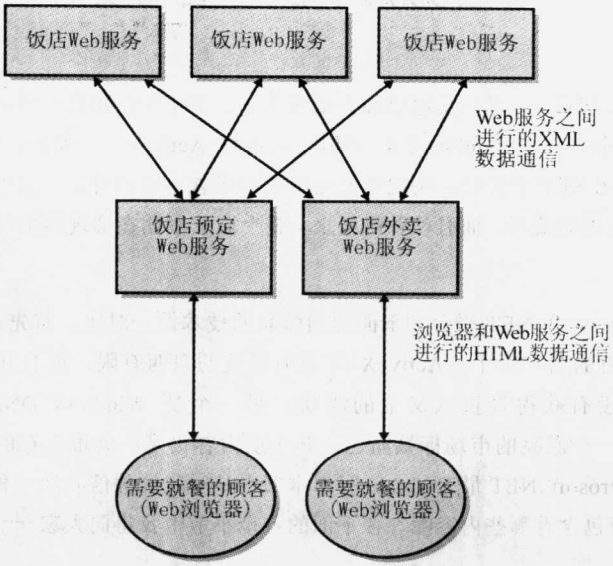

图 0.2

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、寻呼机、汽车、微波炉、电冰箱、手表、音响设备、游戏控制台，以及各种其他的设备都将能够参与到 Web 服务的新世界中。

在客户端，操作系统必须能够在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。如果用户想创建一个窗口、或者一个菜单，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，Windows 只是一个推荐，而非必须。

不管微软是否参与其中，充满 Web 服务的全新世界都终将到来。微软整个.NET 平台创新的目的就是帮助开发人员来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。我们可以直接把我们的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

我和微软以及他们的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。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也是个个雄心壮志。

我们不妨来将 Microsoft .NET 平台和下面几项微软的技术做一对比。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。和前面两项技术相比，我显然对 Microsoft .NET 最有信心。写作本书也是对我这种信心的一种佐证。那么，到底整个 Microsoft .NET 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 产品线都添加了 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、可扩展的即时消息通知应用程序。该应用程序允许第三方厂商(如 Expedia、United States Postal Service 等)和它们的用户进行无缝的通信。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。

Microsoft XML Web 服务：.NET My Services

微软当然不会满足于仅仅作为一个 Web 服务的底层技术提供商，他们也希望能在 Web 服务领域内大玩一把。自然，他们也会创建自己的 XML Web 服务：其中有些可能是免费的，但有些可能就要收费。微软的初始计划提供以下一些.NET My Services：

- .NET Alerts
- .NET ApplicationSettings
- .NET Calendar
- .NET Categories
- .NET Contacts
- .NET Devices
- .NET Documents
- .NET FavoriteWebSites
- .NET Inbox
- .NET Lists
- .NET Locations
- .NET Presence

- .NET Profile
- .NET Services
- .NET Wallet

这些面向消费者的 XML Web 服务被微软称作“.NET My Services”。大家可以到 *http://www.Microsoft.com/MyServices/* 站点上获取有关该服务的更多信息。随着时间的推移，微软不仅会创建更多面向消费者的 Web 服务，还会创建一些面向商业用户的 Web 服务。除了这些向外公开的 Web 服务外，微软也会创建一些供内部销售、付账等使用的 Web 服务。当然只有微软的员工才可以访问这些内部服务。我预料 Web 服务首先会在许多公司内部的网络中流行起来。而面向整个互联网公众的 Web 服务和使用这些 Web 服务的应用程序则可能进展要稍微慢一些。

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#(音作“C sharp”)、.NET 框架。

重要 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，以及独立的组件模块。而本书所述的内容适用于上述任何类型的应用程序。

.NET 框架包含两个部分：通用语言运行时(CLR)和.NET 框架类库(FCL)。.NET 框架本身又是.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 还只能运行于各种版本的 Windows 平台上，包括 Windows 98、Window 98 第 2 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 服务器家族的名称来由。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。微软已经创建了一个免费的.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，因为我们不必再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任何以前的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 的其中一个目的就是简化 Win32 和 COM 环境下所需要的各种繁杂的基础构造。在 CLR 下，我们将可以从此远离如下这些概念：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 框架应用程序要和现存的一些非.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、注册表设置以及快捷链接。另外，要完全卸载一个应用程序更是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 框架应用程序的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、桌面以及【快速启动】栏而已。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)。只有到了运行时，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、PowerPC 等。当用户改变他们的硬件或者操作系统时，他们便会看到这样的技术带来的价值。

- 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的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，然后再在 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)则描述了一个语言要和其他的语言很好地集成在一起所必须要遵循的规范。微软自己已经提供了几个面向 CLR 的语言编译器：托管扩展 C++、C#、Visual Basic .NET(包括 Visual Basic 脚本即 VBScript，以及 Visual Basic for Applications 即 VBA)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的不正常的运行行为。CLR 为此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 将会阻止我们向其传递一个 8 字节的数值。类似地，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，然后便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，这使得我们可以很容易采用最合适的语言来做它最擅长的工作。但是调试怎么办呢？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各种不同的报告错误的方式。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 来报告错误，而还有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。异常使得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。而且和状态码与 HRESULT 不同的是，异常不能够被忽略。最后，CLR 还提供了内置的堆栈遍历机制，这使得我们可以很容易定位任何的 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，它假设的是所有的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)的依赖的增加，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

  微软很清楚很多开发人员都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件巨大的工作，其结果往往会阻止开发人员对.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，但是他们很快将会注意到采用 CLR 的应用程序所

具有的品质和特性。另外，采用 CLR 的应用程序的开发时间和部署时间都要比原来 Windows 下的应用程序快许多，这一点也可以从用户的反馈和公司的营收报表中看得出来。

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 2 版以及 Windows Me 上。

和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。它既支持创建面向 32 位和 64 位的 Windows 应用程序，也支持创建面向.NET 框架平台的应用程序。Visual Studio .NET 另一个重要的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 而直接利用该 SDK 来开发出面向.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。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。但是 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，本书仅将重点放在那些每个.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并不会讲述 Windows 窗体、XML Web 服务、Web 窗体等这些具体的应用程序模型，本书的内容适合于所有这些应用程序模型。

另外，我也不会在本书中讲述任何具体的编程语言，我假设大家至少熟悉一门编程语言，比如 C++、C#、Visual Basic 或者 Java。我还假设大家熟悉面向对象的一些概念，比如数据抽象、继承和多态。对这些概念的坚实理解对于掌握.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，我强烈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，但是我仍会在那些和.NET 框架密切相关的主题上有所着墨。所有的.NET 框架开发人员都要理解这些主题，本章不仅会讲解它们，还会在很多地方用到它们。

最后，虽然本书由于讲述的是.NET 框架中的通用语言运行时而不涉及具体的编程语言，但是为了演示其中的工作机理，我还必须提供许多代码示例。为了保持一个编程语言不可知论者(agnostic)的立场(译注：编程语言不可知论者认为编程语言并不是程序设计的全部，但又不否认编程语言的重要性)，我想最佳的选择就是 IL 汇编语言。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 提供的任何特性。

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。因此我决定使用 C#作为我在本书中主要的编程语言。选择 C#是因为它是微软设计用来专门开发面向.NET 框架的编程语言。如果大家使用的不是 C#语言，那也没什么大碍——只要能读懂演示代码就行了。

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 2 版、Windows Me、Windows NT 4、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 就必须购买。

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”确定为本书的目标，并为此投入了很多精力。然而既便有着一流的团队，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。