HZ BOOKS

微软公司核心技术书库

Microsoft Press

Introducing Microsoft .NET

Second Edition

Microsoft .NET 精髓

（美） David S. Platt 著

黄慧萍 芦阳 等译

Microsoft .net

机械工业出版社
China Machine Press

微软公司核心技术书库

Microsoft Press

Introducing Microsoft .NET

Second Edition

Microsoft .NET 精髓

(美) David S. Platt 著

黄慧萍 芦阳 等译

Microsoft .net

机械工业出版社
China Machine Press

本书介绍微软最新的开发平台.NET。在介绍.NET的基础知识之上，还介绍了一些.NET的高级功能，如自动内存管理、线程以及更方便地访问所有系统的服务等功能。本书的代码充满趣味，而又极具参考价值；同时，本书的幽默写作风格，使本书更易于理解，更具可读性。

本书适合所有想了解.NET的技术人员、技术管理人员、学生、教师参考。

David S. Platt: Introducing Microsoft .NET, Second Edition (ISBN 0-7356-1571-3).

Copyright © 2002 by David S. Platt.

Original English language edition copyright © 2002 by Microsoft Corporation; Published by arrangement with the original publisher, Microsoft Press, a division of Microsoft Corporation, Redmond, Washington, U.S.A.

All rights reserved.

本书中文简体字版由美国微软出版社授权机械工业出版社出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

本书版权登记号：图字：01-2002-6557

图书在版编目（CIP）数据

Microsoft .NET精髓 /（美）普拉特（Platt, D. S.）著；黄慧萍等译. - 北京：机械工业出版社，2003.9

（微软公司核心技术书库）

书名原文：Introducing Microsoft .NET, Second Edition

ISBN 7-111-11383-7

Ⅰ.M… Ⅱ.①普… ②黄… Ⅲ.计算机网络 - 程序设计 - 教材 Ⅳ.TP393

中国版本图书馆CIP数据核字 (2002) 第100628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：武恩玉

北京昌平奔腾印刷厂印刷 · 新华书店北京发行所发行

2003年9月第1版第1次印刷

787mm × 1092mm 1/16 · 16.75印张

印数：0 001-4 000册

定价：35.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线电话：(010) 68326294

译 者 序

微软的.NET是一个运行在Windows 2000操作系统下的附加运行环境，未来的.NET版本极有可能成为操作系统的一部分。.NET框架是一种运行环境，在此环境下，程序员能够更容易地迅速写出优良、健壮的程序代码，并且能够方便地管理、部署和修改代码。你所编写的程序和组件都可以在该环境中执行。它为程序员提供了一些新奇的功能，例如自动内存管理（垃圾收集），以及更方便地访问所有系统的服务。它添加了许多实用功能，例如易于访问因特网和数据库。它还为代码复用提供了一种新的机制，并且比COM更加有效和灵活。.NET框架更易于部署，因为它不需要进行注册表设置。它还为版本的制定提供了标准化、系统级别的支持。程序员可以在任何一种与.NET兼容的编程语言中使用上述全部功能。

本书是一本提出思想的书。书中的代码对.NET主题进行高层次且易于理解的介绍。这本书富有很多有趣的代码，因而显得充满趣味，同时又极具参考价值。不过本书的目的还是在于介绍.NET的主要思想。你需要以一种崭新的方式来思考.NET，才能掌握.NET的神奇魔力所在。

我们很荣幸能够有机会承担本书的翻译工作。在翻译过程中，我们经常为一句话、一个术语进行反复的讨论，到处查找资料，力图使本书的翻译能正确、贴切地反映原文的意思，同时让句子、段落符合中国人的语言习惯。机械工业出版社编辑具有高度的职业精神，在许多方面对翻译工作进行规范，是本书翻译工作得以顺利完成的另外一个关键因素。我们衷心地希望你能够从本书中有所收获！

本书由黄慧萍、芦阳等主译。具体参加本书翻译、录排、校对工作的人员还有：龚克、田蕴哲、牛志奇、李红玲、白红利等。龚超、龚建同志对翻译稿进行了严格细致的复审。

由于时间仓促，且译者经验和水平有限，译文难免有不妥之处，我们殷切地期望你能给我们提出中肯、尖刻的意见，以便于提高翻译水平，把更好的图书呈现给大家！

译 者

2003年8月

前　言

我时常在想，这个被称为Microsoft .NET的产品听起来真的很酷。我还记得曾经阅读过Mary Kirtland发表在《Microsoft Systems Journal》杂志的1997年11月和12月两期刊物上的介绍COM+的文章，当时他指出COM+是能够为程序员提供各种各样有用服务的运行环境，例如，它可以提供跨语言继承和运行时访问控制的功能。作为使用COM的程序员，我非常喜欢COM+这种新的运行环境，因为它承诺可以解决我在COM中遇到的许多问题。

接着，微软决定将下一个版本的微软事务服务器命名为COM+ 1.0，而且还有可能将其集成到Windows 2000中，这也就是Mary Kirtland曾经说起的COM+ 2.0版。再后来，微软又将COM+重命名为Microsoft .NET，于是我自己生造了一个词MINFU来形容微软这种混乱的命名规则。不过，名称的混乱并不影响产品本身的优秀，它听起来的确是个很好的产品，所以当微软出版社问我是否愿意再写一本关于.NET的书籍时，我真的很激动，他们要求我能够像在《Understanding COM+》（微软出版社1999年出版）一书中介绍COM+ 1.0那样出色地在新书中介绍.NET。你手上拿着的，并且是我希望你购买的这本书就是我的成果。

我曾经担心微软不让我以自己的方式来撰写这本书，而是给我一些限制的条条框框。不过，好在这种担心在本书的写作当中并没有发生。我在本书中所讲述的任何内容，无论你赞同也好反对也好，代表的仅仅是我个人的观点。不过有一点是很明显的，即我喜欢.NET并且认为它能够给用户带来巨大的经济效益，微软也同意我的观点。当老板要你为自己找几个推荐人的时候，你会找一个认为你很优秀的人呢，还是找一个认为你比较平庸的人做推荐人呢？我所认识的大部分程序员都会在这两种类型的推荐人中各找几位。在本书的草稿完成之时，我曾收到了一些内部反映意见，有一位项目经理对于我在书中对.NET管理工具提出的批评表示了异议，并且写信给我说："在我看来，你对于管理工具的这些意见使得本书更像是在作评论，而不是在讲授.NET的知识，这是你的初衷吗？"我回信说："我对于本书既指出了.NET的优点又指出了其不足而感到非常自豪，在我的书中，既谈到了.NET带给我们的种种益处，也提及了我们因此而付出的代价。如果我只谈优点，就显得不够诚实了。恕我直言，我觉得这是在说明.NET的特性。如果你认为我不应该在书中一直使用'在我看来'这样的词语，那么请原谅我的用词不当。"

我不能忍受干巴巴的阅读，就像我无法容忍吃干肉或干鱼一样。我还记得在大学第一年的生活，那时我为了给自己的化学实验报告增添几分生趣，就加进了几个笑话，谁知这却给我带来了不及格的惨痛打击。"科学容不得任何浮躁。"我的教授说，"重新做一次实验报告。并且通篇都要使用书面语。"他是我遇见过的惟一一个始终小心翼翼地把胡子修理成眉毛型的人。或许我的教授希望自己变得无趣，以使他的生命看起来更长久一些，但是他不应该因此而剥夺其他人的生活乐趣呀！

对于我来说，最好的作者就是最会讲故事的人，尤其是在科学和历史这样比较枯燥的领域里，更是如此。例如，我非常喜欢Laurie Garrett写的《The Coming Plague》（Penguin，1995）和

此为试读，需要完整PDF请访问：www.ertongbook.com

Evan S. Connell写的《Son of the Morning Star》(North Point, 1997)这两本书，以及William Manchester写的题为《The Last Lion》(Little Brown, 1983)的邱吉尔传记。再想一想你们的大学课本，由那些像我以前的教授一样的人编写的课本教材。哪一种书我们更不愿去读？下面再让我们来读一段文字，它讲述的是在1894年巴黎爆发一场大规模疾病时，Emile Roux发现青霉素以及首次用于人类实验的一段经历，其原文如下：

Roux looked at the helpless doctors, then at the little lead-colored faces and the hands that picked and clutched at the edges of the covers, the bodies twisting to get a little breath....

Roux looked at his syringes—did this serum really save life?

"Yes!" shouted Emile Roux, the human being.

"I don't know—let us make an experiment," whispered Emile Roux, the searcher for truth.

"But, to make an experiment, you will have to withhold the serum from at least half of these children—you may not do that." So said Emile Roux, the man with a heart, and all voices of all despairing parents were joined to the pleading voice of this Emile Roux.

你还可以在1926年首次由Paul de Kruif出版的《Microbe Hunters》一书中找到Roux的选择及结果，该书几十年来再版过好几次，最近的一次是由Harcourt Brace公司在1996年重新出版的。这本书中没有太多学术性的客观论述，可是你更喜欢读哪一种书籍呢？我知道自己应该写哪种形式的书籍。我没有de Kruif那么好的口才，而且我也不大相信当我112岁的时候，还会有人重新出版这本书。但是，我已经尽我所能将这本书写得更通俗易懂，有多少科技作者尝试过这么写作呢？

De Kruif在他的书结尾处写道："This plain history would not be complete if I were not to make a confession, and that is this: that I love these microbe hunters, from old Antony Leeuwenhoek to Paul Ehrlich. Not especially for the discoveries they have made nor for the boons they have brought mankind. No. I love them for the men they are. I say they are, for in my memory every man jack of them lives and will survive until this brain must stop remembering." 正如我在本书的后记中所说的那样（现在跳到那儿不合适；你必须先读完这本书），因特网使人类得到了进化。Microsoft .NET就是要做这方面尝试的一个产品。我觉得自己能被选中向读者介绍.NET，并且以我自己的讲述方式来编写本书实在是一件很荣幸的事情。我之所以放弃了原来的出版社，而选择了微软出版社，就是为了更好地与他们的项目小组进行交流，与他们探讨将来会是怎样的以及为什么会是那样。早期的读者曾经说过我的文字成功地做到了这一点。我的确是做过这方面的努力，希望他们是对的。

任何一本书都是一个集体劳动的结晶，就像一次卫星发射一样，只不过规模要小得多。书的作者就像宇航员一样获得了诸多的荣誉，但是如果没有其他人为本书作出的辛苦劳动，你是无法读到本书的。就像阿波罗项目组中数千个成员一样，该书背后的大部分工作者都不为人们所知。（虽然我觉得在电影《阿波罗13号》中，由Ed Harris 扮演的飞行控制员Gene Kranz抢了Tom Hanks扮演的Jim Lovell的风采。）除非有人把本书拍成电影，否则恐怕没有多少人知道这些

幕后工作者付出的贡献。

这里首先要感谢本书的主编John Pierce先生。在我几年前写《Understanding COM+》（微软出版社出版）一书时，他就是那本书的副主编。我很高兴这次他能成为我这本书的主要负责人。他和我一样很有幽默感。我知道他不会批评我的写作风格，或者改变我的写作语调，不管你喜欢也好讨厌也好，你肯定认为我的写作基调的确是与众不同的。John Pierce帮助我把文字以更好的方式组织起来，这一点要比我自己做得好。

其次还要感谢本书的技术编辑Jean Ross和Marc Young先生。他为我所有的技术问题一一作出了解答，而且往往是在时间紧迫、代码经常变化的情况下为我解答的。除了Jean和Marc以外，还有很多.NET开发项目小组的成员都抽出自己宝贵的时间为我解答疑难。在此我要特别感谢Susan Warren、Keith Ballinger、Mark Boulter、Loren Kohnfelder、Erik Olson、John Rivard、Paul Vick、Jeffrey Richter和Sara Williams等人。

Ben Ryan早在一年前就开始了收集信息工作，在他离开后Danielle Bird又接手了该工作，最后阶段是由Anne Hamilton完成的。微软出版社的产品经理Teresa Fagan一听到《Introducing Microsoft .NET》这个题目，马上就想到了一个好主意，他说："嘿，你可以让它成为一个Web站点。"于是，我立即冒雨冲回我的写作室，在其他人有所动作之前抢先一步实施该计划。它好比是圣代冰淇淋上的草莓。

最后，我要感谢我的妻子Linda，现在她已经是我们可爱的女儿Annabelle的母亲了。Annabelle现在不仅会叫出单个对象（"Simba"）和调用这些对象的方法（"Feed Simba"），而且还知道对象是类的实例（"Simba is a kitty cat"）。

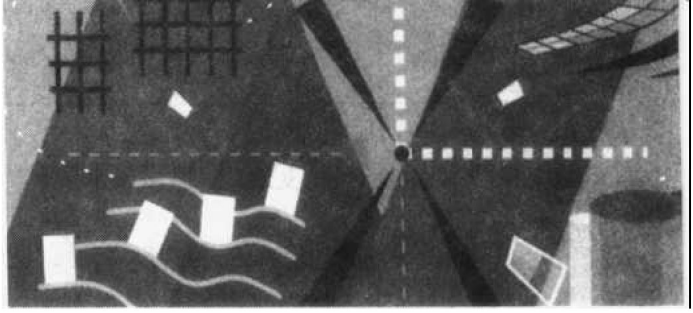
大卫·S·普拉特（David S. Platt）

www.rollthunder.com

美国马萨诸塞州伊普斯威奇

2001年4月和2002年3月

目　录



第1章 引 言

1.1 庞大的因特网

因特网是极其广阔的。（恼怒的读者们会说："我花了这么多钱就为了让你告诉我这个？"作者回答："什么？你说什么？难道我说的不对吗？"）

独立的桌面PC机远不如联网的PC机用处大。

仅仅一台独立的桌面PC机是令人厌倦的，这就像一个单细胞的变形虫一样毫无趣味。当然，你可以使用PC机玩"纸牌游戏"，它不允许你撒谎（这也算是一个优点吗？），而且你还可以很方便地使用附件中的"记事本"。但是，和变形虫具有的革命性价值不同，单独的桌面PC机带给社会的巨大经济利益已经被很好地证明了。但是，如果桌面PC机仅仅局限在自己有限的一个机箱范围内，显然不可能再提供更多有趣和有用的功能。然而，当你将你的PC机通过因特网和世界上的其他PC机，以及其他智能的非PC设备（例如掌上电脑、冰箱等等）相连时，不需要任何其他的硬件花费，就会发现有趣的事情发生了——这就类似于许许多多的单个细胞聚集到一块儿，并进化成人脑一样，人脑不仅能够作曲、演奏和欣赏交响乐；还能够飞向月球；或者消除自身的某些种类。这样奇妙的事情是不是要比玩"纸牌游戏"有趣得多呢？

因特网以前所未有的高速度改变着社会。

Web起初是浏览枯燥的物理报告的一种手段，并从此开始飞速发展（本世纪对其的了解）。Web的出现极大地方便了各种类型数据的发布。因特网日益扩展的内容吸引了更多的用户，同时，日益增加的因特网浏览者又成为了因特网上更多内容的提供者，这样形成的一个良性循环不但没有丝毫要结束的迹象，而且就在我写下这些文字时，它还在继续增长。就在昨天我还使用Web浏览了错过的一场曲棍球比赛的精彩视频片断。接着，我又利用Morpheus从分布于10 000个不同用户的硬盘驱动器上的500 000首歌曲中挑选了我所喜爱的一些音乐（当然，我只查看那些合法的歌曲）。然后，我又通过因特网和我的祖母讨论了一下关于安装摄影机的事情，这样即使远在804.5公里（500英里）之外，她也能看到小外孙的婴儿床了。因特网使得我们的世界完全不同于以前了，甚至与5年前大不一样。

因特网的硬件设备以及带宽费用都变得越来越便宜了。

连接到因特网所需的硬件设备以及传输数据所需的带宽都变得越来越便宜了。只需要几百美元就能够在我的PC机上安装一台Web摄像机，而且再不需要什么其他的花费，就能够通过我现在的电缆调制解调器连接到因特网，让母亲看到我的小女儿了。试想一下在十年前，甚至五年前，需要花多少钱才能买到一台摄像机，并且租赁一条从麻省到宾州的专用连接线路。因特网硬件和带宽的价格将会迅速下降，甚至低于Crack Jack的价格。

1.2 标准提升:常见的基础设施问题

因特网软件提出了一些新的问题，要解决这些问题不仅更加困难，而且花费的代价更大。

尽管硬件和带宽的价格便宜，但是仍然存在着一个障碍。Platt的第二定律就阐述了宏观上骰子的数量是恒定的[⊖]。如果某人的骰子数量较少，那么这不过是因为他已经事先扔了一定数量的骰子了，决不会有什么能够使他的骰子消失。同样的道理，如果硬件和带宽变得越来越便宜，越来越容易得到，那么这就意味着，编写运行在这种环境下的软件就会变得越来越困难，其耗费也越来越高。实际上，这种状况的确也被证实了，任何试图证明这个结论的人，一定会得到肯定的结果。因特网应用程序中存在的问题并不是商业逻辑问题，它和一般的桌面应用一样（一个小于0的数字就意味着你的账户已经透支了）。然而，由于因特网本身的公共性、不可控制性以及不同的特性，当你通过因特网在互连的不同PC机上执行一个应用程序时，就会产生一些新的问题。试想一下在自己的房间内照料一个初学走路的孩子是一件多么容易的事（相对而言），可是要在中心火车站照料一个初学走路的孩子该是多么困难呀。同样的孩子，同样的目标（安全、有趣），然而要求却完全不同。

桌面应用程序一般根本就不存在什么安全性问题。

例如，考虑安全的问题。很多用户都使用Quicken或者类似的产品在自己独立的PC机上保存个人财政记录。早期版本的Quicken开发者们并没有编写任何安全性代码。他们轻松自在，或者更确切地说是他们的用户轻松自在，因为只要将自己的PC机锁好，别人就不能偷去自己的钱。疑心重的用户可以购买一种产品，以便利用密码来保护自己的PC机，但是，几乎没有人这么做。

一旦新鲜劲过后，许多用户就会发现桌面PC机上安装的Quicken并不是那么有用。它并不比原来的支票记录单用处更大——相反使用支票还会比这个愚蠢的程序更快捷、更容易。直到Quicken能够连接到因特网，并且提供了诸如电子账单收据功能、电子支付，以及自动下载银行存款支付报告书和信用卡支付报告书等功能，使得用户能够与其他的财政方进行交易，这时它才带给用户一些网络益处。（也就是说，如果不是因为用户界面如此差劲，那么Quicken将会带给用户许多网络益处。尽管它并没能非常好地解决上述新功能的复杂性问题，以至于同时向用户提供过多模糊的选择。但是这并不属于因特网的问题。）

因特网应用程序需要为运行的各个阶段都提供安全保障。

但是，一旦Quicken的操作在其运行的单机上遗留下了安全隐患，那么这些操作就更需要解决安全性问题。例如，当用户通知电子账单支付中心写一张电子支票给话务公司时，电子账单支付中心就必须确认这个请求来自于该账号的真实主人，而不是来自于一个为了避免破产而企图给自己提前支付几个月话费的话务公司。我们还需要对交易双方之间的数据流加密。你肯定不希望邻家的小孩在你的电缆调制解调器线路上使用小型探测器偷窥到你的账号以及你购买了什么——“花295美元换取Hunky Escort服务？不知道她丈夫知不知道这事儿？”

⊖ Platt的第一定律被称作“指数爆炸”。该定律说的是每个软件工程所花费的时间是当初最佳估计时间的3倍。

此为试读，需要完整PDF请访问：www.ertongbook.com

安全代码相当难以开发。

由于雇员频繁跳槽，所以安全代码的编写、测试、调试、部署、支持和维护都显得相当困难。你必须雇佣对安全性问题有全面了解的人——他们需要知道如何验证用户身份，如何决定是否允许用户的操作，如何加密数据以便经验证的用户能够访问数据，而偷窥者则不能，如何设计工具供管理员使用，使管理员能够设置或删除用户的安全权限，等等。

安全性问题以及分布式计算中存在的其他类似问题都属于普通的基础设施。

因特网计算带来了其他许多类似的问题，我将在后面的章节中一一阐述。正如我在《Understanding COM+》这本书中所解释的那样，这些问题都有一个共同的特征，即它们都与你的商业过程无关，也和你的客户支付给你的东西无关。这些问题只和网络的基础设施有关，就像高速公路系统和电力输送网一样，就像人们的日常生活所基于的基础框架一样。

基础设施，而不是业务逻辑，会导致项目失败。

研发基础设施会导致项目失败。我从来没有见过哪个项目是因为业务逻辑出现问题而失败的。你比任何人都了解业务过程这也是你编写软件来支持它的原因。但是你并不了解基础设施（除非它是你开发的产品，就像微软很清楚自己开发的产品一样）。几乎没有人是安全认证算法或者加密方面的专家。如果你试图自己编写这方面的代码，那么其结果只有两种情况。要么你写出的是一个无用的东西，因为你不知道自己正在做什么（并且你最好希望没有坏小子注意到这件事），要么你努力地尝试写出一个工具，然而耗尽资金也无法将其完成。十二年前，当我在开发一个基于因特网发布的外部交换应用程序的预先版本时（我曾在《Understanding COM+》一书中提及此事），就碰到了上述这两种情况。

1.3 做最好的计划

软件开发者们自欺欺人。

软件开发者们往往在项目开始之初作了最好的计划，并满怀最高的期望。就像一个酒鬼在去聚会的路上那样，我们发誓要通过细致地设计避免错误出现，一定要仔细地检查文档和代码。我们会在开发过程中一直不停地对软件加以测试，并且在代码确定之后不再添加任何其他功能。总之，我们会制定实际的进度表，并且严格地遵守它。（“操作系统的那项功能并不能完全按照预期的那样工作，但是我可以在一周之内写出一个更好的程序，所以没有必要修改我们的程序要求。还有两个星期的时间，除非我会遇到一些困难。我们还需要测试的时间吗？”）一个健壮有效的应用程序就在我们的掌握之中，我们所要做的就是遵守纪律。绝对不能玩“纸牌游戏”。至少在工作结束之前不能。好吧，只在午餐时玩一次游戏。“我输了，再来一次，怎么样？什么时候才能熬到4:00呀。”我所见过的每个项目都是以这样一种充满希望的姿态开始的。

是不是每个开发者都遵守了原先的承诺呢？没有。从来就没有，将来也不可能。我们内心知道其实我们在做出种种允诺的时候就在撒谎。这是一种病。就像吸毒，这样做只有一条出路（不算那些死去的人，几乎每个人都以同样的比率照此方式行事）。为了写出成功的因特网应用程序，软件开发小组的所有人都需要像戒毒的人那样去做，在达到目标之前至少严格地遵守以

下两个步骤：

1) 承认我们是无能的——我们的生活变得难以管理。

2) 相信有一种更强大的力量能够帮助我们恢复健康。

你不能承担自己建造基础设施的巨大花费。

应用程序员必须承认我们对于因特网的基础设施的确无能为力，这使得我们的项目变得难以管理。我们无法建立这个项目，因为它耗时太长，花费太高，我们不知道如何去建立这个项目。我们甚至不能尝试去建立它，因为注定没有好结果的。最不可能的情况就是我们编写出了正确的基础设施代码，于是那些没有代码的竞争者也开始长期使用我们提供的免费资源了。你并不是在建造自己的高速公路供自己的小车行驶，也不是在安装自己的能源生成设备（除非你生活在2001年早期的加利福尼亚州）。

你希望由别人来建造基础设施，而你则只管使用就行了。

值得庆幸的是，你的因特网应用程序的基础设施需求和其他人的要求完全一样，这就像你对高速公路和电力的需求和其他人的需求相似。正是由于这种巨大的普遍需求，政府修建了高速公路，能源公司修建了能源工厂，而你则有偿地使用这些设施，要么直接地支付账单，要么间接地通过纳税来支付使用费。因为政府和公司能够雇用或者培养最优秀的人才来完成预定的目标，而且政府和公司能够将开发费用分摊到更多的项目上，所以他们能够获得大规模的经济效益。

我们真正需要别人来做的工作就是计算出政府为高速公路的修建做了什么（或者准确地说并不是政府为高速公路做了什么，但是你有一个最基本的想法）。就像正在戒毒的人相信只有那种创造世界的能力才能恢复他们的生活一样，开发者们也需要一种更高级的力量来为我们提供因特网的基础设施，帮助我们将开发所付出的努力转化为正确的结果。

1.4 .NET究竟是什么

微软的.NET提供了预制的基础设施，以解决编写因特网软件时遇到的普遍问题。

微软的.NET就是这样一种东西——即为了解决因特网应用中存在的普遍问题而预先建立的基础设施。微软的.NET最近已经得到了相当大的关注。为什么有5000个近乎狂躁的人于2000年7月聚集在佛罗里达州奥兰多市，并且发疯似地欢呼狂叫。这不是因为他们买到了淡季的打折机票而欣喜若狂，也不是因为他们喜欢这种折磨人的高温或者喜欢中暑，而是因为他们首次听说了微软的.NET。

微软的.NET的服务器版可运行在Windows NT、Windows 2000以及Windows XP Professional操作系统下，其客户版可运行于Windows 98、Windows Me、Windows XP Home下。目前它只是一个附加的服务补丁，以后的.NET版本极有可能成为操作系统的一部分。今后的版本也有可能会允许.NET的一部分运行在其他版本的Windows操作系统下，或者，也许我们还会看到.NET运行在其他的操作系统平台下。微软的.NET提供了下述服务，本书将在后面的章节中对其进行介绍。这些服务包括：

- .NET提供了一种新的运行环境，即.NET框架。.NET框架是一种运行环境，它使得程序员

能够更容易、迅速地写出优良、健壮的程序代码，并且能够方便地管理、部署和修改代码。你所编写的程序和组件都在该环境中执行。它为程序员提供了一些新奇的功能，例如自动内存管理（垃圾收集），以及更方便地访问所有系统服务。它添加了许多实用功能，例如易于访问因特网和数据库。它还为代码复用提供了一种新的机制——更易于使用，并且比COM更加有效和灵活。.NET框架更易于部署，因为它不需要进行注册设置。它还为版本的制定提供了标准化、系统级别的支持。程序员可以在任何一种与.NET兼容的编程语言中使用上述全部功能。我在本书的第2章中讨论了.NET框架。

- .NET为创建HTML页面提供了一种新的编程模型，称为ASP.NET。尽管智能的单机程序仍在不断涌现，但是在不久的将来，大多数因特网通信都会以通用浏览器作为前端。这就要求服务器能够使用HTML语言来构造页面，以便浏览器识别并显示给用户。ASP.NET（ASP的下一个版本）是一种运行在因特网信息服务（IIS）下的新环境，它使得程序员能够更容易地编写代码来构造基于HTML语言的Web页面，供浏览器查看。ASP.NET提供了一种新的与语言无关的代码编写方式，并将其与Web页面请求相关连。它提供了.NET的Web窗体，它是一种与控件交互的事件驱动编程模型，这使得编写Web页面变得就像编写普通的Visual Basic窗体一样。ASP.NET包含了良好的会话状态管理和安全功能。它比原来的ASP更加健壮，性能也得以提高。我将在本书的第3章中讨论关于ASP.NET的问题。
- .NET提供了XML Web服务，该服务为因特网服务器程序提供了一种新的方法，使得它们能够将自己的方法显示给任意的客户端程序。尽管通用浏览器会一直保持着其重要的地位，但是我相信将来必定属于专用应用程序和设备的天下。Web会变成一种新的场所，它并不用于在普通浏览器中处理数据，而是供专门的客户程序（例如用来查找音乐的Napster）实现跨越因特网的功能，它们向服务器提出请求并接收数据，将数据显示在一个专用的用户界面中，或者根本不需要用户界面，而是直接在机器之间进行数据通信。微软的.NET提供一组新的服务集，该服务集允许服务器将其功能放置到任何一台机器中的任何一项客户程序上，而且该机器可以运行任意的操作系统。客户程序使用XML和HTTP向服务器发出请求。用这种方式显示的服务集就称为.NET Web服务。这种崭新的设计并不是等待用户去选择使用它，而仿佛是在向人们宣告："购买我们的操作系统吧，因为我们提供了大量预制的支持，能使你更容易地编写出因特网应用程序来与世界上的任何一个人交谈，而不用考虑这些程序在哪儿运行，或者这些程序的功能。"我将在本书的第4章中讨论.NET的Web服务。
- .NET提供了Windows窗体，它是一种使用.NET框架编写各种客户程序的新方法。一个使用XML Web服务的专用客户端应用程序必须提供良好的用户界面。高质量的界面能够提供更佳的用户感受，正如Microsoft Outlook的专用界面要比Hotmail的普通Web用户界面好得多。微软的.NET提供了一种新的软件包，它被称为.NET Windows窗体，这使得程序员能够使用.NET框架轻易地编写出专用的Windows客户应用程序。不妨试想一下，如果在任何一种编程语言里都能使用Visual Basic，那该是一件多么奇妙的事情啊，现在你可以想像得到.NET的Windows窗体该是怎样一种模型了吧。我将在本书的第5章中介绍.NET Windows窗体。
- ADO.NET为.NET框架内的数据库访问提供了良好的支持。几乎所有的因特网编程环境都要涉及到数据库的访问。至少就目前来说，大部分的因特网程序所完成的主要工作就是从

客户端收集信息，并提交一个访问数据库的请求，然后将结果返回给客户端。.NET使用新的ADO.NET技术为数据库操作提供了良好的支持。本书的第6章专门讨论ADO.NET。

- 对处理XML文档和流的超强支持。现代分布式计算环境需要应用程序能够处理XML。对于编写处理XML文档和流的应用程序而言，.NET框架可以提供很强的支持。我们将在第7章讨论XML。
- 通知异步事件的标准化机制。在.NET之前基于COM编程的环境下，提供从服务器到客户端的标准化回调机制是比较麻烦的。.NET框架可以为某一方对另外一方的异步调用提供标准化机制。我们将在第8章讨论事件机制。
- 支持编写多线程代码。Windows操作系统在1993年推出的32位Windows NT操作系统中第一次实现多线程机制。不幸的是，由于操作系统对多线程的底层支持，使得多线程程序很难编写。.NET框架可以提供很强的支持，使得一般程序员就可以充分利用操作系统的多线程能力。我们将在第9章讨论多线程机制。
- 对编写自定义Windows窗体和Web窗体控件的支持。控件是一个处理用户界面的功能单元。借助于控件，Windows窗体（第5章）和ASP.NET Web窗体（第3章）可以进一步增加其功能。.NET框架也支持用户开发自己的控件，不仅可以自己使用，也可以销售给第三方。第10章和第11章分别讨论了如何编写Windows窗体控件和Web窗体控件。

1.5 关于本书

直到我撰写《Understanding COM+》一书之前，我曾经编写的所有书籍都包含了大量用C++编写的程序代码，这些书籍都属于针对底层，教会读者如何实现某些功能的手册或指导书。当然这些书籍对于那些使用C++编写硬件核心程序的程序员来说，是相当有价值的参考书，然而这些读者毕竟只占购买电脑书籍的用户人数中的一小部分而已，这样就使得许多购买者不满意。我希望那些不了解或者不喜欢使用C++的开发者们也能够参考这本书。而且，我还发现我原先这种基于C++的写作方法对于经理们来说几乎毫无益处因为他们从来不看例子程序（据我所知只有一个例外，我赠送了那人一本书，他经过一番刻苦研读，终于把我书中的每个C++例程都看明白了）。我真心希望自己的书籍能为这一类的读者所用，而不仅仅只是对程序员有益处。无知的（或者更严重地说，未接受完整教育的）经理的确是极其危险的人物。如果我能通过自己撰写的书籍来消除他们的无知，那么我就是为现代文明做出了一项巨大的贡献。

本书的例程源代码以及关于.NET的安装指导都可以在本书的Web站点上下载得到。

这本书延续了我在上一本书中的写作风格，当时我在那本书中采用了David Chappell在《Understanding ActiveX和OLE》（Microsoft出版，1996年）一书中所使用的写作形式，即：大量的解释，大量的图表，而以文本描述的代码则非常少。尽管我是那么欣赏David Chappell的写作方式，可我还是觉得不能缺少代码（就像我在吃完一顿寿司后一定还要再来一块巧克力蛋糕一样）。我发现编写代码有助于我理解David Chappell的思想，这也就像我在阅读斯蒂芬·霍金的《时间简史》一书时，喜欢写出数学等式来帮助自己理解文字性的描述。所以我在这本书的每一章中都使用了示例程序，其中有些程序是我自己编写的，有些则来自于微软的例子。当然，在本书的网站http://www.introducingmicrosoft.net上也能找到这些示例程序的源代码。经理和设

计师们不会因为面对大量的代码而一片茫然，同时喜欢阅读程序代码的程序员们也能满足自己的喜好。这本书中的示例程序是用Visual Basic .NET编写的，因为我觉得我的大部分读者对这种语言更熟悉。但是，在本书的第一版出版后，很多读者希望能看到C#代码，因此，在本书的网站上，我也提供了所有例子的C#版本。如果你想运行这些程序，那么必须具备以下条件，计算机的操作系统为Windows 2000 Server，或Windows XP Professional，同时还需要安装Microsoft .NET SDK，以及Visual Studio .NET。关于运行这些示例程序的系统和安装的详细要求请参阅本书Web站点中的说明。

本书的每一章都从上至下论述了一个单独的主题。

本书的每一章都集中论述了一个单独的主题。每章开始之初，我都是先介绍有待解决的体系结构问题。接下来我会向大家解释.NET提供的基础设施中的高层体系结构，这种基础设施能够帮助你用尽可能少的代码来解决前面提到的问题。然后，我会给出一个采用这种解决方法的最简单的例子。经理们可能会在这一部分之后就不再继续读下去了。不过我还要继续与读者讨论一些更深入的问题——其他的可能性、边界情况以及与此类似的问题。总之，在整个章节中，我会始终遵循一个原则，那就是“不要有过多的示例程序”。

1.6 歌唱硅片

现代诗歌意境肤浅。

我很讨厌现代的诗歌。因为我发现这种诗歌的大部分内容都和夸夸其谈的政治性文章没有什么两样。现代诗歌既不押韵，也没有任何节奏感，仅仅就是作者（他们通常都有自己的收入或者纳税人给他们的补助金，否则这些人一定会饿死）仿佛受到什么致命的欺骗一样在那儿大发感慨，似乎有很多重要的东西要讲。或许是我虚弱的大脑不愿花力气去分析现代诗人故意的胡言乱语。不知道你们是怎样看待这些诗歌的，反正我的大脑是要用来思考别的事情的。

Rudyard Kipling的诗歌相当好。

相反，我却很喜欢古代的诗歌，尤其是Rudyard Kipling的诗歌。他的诗歌在政治性方面却并不符合现代的标准——如果你想知道为什么这样评价他，不妨读一下他的那首诗“The White Man's Burden”。不过，这并不是他的过错，因为他可以说是其所处时代的产物，就像我们也是现代社会的产物一样。Rudyard Kipling获得了1907年的诺贝尔文学奖，所以我相信那时起就有人喜爱他的作品了。祖父给了我一本他写的书《Just So Stories》。小时候，父母读着这本书哄我睡觉，同时，这本书也是我开始学会自己阅读的书籍之一。Kipling的诗歌一直陪伴着我度过了中学时期的英语课，因为我发觉阅读课本上由他写作的那一部分要比听老师讲课有趣得多。他的诗歌一直在我耳边萦绕，让我难以忘怀，从来就没有谁的诗歌能像他这样打动我，相信以后也不会有。

Kipling曾经写了一首诗赞颂一位叫作McAndew的远洋轮船工程师，这首诗的全部内容都可以应用于今天的程序员。

你也许会问，我说的这些和计算机工作者有什么关系呢？在过去的这几年中，计算机技术的创新可谓令人难以置信，这使我想起了Kipling于1894年发表的一首诗《McAndew's Hymn》。

大部分人都认为当今的社会与一百年前大不相同，而科学领域中的每一项技术更是莫不如此。可是，我仍然要惊讶于Kipling当年的感受居然和我现在的感受如此一致。上述那首诗的标题所表达的就是，一位苏格兰的远洋轮船工程师对于他那个时代最伟大的一项技术成果："船舶蒸汽机"所发出的深思。然而像这样伟大的技术发明只是一个开头而已，接下来的历史以及将来还会不断涌现许许多多伟大的技术创新。要想阅读整首诗，你可以访问http://home.pacifier.com/~rboggs/KIPLING.HTML。你或许会认为Scotty 是最早期的苏格兰工程师，但是我却认为Gene Roddenberry的成绩都是建立在Kipling的McAndrew这首诗上的。

这首诗包含了摩尔定律的早期公式。

举例来说，几乎每个程序员都知道摩尔定律，对吗？摩尔定律认为对于某个确定的价格，计算机的计算能力会每隔十八个月就增长一倍。许多程序员可能还知道与其相辅相成的一个定律，即Grosch定律，该定律认为无论计算机的硬件性能如何的优越，都不会形成什么好结果，因为这些硬件的优越性会被软件的不良性能所抵消。甚至还有人知道Jablokow推论，该理论的观点更简单，即"And Then Some"。然而早在那些剽窃者将这一观点冠以Moore的名称，并标榜为摩尔定律之前，McAndrew就已经在一百多年前阐述了这一观点。当我凝视着原始的4.77M IBM PC机（带有两个软驱和256K的内存）时，我就想起了Kipling的话。

[I] started as a boiler-whelp when steam and [I] were low.
I mind the time we used to serve a broken pipe wi' tow.
Ten pound was all the pressure then - Eh! Eh! - a man wad drive;
An' here, our workin' gauges give one hunder' fifty-five!
We're creepin' on wi' each new rig - less weight an' larger power:
There'll be the loco-boiler next an' thirty mile an hour!

就像Rodney Dangerfield一样，我们每个人都喜欢得到别人的尊敬和欣赏。早在原始社会时期，居住在山洞中的原始人就会查看锋利的石头并且说："多么别具一格的不规则形状呀。我在想它是否算得上当今最流行的样式了。"回忆一下中学时那些女生疯狂地涌向足球运动员的场景，这些运动员中的大部分人（不是全部），有谁会像石头一样一言不发呢？在这些女生眼里，优异的成绩A并不算什么，即使我获得了象棋冠军的称号也敌不过大学体育代表队的那些运动员们。尽管我知道最后我所挣的钱要比那些校级运动员（父亲说过这些运动员却比其他任何人更能吸引异性）多得多，可我还是无法吸引女生的目光。McAndrew也由于和我相同的原因而大声哭喊，只不过他的口才更好，不妨看看他是怎么说的：

Romance! Those first-class passengers they like it very well,
Printed an' bound in little books; but why don't poets tell?
I'm sick of all their quirks an' turns-the loves an' doves they dream-
Lord, send a man like Robbie Burns to sing the Song o' Steam!

我既不是Robbin Burns，并且母亲也不喜欢听我唱歌。不过我已经尽我所能向大家生动地讲述了这个故事，仿佛我今天亲眼看到了故事的情节一般。希望大家喜欢它。

此为试读，需要完整PDF请访问：www.ertongbook.com