实用培训教程系列

Visual C++.NET 中文版

实用培训教程

- 深入剖析 Visual C++.NET 开发应用程序的方法和技巧
- 思考与练习相结合，检测您理解与掌握 Visual C++.NET的程度

晓 东　王胜海　等编著

2C-43

清华大学出版社
http://www.tup.tsinghua.edu.cn

本书配光盘一张

实用培训教程系列

Visual C++ .NET 中文版
实用培训教程

晓东　王胜海　等编著

清华大学出版社

(京) 新登字 158 号

内 容 简 介

本书由浅入深地介绍了微软最新的开发工具 Visual C++.NET，它是微软公司具有里程碑意义的软件开发套件——Visual Studio.NET 的重要成员之一。书中介绍了 Visual C++.NET 的概念和开发特性，并用了一定的篇幅说明了.NET 开发的基石——.NET 框架，以及.NET 应用程序开发的核心内容，然后对 Visual C++.NET 的语言核心——C++托管扩展(Managed Extension for C++)作了介绍，接着以大量的实例讲述了 Visual C++.NET 的 COM 技术基础、ATL Server 和 ATL Server Web 服务开发，并对新旧技术作了对比性的说明和讲解。最后，又对 Visual C++.NET 的统一事件模型做了重点讲解，这是 Visual C++.NET 的一个重要的新特性，也是其强大的功能之一。

本书在列出大量应用操作实例的同时，在书中随附的光盘中给出了所有实例的源代码，以帮助读者总结每一章的要点，检查并巩固所学的知识与技术。另外，光盘中还附带有安装所需要的补丁程序。

本书适合 C/C++的所有用户学习，尤其适合 Visual C++ 6.0 的用户学习。通过本书的学习读者可以很快地过渡到 Visual C++.NET，能够成为新一代优秀的开发人员，同时它也是一本非常适合初学者的优秀的培训教材。

书　　名：Visual C++.NET 中文版实用培训教程
作　　者：晓东　王胜海　等编著
出 版 者：清华大学出版社(北京清华大学学研大厦，邮政编码：100084)
http://www.tup.tsinghua.edu.cn
责任编辑：张秋香
印 刷 者：北京密云胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092　1/16　**印张**：21.75　**字数**：516 千字
版　　次：2002 年 2 月第 1 版　2002 年 2 月第 1 次印刷
书　　号：ISBN 7-900641-35-1
印　　数：0001～5000
定　　价：38.00 元（含光盘）

第1章　进入.NET世界

在这一章里，我们将首先对什么是.NET，.NET 对用户的重要意义，以及.NET 战略的基本问题作出明确的解释，以使读者可以很快地从概念上对.NET 有一个大致的了解。

1.1　.NET 简 介

首先要说明的是：.NET 的含义并不简单。在讲解其定义之前，让我们先来看看其设计目标，以达到事半功倍的效果。

Microsoft.NET 首先是飞速发展的网络形式的产物，目标是使程序员，特别是使用 Microsoft 技术的程序员能够在一个应用程序开发平台上快速开发出功能强大的 Web 应用程序。.NET 代表了一个集合、一个环境、一个编程的基本结构，作为一个平台来支持下一代的因特网，这是一个可实现的环境。.NET 也是一个用户环境，是一组基本的用户服务，可以作用于客户端、服务器，以及任何地方，并与该编程模式具有很好的一致性，而且有新的创意。

从设计角度上看，Microsoft.NET 平台和 Windows 平台较为相似。该平台的建立，使用户能创建出更多的应用软件。Windows 平台上可以使用大量第三方技术，促使个人计算机产业发生了巨大的变革。同样地，.NET 能够让遍布全球的网站提供更有价值的服务，还能让用户以非常简便的方式从多个网站上获取信息。

可以这样说，.NET 是一个可以支持下一代因特网服务和运营的平台。这个平台包含微软新一代的操作系统—— Windows XP、大量的因特网服务软件、对各种设备(PDA、机顶盒、信息家电等)的支持和微软提供给开发人员的最新应用软件开发套件—— Visual Studio.NET 等。在这个平台上，用户可以得到完整地接受服务，开发人员可以得到平台在开发方面的有力支持，以创建各式各样的应用软件。这个平台可以简单地用图 1-1 来表示。

除此之外，.NET 战略同时也是为了对付日益流行的 Java。也就是说，微软将未来 IT 行业发展的一切都以一个名词来概括，这个名词就是.NET。

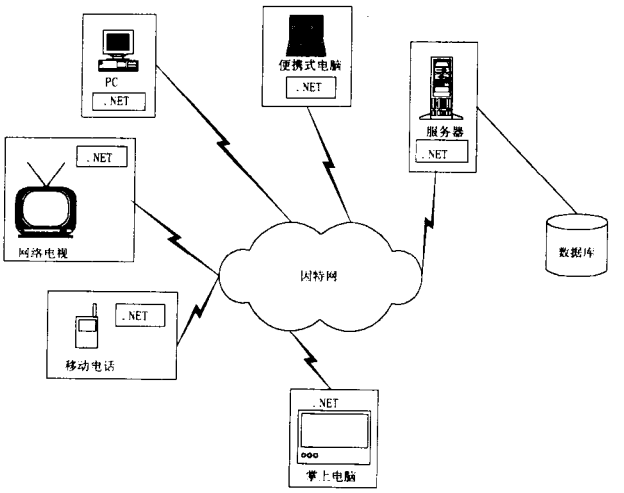

图 1-1　NET 作为支持因特网的平台将所有类型的设备连接起来

1.1.1　.NET 的核心内容

微软认为.NET 的核心内容就是要搭建第三代因特网平台，同时建立这个平台上的运营规则。这个网络平台将解决网站之间的协同合作，从而最大限度地获取信息。在.NET 平台上，不同网站之间通过相关的协定联系在一起，网站之间可以自动交流、协同工作，以提供最全面的服务。

微软认识到服务走向因特网是 IT 产业发展和世界经济发展的必然方向，所以.NET 的核心目标就是对其现有的计算模式和软件开发方式进行有针对性的调整，使之符合第三代因特网运营的需要。并利用其现有的科技开发实力，同诸如 IBM 这样的厂商修改和定义网络通信协议，建立未来在.NET 平台上的运作规则，搭建第三代因特网平台。

在计算机开发的最初时代，还没有人能意识到组件技术的意义。开发编程大多是 DIY(Do It Yourself，自己动手做)式的做法，设计带窗口的图形界面、菜单、显示图像等全部要由自己编写代码来实现。

Windows 平台为广大的程序员们带来了一个通用的系统函数库—— WindowsAPI。通过 WindowsAPI，界面的实现变得非常轻松容易。自 Windows 95 开始，单是按钮、菜单已经不能吸引住用户，界面变得更复杂也更丰富了。一个功能强大、外观漂亮的控件的编码量甚至超过了以往 DOS 程序中所有界面工作的总和。在界面复杂化的同时，应用程序的应用逻辑也在日益复杂化，同时，计算机在各个领域的应用也更广泛，比如在金融、财务及科学等方面。

但是，WindowsAPI 对程序的复杂化起了推波助澜的作用。程序员们迫切地需要一种体系来建立可重用代码。什么是可重用呢？这个问题切中了 COM(component object Model，组件对象模型)的要害，可重用也就是对一个软件中普遍适用的功能进行抽象化，从而达到

不用修改或几乎不用修改就可以直接被其他需要该功能的软件直接使用的目的。

软件可重用也经历了好几个时期。在可重用性开始被人们重视的时候，还没有一个完整的体系来从底层体现它，那时可重用还大多是代码级的可重用，也就是说实现可重用的方法是拷贝相同功能的代码。这个方法有限得很，而且有时还很不方便，源代码不是每一个软件制作者都愿意公开的。代码级重用没有延续多少时间，很快就有另外一种重用方式出现了，这就是库重用。库重用重用的不是源代码，而是二进制库，开发人员把可以重用的源代码编译成库文件(一般是后缀名为 LIB 或者 OBJ 的文件)，然后其他的开发人员可以直接使用库文件中的函数(通常是函数)。库重用相对于源代码重用来说，无须再拷贝任何东西，而只要简单地引入库文件就可以。因为库文件已经是二进制的了，所以开发人员可以放心地发放和使用重用库，而不用担心源代码会泄漏；另外，不像源代码重用那样局限于某一种语言，虽然库文件在各种语言之间还不能完全通用，但是通常在同一个公司的产品中能通用。

库重用虽然缓冲了重用矛盾，但是重用难题并没有被解决。与此同时，模块化的问题随着面向对象理论的出现和应用而迎刃而解，有人开始借鉴面向对象技术来化解软件重用问题。对象重用是继库重用之后出现的一种方案。对象重用不是单纯的重用，而是介于代码重用和高层的逻辑重用之间的一个中间产物。对象重用需要借用代码重用，仍然是需要使用代码来达到重用目的，但是又不像代码重用那样只拷贝代码或者引入一个库就了事。对象本身带有一定的逻辑，可以相对地独立于系统，所以对象重用在库重用的基础上进一步消除了重用者对重用代码的依赖，也就是说，重用者只需较少地了解重用代码就可以使用。

对象重用还是不尽如人意，微软提出了一个 DLL(动态链接库)方案，DLL 属于操作系统级支持的接口重用方案。接口重用使得软件重用第一次在 PC 编程世界焕发出光彩。接口重用不是代码级重用，但也不是真正的高层的逻辑重用，它只能算是一个中间产物。接口重用的思想是让开发人员编写好所有的可重用代码，并根据操作系统的规则编译成接口重用文件(这里指的是 DLL 文件)，然后重用者通过对该文件的接口查询、引入来使用该文件内部的代码。接口重用第一次从代码级重用中脱离出来，代码级重用都需要对语言平台进行指定，而接口重用则不然，接口重用可以采用任何语言，这样 DLL 库就可以让重用在所有语言中进行。接口重用不仅使重用者脱离了代码上的问题，而且也使得重用代码可以脱离原系统而作为一个独立的部分在各个软件之间共享。虽然 DLL 很好地解决了重用性问题，但是 DLL 是静态的。之所以说是静态，是因为 DLL 本身什么也不知道，就像库文件一样，它自己只能等待程序来调用它，而不能有任何动作和交互。

微软在 DLL 的思想上再进了一步，ActiveX 也参与到重用性的混战中了。ActiveX 和 DLL 最大的不同在于 ActiveX 能解释自己。也就是说，ActiveX 能通过调用它的程序进行交互，它本身也可以作为一个软件的一部分参与应用事务。这是一个了不起的进步，现在重用这个词已经不仅仅限于代码上的节省了，而且延伸到应用逻辑领域。因为 ActiveX 能参与应用程序的交互和逻辑处理，所以应用程序可以在 ActiveX 的基础上构筑更大、更复

杂的应用。这样，编码工作实际上就被分散了，一部分人去专心编写他们的 ActiveX，而另外一部分人则继续他们的应用程序编写，然后两部分工作一合并就可以产生一个完整的程序，这样程序的稳定性也大大增加了。

ActiveX 并没有停滞不前，COM 的产生为重用带来一个新名词——组件对象模型。COM 是 ActiveX 二代，不仅具有更完善的重用性能，而且对网络的支持也更好。现在的 COM 已经被集成到了 Windows 系统中，成为操作系统的一部分。

微软认为.COM 是信息海洋中一个一个的“数字孤岛”。而将这些数字孤岛连接起来，打破不同的操作平台、不同的网络设施以及业界遗留下来的藩篱，从而实现因特网统一的解决方案就是.NET。这个方案如图 1-2 所示。可以看出，微软在基本统一了 PC 之后，不但想向因特网进军，而且还要通过进一步的革新和发展来统一因特网。

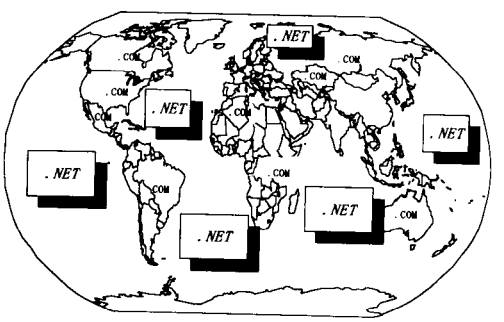

图 1-2　.NET 将.COM“数字孤岛”联系起来

.NET 组件使用非专用的、独立于平台的 XML 与其他组件在网络上进行通信，可以轻易地跨越防火墙，而且不受网络协议的限制。.NET 组件与 COM 组件不同，不必放在系统注册表中，而是在被称为“清单”的区域中封装全部信息，详见第 3 章的具体介绍。

1.1.2　.NET 对用户的重要意义

从只有极少数专业人士才能接触到的大型计算机到安装有 DOS 的 PC 机，为 DOS 操作系统的简单易用使得使用电脑的人数大大增加。与此同时，计算机使用人数的增加也刺激了 PC 业的发展，进而对 PC 操作系统的发展产生强大的推动力。从 Windows 开始装备 PC 机的那一天起，使用电脑的人数呈几何级数增长。因为，Windows 操作系统不仅提供了比 DOS 更为强大的功能，而且提供了更为友好的人机交互界面，使人们能够比以往任何时候都可以方便地使用计算机，并将计算机看成是辅助自己完成繁琐工作的得力助手。但是，因为受限于当今软件科技的发展水平，距离“科技以人为本”、“科技服务于人”的理想标准，还有相当长的一段路要走。

.NET 对最终用户来说非常重要。以前人们必须具备相当的技术功底才能让计算机很好地为他们服务。很多计算机概念令人难以理解，以致于最终用户在掌握计算机使用技能时总是需要花费大量的时间和精力。但是，随着.NET 平台的诞生与发展，在.NET 网络平台

上，人们可以使用键盘、鼠标、手写、语音和语言等各种方式进行人机交流，这将极大地降低人们使用网络的难度，从而为构建全真虚拟网上社会打下基础。.NET 平台提供的强大功能可使用户轻松地进行因特网连接，并完成那些在当时看来十分费时、费力的事务。用户可以自由访问、查看及使用自己的数据。

随着宽带技术的发展和普及，以及新一代因特网技术的发展，越来越多的非 PC 设备和信息终端在提供网络服务方面扮演着越来越重要的角色。例如，传统的电视将被机顶盒代替，人们可以与电视节目进行交互，这将极大地丰富人们的生活。

.NET 的出现，意味着人们只用一种简单而统一的界面就可以编写、浏览、编辑和分享信息。由于.NET 世界中的“世界语”是 XML，而作为信息交换载体的所有文件都以 XML 格式存在，所以任何规模的公司都可以使用相同的信息交换形式与供应商、伙伴和客户分享信息。这样，一种全新的协同工作模式就随着.NET 的应用而应运而生。

1.1.3　微软的.NET 战略

.NET 战略，大概是微软公司自创建以来制定的影响最为深远的决策。其意义之深远、涉及面之广、实施难度之大，恐怕只有从微软这样的“巨人”嘴里说出来才会有人相信。并且，微软的这个战略确确实实地已经成为了 IT 行业的业界之最，因为用微软自己的话来讲，“真正的竞争对手又是谁呢？从某种意义上说，这是一个很好的问题。没有任何对手宣布过他们的.NET。说实话，无论是从开发商或者是最终用户的角度来看，都找不到任何一种其功能类似于.NET 的软件产品。也没有任何其他人士能够做出类似的东西来。”

的确，这个战略实施的难度太大了，因为这个战略的实施进度表不仅仅被微软控制，还必须与成千上万的服务商和诸如 IBM、SUN 这样的竞争对手协同工作，制定因特网未来的规范，这样才能完成这一梦想。微软认识到了这项计划实施起来的艰巨性和复杂性，所以微软公司的领袖把这个战略当成是微软的世纪豪赌。

实际上这是在赌一场软件业的改革。将软件作为一种服务的概念非常引人注目，它将会引起软件同行的关注，但是要等多久人们才能彻底革新他们的业务模式呢？

应用程序和网站集成以及 XML 的重要性正在日益受到重视。大量的服务器、客户和其他零散服务会变得非常重要。微软公司力争把用户界面变为一个能够进行本地存储、自主管理、以各种方式定制工作模式和数据的世界。

由于 Windows 2000 和 Windows XP 是.NET 的基础核心，托管.NET 工作的操作系统和平台是否成功也是极其关键的因素，对于成功地实现可扩展性、可靠性和可管理性至关重要。

1.1.4　理解“软件作为服务”

.NET 的设计目标之一就是让“软件作为服务”。但是对于这句话，我们应当怎样理

解呢？

在人们日常生活中，离不开各式各样的服务。比如，如果用户在当地 ISP 申请了网络使用账号，那么当网络设备连接妥当之后，就可以享受美妙的因特网服务；当观众在所属地区申请了有线电视服务，并按手续缴纳了服务费用后，就可以在闲暇之余，享受精彩的电视节目。两者都是客户通过对某些服务进行“订阅”，并按时缴纳服务费用后，得以享受相关服务的例子。其实对于“软件作为服务”的最简单的认识与上面的例子无异，在网络上开通的各种联机杂志、金融信息等服务，就是我们身边形式最简单的软件服务。

所谓“软件作为服务”其实就是一种提供软件服务的机制。最终用户可以根据自己的需要来“订阅”应用软件的功能，并且这种“订阅”是有选择性的，用户可以在一个庞大的应用程序中选择他们所需要的那一部分。这种“订阅”机制不同于传统应用程序，它可以满足最终用户对于自有配置应用的需求。

举个例子，像 Word 这样的字处理软件，虽然功能强大，但是仍然有很多用户对其感到不满，因为 Word 中包含了太多他们根本就不需要的功能。用户都希望能够对软件进行自由配置，因为只有这样，才可以在得到足够功能的同时，省掉为不需要的软件功能而支付费用。

随着.NET 应用的普及，软件最终可以实现在因特网上的动态安装—— 用户可以只安装感兴趣的或当前工作需要的部分，或者干脆将整个程序都放在网络上。这很像现在多数人使用的基于 Web 的电子邮件系统，只要连接到因特网，不论使用的是哪一台电脑，也不管这台上网的电脑物理位置在何处，都可以自由地浏览、编辑自己的邮件。这种方式还有助于拉近软件开发人员和最终用户的关系，因为一方面，用户可以从频繁的升级和补丁中获得软件的新特性，而另外一方面，软件开发人员也可更好地理解用户的需求，进而开发出更受消费者欢迎的产品。

最终用户可以从.NET 的许多方面享受到动态安装和订阅服务带给他们的好处。为了实现这些令人向往的功能，装配件作为.NET 技术的一个核心概念被引入，使用装配件，应用程序可以通过一个清单(manifest)来维护应用程序的组件信息。当然，这不仅是一个专属因特网的新特性，它同时也是解决 DLL 兼容问题的基础。有关装配件和清单的详细内容，请参阅本书第 3 章。

“软件作为服务”的另外一个方面，就是在网络上能够暴露可编程接口，并通过这种方式来共享站点的服务。而当今，Web 网站的开发模式决定了站点彼此之间是相互隔离的“孤岛”。举个例子，如果某个 Web 站点开发人员要为他的站点提供信用卡验证功能的支持，那么他要么自己开发软件，要么从别人那里购买软件模块。而.NET 采用的方式将简化他们的工作，因为相关的服务就放在网络上共享的地方，随时为需要它们的人们准备着。开发人员可以直接引用这些接口来完善自己应用程序的功能。这个允许在 Web 站点之间暴露可编程接口的技术叫做 Web 服务(Web Service)，它通过将标准的 XML 语言和 SOAP(简单对象访问协议)作为信息描述和交换的手段，使“软件作为服务”不再是梦想。有关 Web 服务的具体内容，在本书的第 9 章中有更为详尽的论述。

1.2　.NET 开发框架概述

在这一节中，读者可以了解到软件开发模式不断发生变革的原因，并可以对.NET 为软件开发带来的新特性有一个初步的了解。

1.2.1　变革原因

Windows 操作系统从诞生之初一直发展到今天，经历了若干个版本的升级，并由此不断带动着应用软件开发模式的更新和变革。在这里，我们首先对这些变革做一个简要的回顾。

早在 Intel 的 8088/8086 微处理芯片几乎已经在世界上每一台 PC 机上装备的年代，也正是 DOS 操作系统盛行的时期。开发人员一定还记得(或听说过)在 DOS 上进行开发的方式—— 一切系统资源都是属于开发人员的。那时，开发系统级的程序(如驱动程序)和开发一个应用程序没有太大的区别，因为开发人员可通过对公开或未公开的中断进行调用，或采取直接写屏方式来编写应用程序。这种开发模式持续了相当长的一段时间，一直到 Windows 操作系统诞生并流行起来，才改变了这种开发模式。在 DOS 下的开发模式可以用图 1-3 来表示。

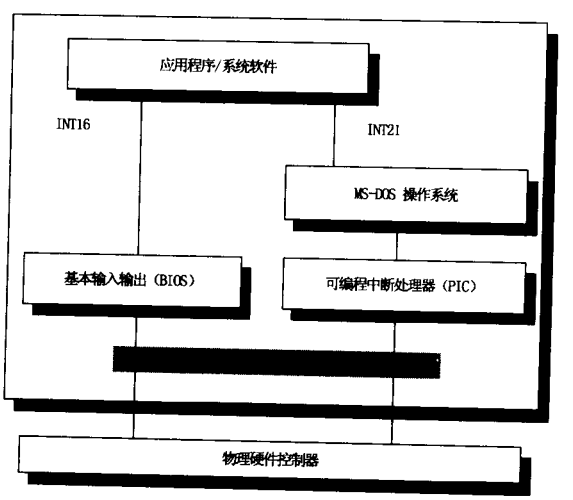

图 1-3　MS-DOS 应用程序开发模式

注意：

就是在这个时期，也许很多人都会忽略一件重要的事情—— Windows 的第一个版本 Windows 1.0 诞生了。但是，因为当时硬件的发展无法满足 Windows 操作系统的要求，所以直到 80286 芯片推出的最初那段时期，Windows 操作系统都没有普及开来，DOS 下的软

件开发模式仍然是软件开发的主流。

后来，随着 PC 工业的高速发展，一代代更加强有力的处理器被不断地制造出来并推向市场，PC 机也因此拥有了越来越强大的处理能力。

在 80286 时代，由于 80286 CPU 是一个 16 位处理器，所以在 16 位保护模式下，它的寻址范围突破了 8088/8086 的 1MB 字节的寻址界限。而正是由于 CPU 寻址能力的提高，才为先进操作系统的进一步发展铺平了道路。到了 80386/80386$^+$时代，处理器已经可以在 32 位保护模式下寻址 4GB。这项技术进步不但让 Windows 3.x 逐步普及到了 PC 机上，而且在理论上，还为功能更为强大的操作系统(如 Windows 95)的产生和发展奠定了物质基础。

那时，开发一个标准 Windows 应用程序仍然被公认为是一件很困难的事情。因为开发一个 Windows 应用程序意味着必须使用一种高级语言，并利用合适的软件开发包(SDK)提供的支持文件来构造应用程序。这个时期(从 Windows 1.0~Windows 2.x)系统提供给开发人员的应用程序编程接口(API)还是比较原始的 Win16，在程序的代码中，API 接口不能很好地同系统级的代码实现分离，并只利用了基础硬件体系的一部分特性；任务和任务之间的调度是伪多任务机制，而非真正的抢先式多任务机制。因此，单纯从开发模式上讲，Win16 还没有形成一个完善而科学的开发模式。这种开发模式可以用图 1-4 来表示。

第一次软件开发方式的变革就发生在随后的日子里。随着硬件的飞速发展，强大的 80486 和 Pentium 处理器的诞生和普及为新一代操作系统的产生提供了可能。当这些强大的操作系统(如 Windows 95 和 Windows NT)被成功地推向市场之后，一个著名的软件开发模型随之推出，它就是 Win32 API。这个应用程序编程接口代表了平台编程接口总体质量的一次非常显著的提升。Win32 API 与 Win16 API 相比，具备的最大优点就是抛开了对 DOS 系统服务的依赖，实现了真正的多线程、抢先式多任务、抛弃了内存分段的限制、增加了异步输入模式以及完成了真正的应用程序之间的隔离(即应用程序认为其独自享用一个 4GB 的虚拟内存空间)等。直到今天，Win32 API 软件开发模式仍然是 Windows 系统核心开发模式之一，这种开发模式可以用图 1-5 来表示。

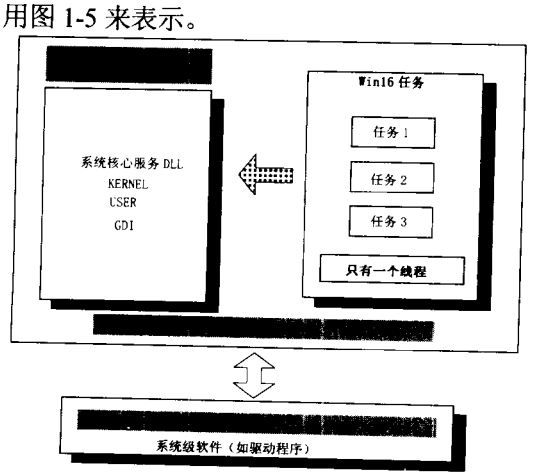

图 1-4　Win16 的应用程序开发模式

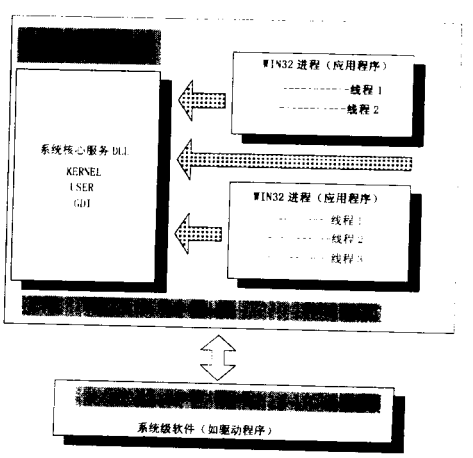

图 1-5　Win32 应用程序开发模式

随着 Win32 API 体系的逐步稳定，以及其包含的基本服务接口的不断完善，越来越多的 Win32 应用程序被开发出来。在这些应用程序当中，不乏像 Office 那样的软件产品。由于这些软件规模巨大，往往需要成百上千的工程师进行协同开发才能最终实现软件的所有功能，因此业界对用于应用程序管理的技术提出了更高的要求。在 Win32 编程模式中使用动态链接库进行软件管理的传统技术，已不能满足需要，另一种软件管理方案被微软推出并逐渐成为 Windows 操作系统的核心开发模式之一，这个方案就是 COM。有了 COM，设计良好的应用程序便可以被分割成为若干组件，而这些组件都可以被单独开发和发布，这样就解决了应用程序的管理问题。另外，COM 具备的一个重要特性——二进制兼容，可以实现不同开发工具的协作开发，对软件产业的发展产生了极大的推动作用。这种开发模式可以用图 1-6 表示。

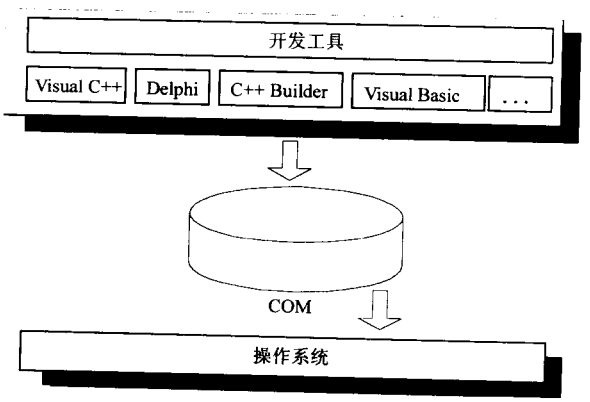

图 1-6　COM 开发模式

最近几年来，随着因特网的高速发展以及全球最热门可热的 IT 经济增长点定位在电子商务领域，Windows 平台上的软件开发技术也沿着基于 Web 的分布式应用程序开发方向不断地发展。

为了实现分布式应用程序的开发，微软对 COM 进行了必要的扩展，从而形成了可以服务于企业应用的 COM+。COM+是一个扩展的 COM 运行环境，在基于 Web 的分布式应用程序架构中扮演着核心角色。众所周知，对于企业级的应用来讲，事务处理、安全服务和同步服务都非常重要，COM+体系就针对这些要求提供了完整的支持。另外，COM+还提供了诸如队列组件、时间服务、内存数据库和负载平衡等核心服务来满足复杂企业级应用的苛刻要求。同时，COM+也被集成为微软操作系统的组成部分，在微软某些操作系统(如 Windows 2000)中，已经内置了对 COM+技术的支持。

在此基础上，微软提出了一个名为 Windows DNA 的分布式应用程序开发模型，这个模型其实就是建立在 COM/COM+技术之上并为分布式应用程序开发提供开发支持的软件平台。Windows DNA 的结构分为三层，分别是表示层、业务逻辑层和数据层。这种分布式应用程序架构如图 1-7 所示。

通过上面的回顾，读者基本上可以对 Windows 平台上应用软件开发模式的发展历程有一个大概的认识了。显而易见，Win32 API 和 COM 开发模式是当今 Windows 平台上的主

流开发模式。这两种模式，从它们诞生到发展至今，都经历了长时间实际应用的考验，已经日趋成熟，而当今每个在 Windows 平台上进行软件开发的人员，在实际工作中也都直接或间接地与它们打交道。

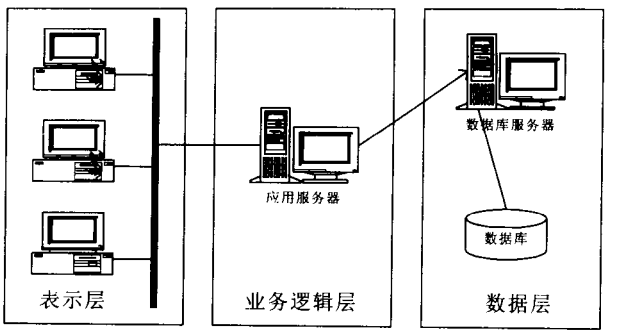

图 1-7　Windows DNA 架构

现在回到本书讨论的主题。既然微软拥有了成熟的软件开发模式，为什么还要提出全新的.NET 呢？这是一个值得我们思考的问题。

其实，没有一家厂商会在没有足够依据的情况下就提出一个革命性的软件开发模式的。事实上，成熟的模式不等于是没有问题的模式，更不等于是完全符合未来发展趋势的模式。

多年来，Windows 操作系统被人们认为是复杂的和不稳定的，这几乎已经成为开发人员和最终用户的共识。那么究竟是什么原因，让 Windows 的名声如此差劲呢？

有一种说法是，Windows 采用的软件技术不够好。但是，这是 Windows 早期版本才有的缺陷，那个时侯，Winodws 的某些产品为了完成向下兼容，在设计中作了非常多的折衷，由此造成了许多错误，使得系统不够稳定。这种说法对于像 Windows NT 这样的操作系统是不正确的，但是 Windows NT 4 及以后版本的操作系统，其各方面的表现也并不是让用户十分满意的。其实，微软操作系统具备很多优点，比如它的分层设计就十分出色；它的驱动程序开发模型 WDM 在设计上也具有一流水准，就这一点来讲，可能很多与之竞争的操作系统都不是对手；COM 架构是一个很优秀的组件模型，它和 OMG 组织的 CORBA 以及 SUN 公司的 EJB 都走在组件技术的前沿。

既然微软在技术上，不但没有什么明显缺陷，而且在某些方面还有一定的优势，那么为什么 Windows 应用程序总是运行不稳定呢？到底造成这些问题的根源是什么呢？

包括微软开发人员在内的许多专业人士对 Windows 进行的研究证实，系统中存在的许多不稳定因素都与过于复杂的应用程序架构有着紧密的联系。

是的，在日趋成熟的现有应用程序架构中，仍然存在着许多困扰开发人员和最终用户的问题，这些问题主要有：

- 由版本问题造成的“DLL 灾难”

应用程序往往需要调用由其他供应商提供的共享动态链接库中的服务来简化自己的开发，但是，这种方式带来了一个大问题——这些动态链接库是由不同厂商开发的，所以

开发人员正在使用着的共享动态链接库的版本非常有可能不是最新的(事实上，开发人员根本没有任何办法保证在他们的开发中，总是使用共享动态链接库的最新版本)。

一般来讲，使用共享动态链接库的应用程序在其发布到最终用户的机器上之前，不会发生运行错误，原因是，通常这些应用程序在推出之前都要经过严格的测试。

然而，问题发生在该应用软件开发商决定发布升级补丁或者更高版本的应用程序时。因为安装新版本的应用程序时，会破坏用户已经安装的老版本应用程序使用的共享动态链接库，一旦这个共享动态链接库没有做好版本的向下兼容工作，则会造成老版本的应用程序出现运行时错误。这就是由版本问题造成的 Windows 平台上的“DLL 灾难”。

当然，如果用户安装了另一个由不同厂商开发的应用程序，并且该应用程序也用到了这个共享动态链接库，而且它本身也存在着版本不同、向下兼容没有完全解决的问题，同样会造成应用程序的运行时错误。

久而久之，这些令人烦恼的问题会让用户在安装新软件时产生恐惧感。用户会考虑软件升级是否值得，因为一旦将新版本的软件安装到机器上，有可能会影响到老版本软件的运行。

- 应用程序的发布通常要影响许多系统组件

在 Windows 操作系统中，安装软件仍是一件复杂的事情，安装过程不可避免地要影响整个系统。例如，安装应用程序不但要将文件安装到用户指定的目录中，而且还要在注册表中进行这样或那样的更新工作，并且还要在桌面、快速启动工具栏或“开始”菜单上建立对应的快捷方式。

由于应用程序与注册表有太多的关联，应用程序如果发生变更，注册表中相关的内容也必须进行更新。正是由于这个原因，用户不能通过对应用程序进行简单拷贝而完成对应用程序的安装，通常必须执行特定的安装程序。还有，用户不能简单地删除应用程序以完成卸载任务，因为可能很多与应用程序有密切关联的组件仍然存在于系统中。

- 应用程序的安全控制机制还很不完善

稍微有一点 Windows 编程经验的开发人员都可以编写一个只要一开始运行，就会删除 Windows 目录下的所有核心文件的应用程序。而如果这个恶作剧程序恰好被用户下载到本地计算机上，并执行了它，那么后果将不堪设想。

对于这样的程序，Windows 系统并没有提供一种安全机制来对其行为加以限制，这个明显的缺陷将让每个最终用户在安装软件时，担心自己会受到伤害。因此，Windows 必须要想办法解决此类安全问题。

对于这些问题，微软也采取了很多的手段来解决，比如引入 Windows Installer 安装工具，但因为这些方式都基于老的技术路线，不能标本兼治，因此收效甚微。为此微软公司提出了新的解决方案——.NET 框架(.NET Framework)。这个框架建立的目标之一就是使开发人员在工作中不会出现上述问题。在接下来的章节中，我们将对相关内容进行详细介绍。

1.2.2　.NET 开发特性预览

今天，因特网发展的浪潮冲击着社会生活的每一个角落，不难预见，人们对因特网的依赖将与日俱增。人们可以利用因特网进行购物、安排旅行、查看产品信息和聊天，也可以使用因特网足不出户地完成个人商务上的联系。今天，因特网上已经存在的技术保障了上述需求的实现，但是，对于人们更为高级的要求，现有技术还不能实现。比如，在几分钟之内，通过网络为远道而来的亲戚在离居住地点很近的一家旅馆订一个房间，或者在即将到来的假期里，通过网络上策划旅游行程并对这次旅行费用进行初步结算等。

因特网尚且不能实现人们愿望的主要原因有两个：一个是因特网上还没有一种将各种服务进行集成的机制。毕竟，传统是很难改变的，产品的供应商总是喜欢使用自己的方式来描述他们的产品和服务。为此，微软决定打破这种格局，与其他大厂商一起提出了用来进行网络数据交换用的公共协议并制定相关业界标准(如 SOAP、XML 等)。另一个原因是应用程序的开发还显得太复杂。为此，微软准备为开发人员提供一个崭新的具有划时代意义的开发平台，以使应用程序的开发更为简单。这个开发平台就是我们本书讨论的中心内容之 ——.NET 框架。

.NET 框架是一个可以构造、发布以及运行 Web 服务的开发环境。从概念上讲，.NET 框架平台代表了一种崭新的软件开发模式，它与 Win32 API 或 COM 一样，是把系统服务以接口形式提供给开发人员的软件开发平台。与以往不同的是，.NET 框架能够更好地完成代码重用、资源配置、多语言集成开发和安全管理等任务，在安全性、易用性以及开发效率等方面远远超过了以前的开发模式。

.NET 框架平台包含两个方面的核心内容：一个是运行时引擎(Runtime Engine)，另一个是基类库(Base Class Library)。对于运行时引擎，可能所有的开发人员都不会感到陌生，因为，从使用标准 C 语言开发工具开始，我们就至少接触过 C 运行时库(C Runtime Library)。后来，我们使用的工具越来越强大，接触过的运行时库的种类也越来越多。像微软基本类库(MFC)、活动模板库(ATL)、VB 运行时库(VB Runtime library)和 Java 虚拟机等。从广义上讲，Windows 本身就可以被看成是一个运行时引擎和运行时库。开发人员都很喜欢使用这些库，因为这些库将复杂的、重复性的东西都准备好了，等着需要它们的人来调用，从而简化应用程序的开发。但是，这些还远远不能满足开发人员对快速开发的要求，于是微软在.NET 框架中提供了一个基类库，这个基类库可以充分地将开发人员编程的难度降下来，让开发人员更为轻松地完成开发工作。

既然.NET 框架可以解决那么多的问题，就势必意味着系统的某些组成部分发生了深刻的变革。下面就让我们一起来看一看.NET 框架中所包含的核心内容，如图 1-8 所示。

.NET 框架具备的主要特点有：

- 一致的编程模式

所有的应用程序服务都由一个通用的面向对象的编程模式来体现，这个编程模式和以

前那种由操作系统通过暴露核心动态链接库输出函数的方式以及通过访问 COM 对象提供接口的方式不一样。

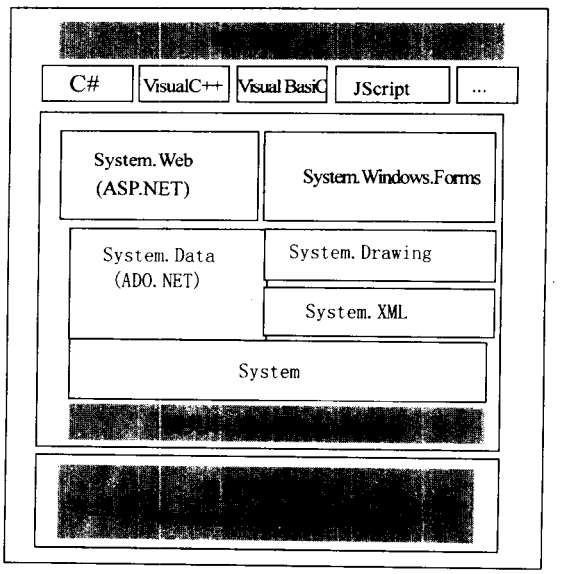

图 1-8　.NET 框架

注意：

可以反过来理解，就是在过去的应用开发模式中，系统没有提供所谓的应用服务，而且这种服务也不是真正面向对象的(当然 COM 可以被认为是面向对象的，但它采用的方式不是标准继承而是相互组合)。

.NET 框架提供了应用程序服务，这种服务是面向对象的，这些都是原来的应用开发模式没有提供的特性。

- 简化的编程模式

新的编程模式极大地简化了 Win32 API 和 COM 编程模式的复杂性。在新的开发平台上，开发人员终于从必须理解复杂的 Win32 和 COM 概念的痛苦中解放出来，也终于可以忘记诸如 GUID，Iunknown，AddRef，Release 和 HRESULTS 等难以理解的概念。

注意：

初学 Win32 API 和 COM 的开发人员，总要经历痛苦的入门过程。如果使用 SDK 开发一个简单的 COM 程序，单是那些“类工厂”、等概念就足够开发人员去了解相当长的一段时间了，而在.NET 编程模式中，这些概念根本就不存在，也更谈不上去了解！

- 一次运行就会永远运行

Windows 应用程序开发人员应该听说过“DLL 灾难”这个术语，这个经典问题会时常给我们制造麻烦。在前面我们提到过，当一个新的应用程序安装到机器上的时候，如果新

安装的组件覆盖了原来程序依赖的组件，就可能导致原来应用程序的崩溃。

在新的应用程序架构中，采用了一些专门的技术来保证应用程序组件之间的隔离，从而保证应用程序总是加载那些原来同它一起经历过开发和测试的组件，这样就解决了“DLL 灾难”问题。

● 一次编写，一次构造，处处运行

现在，最终用户那里运行着太多的 Windows 版本操作系统：Windows 95/98/98SE/Me 、Windows NT 4/2000、Windows CE 以及即将发布的 64 位版本的 Windows。这些操作系统中的大多数都运行在 x86CPU 上，但是 Windows CE 和 64 位版本的 Windows 不是运行在 x86CPU 上的。

对一个托管的.NET 应用程序而言，只要被编写和构造一次，就可以在支持.NET 运行引擎的平台上处处运行，而不需要重新编译源代码。比起以往，这是一个在程序移植性方面的革新。

注意：

“一次编写、处处运行”曾经是 Java 语言发布时提出来的响亮口号，今天，微软也将这个口号提了出来，并且，随着本章讨论内容的不断深入，读者会惊讶地发现，在新平台下运行的.NET 应用程序和 Java 应用程序在运行机制上是非常相似的。

● 跨语言集成

COM 具有的二进制兼容特性可以让不同的语言协作在一起，而.NET 框架可以让不同的语言集成在一起。例如，如果开发人员在 Visual C++中编写了一个托管的类，那么其他开发人员便可以可以轻松地在 Visual Basic 中对这个类进行继承，以产生这个类的子类，在这个类的基础上，使用 Visual Basic 语言继续完善这个类。新的开发模式允许这样，因为平台定义并提供了一个类型系统，这个类型系统在各种语言之间是通用的。

微软提供了若干种语言编译器给开发人员，其中包括 MC++(Managed Extension for C++)、C#、Visual Basic 以及像 JScript 这样的脚本语言。微软还充分地开放了这个标准，让第三方厂商也可以顺利地出品其语言编译器。

● 代码重用

在以前的 Win32 API 或 COM 开发模式下，开发组件是需要开发人员具备相当的技术功底的。但是，在.NET 开发模式下，组件编写是一件很容易的事，所以，可以预见到一个巨大的组件供应商市场会应运而生。

注意：

在自己的软件中，使用功能强大的第三方组件包是一种既省时又可以提高软件功能的手段。由于开发功能完善的组件是一件相当困难的事，所以世界上从事商业组件开发的公司寥寥无几。但是，可以预见，随着新型开发模式的普及，编写组件将会变得和编写应用程序一样简单。