.NET开发专家

C#程序员开发指南

/HOPE/.NET/programming

更多图书请访问 www.bhp.com.cn go

HOPE programming

Microsoft.NET框架结构 | C#基础编 | C#高级编程 | ...

北京希望电子出版社 总策划
易向东 陈 蓓 万 英 编 著

中国林业出版社
China Forestry Publishing House
www.cfph.com.cn

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

.NET开发专家

C#程序员开发指南

/HOPE/.NET/programming

更多图书请访问 www.bhp.com.cn go

HOPE programming

Microsoft.NET框架结构 | C#基础编 | C#高级编程 | ...

C#程序员从入门到精通

北京希望电子出版社 总策划
易向东 陈 蓓 万 英 编 著

中国林业出版社
China Forestry Publishing House
www.cfph.com.cn

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内容简介

本书是一本优秀的学习和应用C#开发程序的指导书。

全书共19章，内容分为三部分：第一部分概述Microsoft.NET的技术特色、体系结构、运行环境等；第二部分比较详细地介绍C#语言，从C#基本语法、面向对象特性，以及异常处理、预处理等多个方面进行介绍；第三部分是C#高级编程，包括线程、界面设计、数据访问、多媒体、COM+服务、XML、Windows服务、Web应用、注册表、文件管理、活动目录、系统管理和诊断以及安全性等专题。

本书概括C#程序设计的各个主要开发领域，同时提供大量实例。内容分析清晰透彻，每个例子都有专门的代码分析部分，以理解所介绍的技术和演示的范例，掌握技术要点和技巧。本书可供软件开发人员使用，也可作为大专院校C#语言的教辅材料。

本书代码从www.b-xr.com下载。

图书在版编目（CIP）数据

C#程序员开发指南/易向东，陈蓓，万英编著.—北京：中国林业出版社：北京希望电子出版社，2006.5
（.NET开发专家）
ISBN 7-5038-4237-7

I. C... II.①易...②陈...③万... III. C语言—程序设计—指南
IV. TP312-62

中国版本图书馆CIP数据核字（2005）第113615号

出版： 中国林业出版社（100009 北京市西城区刘海胡同7号 010-66184477）
北京希望电子出版社（100085 北京市海淀区上地3街9号金隅嘉华大厦C座611）
网址：www.bhp.com.cn 电话：010-82702660（发行） 010-62541992（门市）
印刷： 北京双青印刷厂
发行： 全国新华书店经销
版次： 2006年5月第1版
印次： 2006年5月第1次
开本： 889mm×1194mm 1/16
印张： 34
字数： 837千字
印数： 0001～3000册
定价： 55.00元

前 言

在过去的 20 多年中，C 和 C++一直是商业软件和业务软件开发中使用最广泛的语言。虽然这两种语言都为程序员提供很强的系统和环境控制能力，但这种功能强大和灵活性是以牺牲工作效率为代价的。

对于 C 和 C++程序员来说，理想的解决方案将是快速开发与访问基础平台所有功能的能力相结合。他们需要一个与新出现的 Web 标准完全同步，并且可以与现有应用程序方便集成的环境。此外，C 和 C++开发人员需要底层编码（如果出现这种需要）的能力。

Microsoft 针对该问题的解决方案是提供称为 C#（发音为 C sharp）的语言。C#是一种面向对象的高级语言，程序员可以使用它来为新的 Microsoft .NET 平台快速构建范围广泛的应用程序，这种新平台提供了可以完全利用计算和通信功能的工具和方法。

C# 语言自C/C++演变而来。它在带来对应用程序的快速开发能力的同时，并没有牺牲C/C++的优点。它简化了C++在类、命名空间、方法重载和异常处理等领域的工作，摒弃了C++的复杂性，使它易使用、更少出错。C#主要特点包括：简单、面向对象、与Web的紧密结合、完整的安全性与错误处理、版本控制，以及灵活性与兼容性等。

全书总共 19 章，在内容结构上可以分为三部分：第一部分简单概述 Microsoft .NET 的技术特色、体系结构、运行环境等；第二部分比较详细地介绍 C#语言，从 C#基本语法、面向对象特性，以及异常处理、预处理等多个方面进行介绍；第三部分是 C#高级编程，分成线程、界面设计、数据访问、多媒体、COM+服务、XML、Windows 服务、Web 应用、注册表、文件管理，活动目录、系统管理和诊断，以及安全性等专题介绍如何使用 C#来实现这些高级控制。

本书内容比较全面，讲解详细，概括 C#程序设计的各个主要开发领域，同时提供大量实例，非常适合于读者阅读和实践。内容分析清晰透彻，每个例子都有专门的代码分析部分，能让读者非常容易地理解所介绍的技术和演示的范例，掌握技术要点和技巧。本书可供软件开发人员使用，也可作为大专院校 C#语言的教辅材料。

本书由易向东、陈蓓、万英和罗曼莉组织编写，其他参与本书编写、审校、输入、审查以及排版的人员包括：曾洁玫、周鸣扬、冯军、刘湛清、王强、吴秋丽、王钧、杨宏伟、陈磊、刘永军、蔡丽、郑启迪、周松建、邓欣、慈小勇、张红伟、孟康健、张小潘、张宣帝、李晓明、董彬、李生卫、张庆铭、高迎鹏、李军锋、夏兵、李红玲、马丽、李志、张巧丽，以及史阳等等。周鸣扬老师对全书进行了仔细审查和修改，使得稿件质量得到很大提升，在此对他表示感谢！

作 者

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章　.NET 基础

随着互联网时代的到来，人们逐步进入日新月异的 IT 生活。近来，一个新名词.NET 频频出现。2000 年 6 月 22 日，微软公司正式对外宣布了其.NET 战略，并确定每年为这个新的战略直接投入 40 亿美元的研发费用。. NET 这个曾经模糊不定的庞大体系迅速成为业界瞩目的焦点。2000 年 9 月，微软公司在旧金山发布了 Enterprise 2000。同月，微软原总裁兼首席执行官鲍尔默来到中国就“下一个互联网”主题进行演说，在中国掀起了一股“.NET 旋风”。2000 年 11 月，微软在 COMDEX 计算机大展上发表了 Visual Studio .NET 软件，并展示了其.NET 发展战略的框架体系和开发工具的相关特性，全面加速了微软以.NET 技术进军市场的步伐。

随着.NET 步伐的加快，身为中国的 IT 界人员，要审视度势，认清.NET 计划的本质和发展趋势。

1.1　.NET 概述

.NET 反映了微软高层人士对数字经济的深刻把握，也体现了微软的整体决策能力。Microsoft. Net 是下一代的互联网平台。

为何会称.NET 为下一代的互联网平台呢？互联网自出现始，先是以静态的网页做展现，我们将数据或文件编写成 HTML 放置在互联网上共享，这就是所谓的第一代互联网。然后，人们要求能更方便地应用程序存取这些数据，而不是让人去浏览这些数据，便有了动态网页的出现。此时的 Web 应用程序会动态地从数据库中根据应用程序或用户的要求送出相应的数据，动态地生成展现数据的网页。这个时候还是会发现能通过浏览器所做的工作只是这些基本操作而已。怎样才能打破互联网商所有应用程序、网站、网页各自的独立性，使用业界标准的 XML，在各自的应用程序、网站、网页中描述各自的数据呢？XML 为业界标准，因此数据可以在任何网站、硬件装置与应用程序间交换。微软力推 Web Service 的原因就在于此，利用互联网提供软件服务的最关键因素 XML Web Service 可以把 Web Service 想象成是一个软件组件，只是这个组件可以在互联网上供其他程序调用的组件。

随着微软. Net 的全面铺开，Windows 95、Windows 98、Windows NT 等也都将逐步成为历史。然而，就像人们今天还在回顾微软用 MS-DOS 创造第一次成功一样，这些产品的作用和功绩也都是不可磨灭的。

1.1.1　.NET 面临的竞争和挑战

在软件和技术方面，SUN 公司和 Oracle 公司自然是微软最大的竞争对手。自从微软推出.NET 策略后，SUN 公司和 Oracle 公司都不甘示弱。SUN 公司在微软推出.NET 半年以后宣布一个称作 ONE（Open Net Environment）的概念，在公众面前迎击微软的策略。Oracle 公司则以 Dynamic Web Service 的概念表示与微软抗衡。这些概念的技术取向都是大同小异的，但是从实现的可行性角度来看就不尽相同了。

正是因为未来的竞争格局一定还会发生变化，微软的竞争对手与合作伙伴也就会不断变化，谁能保证今天的竞争者在将来不会成为志同道合的合作者，甚至成为一家人呢？从某种程度上来看，以微软为代表的一些巨头将控制“世界数字”的命脉，这才是必须审慎对待和观察.NET 的原因。

1.1.2　.NET 与 J2EE 的比较

下面就 Sun 公司推出的 J2EE（Java 2 Enterprise Edition）和微软公司的.NET 在面向下一代企业计算应用方面进行比较。

J2EE 平台提供了一个基于组件的方法，用来设计、开发、装配及部署企业应用程序。J2EE 平台提供了多层的分布式应用模型、组件重用、一致化的安全模型以及灵活的事务控制；同时，保证与应用平台无关，基于组件的 J2EE 解决方案不会被束缚在任何一个厂商的产品和 API 上。所以在设计新技术的出发点上应该说.NET 和 J2EE 是非常相似的。但是这两种技术在实现方法和具体的实现技术上都有很大的甚至对立的区别。

需要指明的是，.NET 决不是简单的、改进型的 Windows 操作系统。因为按照微软的设计思想，任何一个操作平台只要安装了 CLR 就能够运行.NET 程序。

1. 开发语言

在开发语言方面，.NET 的支持面是比较广的。C++，VB，C#，Perl 和 COBOL 等语言均得到支持，开发人员可以很容易地找到合适自己的语言。而 J2EE 只支持 Java 语言。这就是说，J2EE 对语言的选择是比较窄的。当然，C#是.NET 支持的最重要的一种语言。相对于 Java 而言，C#支持 JIT（just-in-time）编译方式，而 Java 基于解释方式。同时，微软为不同的平台环境提供了不同的 JIT 编译方式。对于类似 Windows CE 这样的移动计算环境，微软提供了压缩的.NET 框架，也提供了 EconoJIT（经济型编译器）。在一般的桌面环境下，微软提供了标准的编译器。另外，C#将成为一种工业标准，因为 ECMA（欧洲计算机制造商协会）正在接纳 C#，而 Java 语言只由 Sun 公司提供。

2. 支持的标准

J2EE 支持 Java 和 EJB，而.NET 支持 XML/SOAP。就标准的开放性来说，XML/SOAP 好于前者。XML 由 W3C 组织提出，得到众多厂家支持，是下一代因特网上内容表示的标准。XML 能够有效地表达网络上的各种知识，为信息的交换和计算提供新的载体。XML 相对于网络计算的作用，完全可以与计算机起步阶段 ASCII 码的作用相提并论。也可以说 XML 是网络信息的标准代码，它表示的不是符号信息，而是知识化的块状内容。这种标准语言虽然不是程序员设计语言，但是它代表下一代网络上互操作的光明前景。说到这里，不由得让人们想起了“当年”对 Java 的狂热，Java 确实有着非常诱人的初衷，让许多人能够在这样一种理想的感召下为想象中的各种系统之间的互操作能力而投入积极的开发中。但是，实际上 Java 既没有成为人们想象中的成功的商业计算工具，也没有实质上的技术进步。Java 试图从统一平台的角度来实现互操作，但是这可能永远只是一个梦想。真正能够互操作的只能是标准和通用的数据描述语言。SOAP 协议本身也是由微软和 IBM 这样的商家联合推出的。这一切都表明.NET 技术标准的开放性是不错的。. NET 的 SOAP 协议保证一个平台上的组件能够与.NET 平台上的组件进行信息的交换。

3. 跨平台

在现有的条件下，各种各样所谓的跨平台或“编译一次，多处运行”等口号只是商业炒作。Java 的首席设计师 James Goslin 在谈到这个问题时曾经表达过这个看法，所谓的“编译一次，多处运行”只是一种美好的想法。这就是说，基于某种开发平台进行开发是不可避免的。假如用户基于 IBM 公司的 WebSphere 利用 Java 开发商业程序，基本上就固定在这个平台上了。Java 号称的“100%纯”的口号与现实不符。当然，C#也是如此。

在.NET 平台上开发程序的一个重要好处在于可以真正实现“代码重用”。因为在设计.NET 平台时，一个重要的思想就是运行时和具体的语言分开。所有的资源管理、内存分配和变量类型等均由运行时处理。用 C#写的类就直接可以用在 C/C++程序中。只要基于.NET 平台，过去的程序不会因为采用的语言种类不同而作非常大的修改。而 J2EE 平台上只能用 Java 来开发程序，运行时和具体的语言混在一起。

总而言之，J2EE 和.NET 各有各的优点和缺点。两者都是很优秀的开发企业计算软件的平台。但就像不同的人有不同的嗜好一样，在未来的开发中，还要根据自己的具体需要来确定具体的开发平台。

1.2 .NET 技术体系

家喻户晓的微软公司是一个让无数竞争对手心惊胆战的软件帝国。Microsoft .NET 的推出再一次证明了微软公司在软件世界的霸主地位。微软自揭开.NET 的序幕之后，便马不停蹄地一波接着一波推出许多技术。对于微软而言，.NET 是微软公司的一场“圣战”，“.NET 是微软有史以来最大的赌注”。

对于.NET 到底是什么这个问题，有着各种各样的说法。但当时作为首席执行官的鲍尔默应该最能代表微软公司的观点，他说：“Microsoft .NET 代表了一个集合、一个环境、一个可以作为平台支持下一代 Internet 的可编程结构。”确实，这句话基本上简单扼要地表述了.NET 的外特性。

.NET 最根本的精神在于改革分布式电脑运算的面貌，把.NET 称为策略名词比称它为技术名词更加恰当。.NET 彻底地把计算模式从单机、客户机/服务器和 Web 网站的方式转向分布式计算(Distributed Computing)。我们知道 COM 和 Corba 是今天比较流行的部件对象计算模型。但是它们都存在“局部计算”的局限性。也就是说，这些模型都仅仅是本地计算或本网计算的模式，不是把整个互联网当作是一个计算资源体系来加以利用。.NET 则通过一种称作“网络服务”（Web Service，这是.NET 的核心概念）的技术把分布在互联网上的各

种资源有效地通过编程手段整合在特定的应用界面。届时，所有的软件开发商亦可以很容易地借由.NET 的结构，通过互联网提供软件服务。这就好像是现在的电力公司负责发电并通过电线传送电力给我们，然后按照我们用电的度数收取费用。过去，软件开发商必须将软件包装在光盘或文件中提供给客户，会遇很多问题，如软件的新旧版本软件的盗版等。总之，Web 服务是.NET 的核心。

除了 Web Service 这种新型计算机模式，微软还在.NET 中增加了自己已有的研究成果，如自然语言的处理和识别等。这些技术能让用户更加灵活地操作各种计算设备，而不必依赖于传统的鼠标和键盘。自然语言的处理还能提高计算机自行处理各种 XML 信息的智能性。总之，.NET 包含了新一代的计算模式，即跨越全球的分布式计算。这种规则的制定者将有可能从与之配合的商业模式中大获其利。

1.2.1 .NET 技术特征

.NET 具有 4 个重要特点，分别为：基于 XML 的共同语言，定制服务软件，融合多平台和设备及其新一代的人机界面。这 4 个特点基本上覆盖了.NET 的技术特征。

1. 基于 XML 的共同语言

XML 是从 SGML 语言演化而来的一种标记性语言。用作元语言，它可以定义不同种类的应用的数据交换语言。在.NET 体系结构中，XML 用作一种应用间无缝结合的手段，用于多种应用之间的数据采集与合并，用于不同应用之间的互操作和协同工作。具体而言，.NET 通过 XML 语言定义了简单对象访问协议（SOAP）、Web 服务描述语言（WSDL）和 Web 服务发现协议（DISCO）。SOAP 协议提供了在无中心分布环境中使用 XML 交换结构化有类型数据的简单轻便的机制。WSDL 协议定义了服务描述文档的结构，如类型、消息、端口类型、端口和服务本身。DISCO 协议定义了如何从资源或者资源集合中提取服务描述文档和相关服务发现算法等。

2. 定制服务软件

微软首席执行官史蒂夫·鲍尔默在谈到软件服务时说，今天的软件产品仅仅是一张光盘，用户购买软件、亲自安装、管理和维护。但是软件服务是来自因特网的服务，它替用户安装、更新和跟踪这些软件，并让它们和用户一起在不同的机子间漫游。它为用户存储自己的信息和参考资料。这些就是软件和技术特征。

图 1-1 显示了.NET 定制服务软件的技术特性。

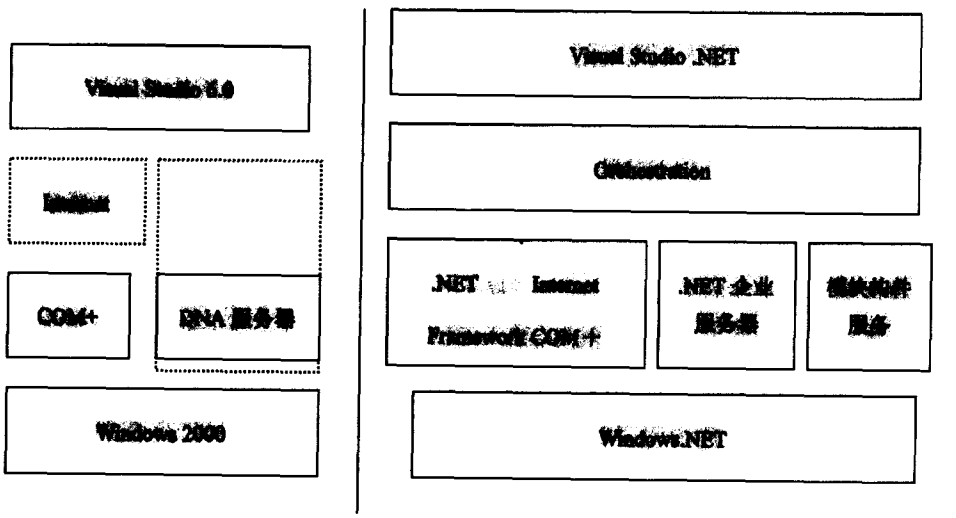

图 1-1 .NET 的技术特征之一：定制服务软件

Orchestration 可视化编程工具产生基于 XML 的 XLANG 代码，它和 BizTalk 服务器、.NET Framework 以及 Visual Studio .NET 都曾是 Windows DNA 2000 战略的重要部分。

伴随着 ASP 产业的兴起，软件正逐步从产品形式向服务形式转化，这是整个 IT 行业的大势所趋。在.NET

此为试读，需要完整PDF请访问：www.ertongbook.com

中，最终的软件应用是以 Web 服务的形式出现并在因特网发布的。Web 服务是一种经过包装的可以在 Web 上发布的组件。. NET 通过 WSDL 协议来描述和发布这种 Web 服务信息，通过 DISCO 协议来查找相关的服务，通过 SOAP 协议进行相关的简单对象传递和调用。

微软的.NET 战略意味着：微软公司以及在微软平台上的开发者将会制造服务，而不是制造软件。在未来几年之内，微软将陆续发布有关.NET 的平台和工具，用于在因特网上开发 Web 服务。那时，工作在.NET 上的用户、开发人员和 IT 工作人员都不再购买软件，安装软件和维护软件了。取而代之的是，他们将定制服务，软件会自动安装，所有的维护和升级也会通过互联网进行。

3. 融合多平台和设备

随着因特网逐渐成为一个信息和数据的中心，各种设备和服务已经或正在接入和融合因特网，成为其中的一部分。.NET 谋求与各种因特网接入设备和平台的一体化，主要关注无线设备和家庭网络设备以及相关的软件和平台。

4. 新一代的人机界面

.NET 的新一代人机界面特征主要体现在“智能与互动”两个方面，包括通过自然语音、视觉、手写等多种模式的输入和表现方法，基于 XML 的可编辑复合信息构架——通过画布个性化的信息代理服务及使机器能够更好地进行自动处理的智能标记等技术。

.NET 的平台及框架是基于微软软件工业基础的又一次升级和演化。然而，还是要尽力保证 Windows 系统及系列产品和.NET 能够融为一体，尽量在微软公司原有的软件资产基础上使.NET 继续成为因特网的中心。

1.2.2 .NET 新特性

.NET 是一种全新的技术，因此.NET 中包括了很多新特性。这里只列出了一些比较重要的特性。

.NET 新特性包括一致的编程模式，简化的编程模式，运行于多个平台，支持多语言的综合，自动资源管理，一致的出错处理方式，安全性和 XML 语言的引入等。

- 一致的编程模式。在.NET 环境中，所有的应用程序都采用通用的面向对象的编程模式，而 Windows 环境中既有 DLL 函数也有 COM 对象。
- 简化的编程模式。这是令开发者最欢迎鼓舞的消息，在.NET 环境下，由于 CLR 的作用，在进行编程时不再需要掌握 GUIDs，Iunkown 和 AddRef 等令人头疼的 COM 知识了。
- 运行于多个平台。对于任何操作平台，只要支持.NET 运行时均可以运行.NET 应用程序。现在所有的 Windows 平台均可以实现这一点，将来甚至在非 Windows 操作系统上也可以实现这一点。
- 支持多语言的综合。按照 COM 的原理，代码重用是建立在二进制代码的级别上的。在.NET 环境下，代码重用可以建立在源码级别上，也就是说，别人用 C#语言写的某个类可以直接在 C＋＋这样的语言中使用。.NET 有这样的巨大威力在于它为所有支持.NET 编程的语言提供了一整套通用类型系统。
- 自动资源管理。对于所有开发人员而言，最头疼的就是内存的处理问题。在.NET 环境下，这个问题得到彻底解决，自动资源管理功能已经纳入 CLR 之中。同时，由于增加了资源回收功能，在一定程度上安全性也得到了保障，诸如内存溢出、攻击等将得到有效控制。
- 一致的出错处理方式。相信所有的 Windows SDK 程序员都对 Windows 环境下混乱的错误处理方式感到厌烦，如 Win32 错误代码、异常情况处理和 HRESULT 等。在.NET 环境下，所有的程序都采用统一的错误处理方式（产生异常）。
- 安全性。如前所述，.NET 的出现是为了迎合下一代因特网环境下的企业级计算，一般的访问控制已经不能满足要求，所以在安全性方面，.NET 相对于 Windows 等其他系统而言，有了更深入的改进，如从装载一个类开始，就进行确认检查；在访问代码和相应资源时，实施代码访问安全措施。.NET 还提供了一整套机制来判断角色和确认身份信息，并且能做到跨进程和跨机器，从而确保所需的代码在远端不会受到破坏。.NET 的安全性也深深嵌入到 CLR 结构中，以确保应用程序本身安全。这些安全机制是对现有操作系统安全机制的一种本质上的扩展，从而使.NET 在安全性上进一步加强。

- XML 和 SOAP 的引入。回忆一下过去的分布式应用程序的设计，通常设计两层应用程序，在此基础上出现了诸如 CORBA，IIOP，RMI 和 DCOM 这样的协议。人们已经熟悉了这样的分布式系统。但是这种系统的弊端就是灵活性差，因为这种设计方式使得应用程序固定在服务器端。而因特网是整个松散连接和分布非常广的世界。原有的 Client/Server 结构已经过时，因此就提出了全新的编程模式，而 XML 和 SOAP 能使这种模式很好地工作。在.NET 中，XML 和 SOAP 已经深深地融入其中并成为非常重要的组成部分。

1.2.3 .NET 组成

.NET 主要由 Windows .NET，.NET Framework，模块构建服务和 Visual Studio .NET 组成。下面介绍这些组成部分的主要内容。

1. Windows .NET

Windows .NET 是融入.NET 技术的 Windows，它紧密地整合了.NET 的一系列核心构造模块，为数字媒体的应用、协同工作提供支持，是微软公司的下一代 Windows 桌面平台。

2. .NET Framework

.NET Framework 的设计目的是便于开发商更容易地建立网络应用程序和 Web 服务，它的关键特色是提供了一个多语言组件开发和执行环境。从层次结构来看，.NET Framework 又包括了 3 个主要组成部分：公共语言运行时（CLR，即 Common Language Runtime）、服务框架（Service Framework）、上层的两类应用表单——面向 Web 的网络应用程序表单（Web Form 或 Web Service）和 Windows 应用程序表单（WinForm）。

公共语言运行时，负责管理内存分配，启动和终止线程和进程，强化安全系数，同时还调整组件涉及的任何其他组件、附件配置。在公共语言运行时层次之上是服务框架，它为开发人员提供了一套能够被任何现代编程语言调用的、统一的、面向对象的、异步的、层次结构的可扩展类库，包括集合、输入/输出、字符串、网络、线程、全球化、安全加密、数据库访问和调试有关服务等库类。在服务框架层次之上是两种应用类型的表单，一类是传统的 Windows 应用程序表单，另一类是基于 ASP .NET 的 Web 网络应用程序表单。其中 ASP .NET 以一组控件和体系结构的方式提供了一个 Web 应用模型，由.NET 框架提供的类库构建而成，通过它可以简化 Web 应用的实现过程。

3. 模块构建服务

模块构建服务（Buliding Block Services）是.NET 平台中的核心网络服务集合。它主要包括以下几个组成部分：因特网 XML 通信（使 Web 站点变成灵活的服务来交换和处理数据）、因特网 XML 数据空间（在 Web 上提供安全的和可编程的 XML 存储空间）、因特网动态更新（为快速开发和动态配置应用提供服务）、因特网日程安排（集成工作、社会和私人的日历）、因特网身份认证（提供从密码到生理数据等多级身份认证手段）以及因特网目录服务和因特网即时信息传递等服务。

4. Orchestration

Orchestration 是基于 XML 的面向应用的软件集成和自动化处理技术。它的目标是不受时间、组织、应用及个人的限制，最大程度和最好地把集成技术和自动化技术结合起来，以便商业事务能够交互并动态、可靠地进行下去。Orchestration 有 3 个基本要求：处理与执行过程分离，即整个处理并不一定非要执行的细节及途径绑定；动态处理，即随着整个处理过程不能对参与的平台、应用及协议等提出限制。.NET 的 BizTalk Orchestration 是上述技术的一个实现，它包括一个可视化的设计环境、一套捆绑的工具和一个 Orchestration 引擎，用于业务流程的处理、管理和调试。

5. Visual Studio .NET

Visual Studio .NET 是基于 XML 编程工具和环境的，它便于快速开发，符合.NET 体系软件服务，使其在独立设备、企业数据中心和因特网之间传送更加容易。

1.3 公共语言运行时

公共语言运行时（CLR）是整个 Microsoft .NET 框架赖以建构的基础，它为 Microsoft .NET 应用程序提供了一个托管的代码执行环境。它实际上是驻留在内存里的一段代理代码，负责应用程序在整个执行期间的代码管理工作，比较典型的有：内存管理、线程管理、安全管理、远程管理、即时编译、代码强制安全类型检查等。这些都可称得上 Microsoft .NET 框架的生命线。

实际上，CLR 代理了一部分传统操作系统的管理功能。在 CLR 下的代码称之为托管代码，否则称为非托管代码。也可将 CLR 看作一个技术规范，无论程序使用什么语言编写，只要能编译成微软中间语言（MSIL），就可以在它的支持下运行，这使得应用程序得以独立于语言。目前，支持 CLR 的编程语言多达二、三十种。微软中间语言是在 Microsoft .NET 平台下编译器输出的 PE 文件的语言。它是 Microsoft .NET 平台最完整的语言集，非常类似于 PC 机上的汇编语言。即时编译器在运行时将中间语言编译成本地二进制代码。

CLR 的设计目的便是直接在应用程序运行环境中为基于组件的编程提供支持。CLR 直接支持组件（包括属性和事件）、对象、继承性、多态性和接口。对属性和事件的直接支持使得基于组件的编程变得更简单，而不需要特殊的接口和适配设计模式。在组件运行时，CLR 负责管理内存分配、启动和中止线程和进程、强化安全系数，同时还调整任何该组件涉及到的其他组件的附属配置。序列化支持允许以多种格式操作存储在磁盘上的组件，包括基于业界标准 XML 的 SOAP。CLR 提供了处理错误条件的有力、协调的方式。每个模块都具有内置的完整的元数据，这意味着诸如动态创建和方法调用之类的功能更容易，也更安全。映射甚至允许灵活地创建和执行代码。

可以控制应用程序使用的组件的版本，这使应用程序更加可靠。组件代码是与处理器无关的和易于验证的中间语言（IL），而不是某一种特定的机器语言，这意味着组件不但可以在多种计算机上运行，而且可以确保组件不会覆盖它们不使用的内存，也不会潜在地导致系统崩溃。CLR 根据托管组件的来源（如来自因特网，企业局域网，本地机）等因素对它们判定以适当的信任度，这样 CLR 会根据信任度来限定它们的执行，如读取文件、修改注册表等某些敏感操作的权限。借助公共类型系统（Common Type System，简称 CTS）对代码类型进行严格的安全检查避免了不同组件之间可能存在的类型不匹配的问题。CLR 下的编程全部是围绕组件进行的。

1.3.1 中间语言（IL）和元数据

为了管理语言的执行过程，.NET 框架中的程序并不编译成机器码，而是编译成微软公司定义的中间语言（MSIL，Microsoft-defined intermediate language）。MSIL 类似于机器指令，但独立于任何特定处理器体系结构，包含其他支持面向对象技术的特性。

MSIL 是将.NET 代码转化为机器语言的一个中间过程。它是一种介于高级语言和基于 Intel 的汇编语言的伪汇编语言。当用户编译一个.NET 程序时，编译器将源代码翻译成一组可以有效地转换为本机代码且独立于 CPU 的指令。当执行这些指令时，实时（JIT）编译器将它们转化为 CPU 特定的代码。由于公共语言运行时支持多种实时编译器，因此同一段 MSIL 代码可以被不同的编译器实时编译并运行在不同的结构上。

IL 和元数据存放在扩展了.exe 和.dll 文件所使用的 PE 格式的文件中。当这些 PE 文件被装入时，运行环境会从中找到并取出元数据和 IL。

1.3.2 即时编译器（JITters）

由 C#或其他能产生托管代码的编译器所产生的托管代码就是 IL 码。虽然 IL 代码被包装在一个有效的 PE 文件中，但是还是不能执行它，除非它被转换成为托管原始代码。这就是 NGWS runtime 即时编译器（也称作 JITters）大显身手的时候。

为什么会对即时编译代码感到厌烦，为什么不把整个 IL PE 文件编译成原始代码？答案是需要时间——需要把 IL 代码编译成 CPU 规格代码的时间。这种编译将更加有效率，因为一些程序段从来就没有被执行过。例如，在我的 Word 处理器中，邮件合并功能从来就没有被编译。

从技术上说，全部的处理过程如下：当一个类型被装载时，装载器创建一个存根（stub），并使它连接每一个类型的方法。当一个方法第一次被调用时，存根把控制交给 JIT。JIT 把 IL 编译为原始代码，且把存根

指针指向缓冲了的原始代码。接着调用将执行的原始代码。在某些位置上（At some point），所有的 IL 都被转换成为原始代码，而 JITter 处于空闲状态。

JIT 编译器有很多，不止一个。在 Windows 平台上，NGWS runtime 装有 3 个不同的 JIT 编译器。

- JIT——这是 NGWS runtime 默认使用的 JIT 编译器，它是优化编译器的后端。在前台（up front）实行数据流分析，并创建了高度优化的托管原始代码作为输出结果。JIT 可以使用不严格的 IL 指令集编码，但是所需资源十分可观。主要的限制在于内存足迹（footprint）、结果工作集以及实行优化所消耗的时间。
- EconoJIT—— 和主 JIT 相比，EconJIT 的目标是把 IL 高速地转换成托管原始代码。它允许缓冲所产生的原始代码，但是输出码并不像主 JIT 生成的代码那样优化（代码小）。当内存紧张时，这种快速代码生成方案的优势就显示出来了——通过永久地丢弃无用的编译码的方式，使代码高速缓存区能适应更大的 IL 程序。因为 JIT 编译快，执行速度也相当的快。
- PreJIT——尽管它是基于主 JIT 的，但操作起来更像是一个传统的编译器。安装了 NGWS 组件，它才能运行，才可以把 IL 代码编译成托管原始代码。当然，最终的结果为，更快的装载时间和更快的应用程序启动时间（不需要别的 JIT 编译器）。

1.4 虚拟对象系统

到目前为止，只看见 CLR 是如何运作的，而没有了解它的技术背景。下面就来讲讲 NGWS 虚拟对象系统（VOS）。

CLR 声明，使用和管理类型的准则是在虚拟对象系统中定制的。设计虚拟对象系统的目的是建立一个框架，允许跨语言集成和类型安全，并且不牺牲执行效率。为了更好地理解这个框架，首先介绍一下开发 C# 应用和组件时很重要的 4 个方面：VOS 类型系统、元数据、公共语言规范以及虚拟执行系统。

1. VOS 类型系统

.NET 跨语言集成的特性来自于虚拟对象系统（VOS）的支持。VOS 提供了一个丰富的类型系统，用来支持各种不同编程语言的完全实现。所以，VOS 不仅支持面向对象编程语言，还支持过程编程语言。

现在，存在很多种近似但有点不兼容的类型。就拿整型来说，在 VB 中，它是 16 位长；而在 C++中，它是 32 位。还有更多的例子，特别是用在日期和时间以及数据库方面的数据类型。这种不兼容使得应用程序的创建和维护复杂化，尤其当程序使用了多种编程语言时。

另一个问题是，因为编程语言之间存在一些差异，不能在一种语言中重用另一种语言创建的类型。（COM 用二进制标准接口部分地解决了这个问题）。但当今代码重用肯定是有限的。

分布应用程序的最大障碍是各种编程语言的对象模型不统一。几乎每一方面都存在差异：事件、属性、永久保存（persistence）等。

VOS 将改变这种现象。VOS 定义了描述值的类型，并规定了类型的所有值必须支持的一条合约。前面提到的支持面向对象和过程编程语言，就存在着两种值和对象。对于值，类型存储于表述（representation）中，操作也在其中执行。对象更强大，因为它显式存于表述中。每个对象都有一个区别于其他对象的识别号。

2. 元数据

元数据是 VOS 中类型描述代码的一种称呼，在编译程序时将源代码转换成中间代码时将自动生成，并与编译后的源代码共同包含在二进制代码文件中。元数据携带了源代码中类型信息的描述。在 CLR 定位与装载类型时，系统通过读取并解析元数据来获得应用程序中的类型信息。JIT 编译器获得加载的类型信息后，将中间语言代码译成本地代码，在此基础上根据程序或用户要求建立类型的实例。由于整个过程中，CLR 始终根据元数建立并管理对应特定应用程序类型，从而保证了类型安全。元数据在解决方法调用、建立运行期上下文界限等方面都有着自己的作用，而关于元数据的一切都由.NET 在后台完成。

对于一种更简单的编程模型来说，元数据是关键，该模型不再需要接口定义语言、（IDL）文件、头文件或任何外部组件引用方法。元数据允许 .NET 语言自动以非特定语言的方式对其自身进行描述，而这是开发

人员和用户都无法看见的。另外，通过使用属性可以对元数据进行扩展。综上所述元数据具有以下主要优点：

- 描述文件。公共语言运行时模块和程序集是自描述的。模块的元数据包含与另一个模块进行交互所需的全部信息。元数据自动提供 COM 中 IDL 的功能，允许将一个文件同时用于定义和实现。运行库模块和程序集甚至不需要向操作系统注册。结果，运行库使用的说明始终反映编译文件中的实际代码，从而提高应用程序的可靠性。
- 语言互用性和更简单的基于组件的设计。元数据提供所有必需的有关已编译代码的信息，以供你从用不同语言编写的 PE 文件中继承类。可以创建用任何托管语言（任何面向公共语言运行时的语言）编写的任何类的实例，而不用担心显式封送处理或使用自定义的互用代码。
- 属性。.NET Framework 允许在编译文件中声明特定种类的元数据（称为属性）。在整个 .NET Framework 中到处都可以发现属性的存在，属性用于更精确地控制运行时的程序如何工作。另外，可以通过用户定义的自定义属性向 .NET Framework 文件发出自定义的元数据。

3. 公共语言规范

CLS（公共语言规范）是 CLR 定义的语言特性的集合，主要用来解决互操作问题。如果一个类遵守 CLS，那么同样遵守 CLS 规范的其他编程语言，将能够使用它的外部可见项。

公共语言规范并不真正是虚拟对象系统（VOS）的一部分，它是特殊的。CLS 定义了 VOS 中的一个类型子集，也定义了必须符合 CLS 的常规用法。

这有什么用呢？如果一个类库遵守 CLS 规则，其他编程语言同样也遵守 CLS 规则，那么其他编程语言的客户也可以使用类库。CLS 是关于语言的交互可操作性（interoperability）。因此，常规用法必须仅遵循外部可访问项目（externally visible items），如方法、属性、事件等。

它的优点是可以进行这样的操作：用 C#写一个组件，在 VB 中由它导出某些结构，为了利用 VB 中的强大功能，可以再从 C#中由 VB 结构导出类。只要所有的外部可访问项遵守 CLS 规则，这一过程是可行的。

CLS 在设计上足够大，可以包括开发人员经常需要的语言构造；同时也足够小，大多数语言都可以支持它。此外，任何不可能快速验证代码类型安全性的语言构造都被排除在 CLS 之外，以便所有符合 CLS 的语言都可以生成可验证的代码。

表 1-1 为给类型和外部可访问项定义的协定规则。这个清单不完整，它仅包含一些很重要的项目。这里不指出本书中出现的每一种类型的 CLS 格式。浏览该表，有助于寻找遵从 CLS 的格式。表中有些项将会在本书中逐步学习。

表 1-1 公共语言规范中的类型和特征

基本类型
bool char byte short int long float double string object(所有对象之母)
(Types) 类型
可以被抽象或隐藏。 零或更多的接口可以被实现。不同的接口允许拥有具有相同名字和签名的方法。 一个类型可以准确地从一个类型派生。允许成员被覆盖和被隐藏。 可以有零或更多的成员，它们是字段(fields)、方法、事件或者类型。 类型可以拥有零或更多个构造函数。 一种类型的可访问性可以是公用的或者对 NGWS 组件来说是局部的；但是，仅公用成员可以认为是类型接口的一部分。 所有的值型必须从系统值型继承。异常是一个枚举——它必须从系统枚举(System Enum)继承。
Arrays(数组)
数组的维数必须是已知的(>=1)，而且最小下标数必须为 0。 要素类型必须是一个 CLS 类型。

（续表）

基本类型
类型成员
类型成员允许隐藏或者覆盖另一种类型中的其他成员。 参数和返回值的类型都必须是 CLS 协定 类型。 构造函数、方法和属性可以被重载。 一个类型可以有抽象成员，但仅当类型不被封装时。
基本类型
方法
方法可以是静态、虚拟或者实例。 虚拟和实例方法可以是抽象的，或者是一个实现。静态方法必须总拥有一个实现。 虚拟方法可能是最后的(或者不是)。
字段(Fields)
可以是静态或者是非静态。 静态字段可以被描述或只初始化。
属性
当获取和设置方法而不是使用属性语法时，属性可以公开。 获取的返回类型和设置方法的第一个参数必须是相同的 CLS 类型——属性的类型。 属性名字必须不同，不同的属性类型用于区分是不充分的。 由于使用方法实现属性访问，如果 PropertyName 是同一个类中定义的一个属性，你不能实现命名为 get_PropertyName 和 set_PropertyName 的方法。 属性可以被索引。 属性访问必须遵循这种命名格式：get_ProName，set_PropName。
枚举(Enumerations)
强调类型必须是 byte、short、int 或 long。 每一个成员是一个枚举类型的静态描述字段。 一个枚举不能实现任何接口。 你允许给多字段设定相同的值。 一个枚举必须继承系统枚举(隐含在 C#中)。
接口
可需要实现其他接口。 一个接口可以定义属性、事件和虚拟方法。实现取决于派生类。
事件
增加和取消方法必须是都提供或者都没有，每一种方法采用一个参数，它是一个从系统代表元(System Delegate)派生下来的类。
自定义属性
可以仅使用下列类型：Type(类型),char, char, bool, byte, short, int, long, float, double, enum (一种 CLS 类型), 和 object。
代表(Delegates)
可以被创建和被激活。
标识符(Identifiers)
一个标识符的第一个字母必须来自一限制集。 通过大小写在单一范围内，不可能唯一地区别两个或更多个标识符(大小写不敏感)。

4. 虚拟执行系统(VES)

虚拟执行系统（VES）是 VOS 的实现，用来驱动运行环境。元数据的生成与使用、公共语言规范的满足性检查以及应用程序执行过程的内存管理均由它来完成。它主要完成以下功能：

- 装入中间代码
- 使用 JIT 将中间代码转为本地码
- 装入元数据

此为试读，需要完整PDF请访问：www.ertongbook.com