



Microsoft.NET 编程语言

C#教程

北京希望电子出版社 总策划
沉 舟 崔培枝 何荣辉 王天章 编 写



本光盘内容包括：

1. 本版书程序源代码
2. 本版电子书



C#



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



Microsoft.NET 编程语言

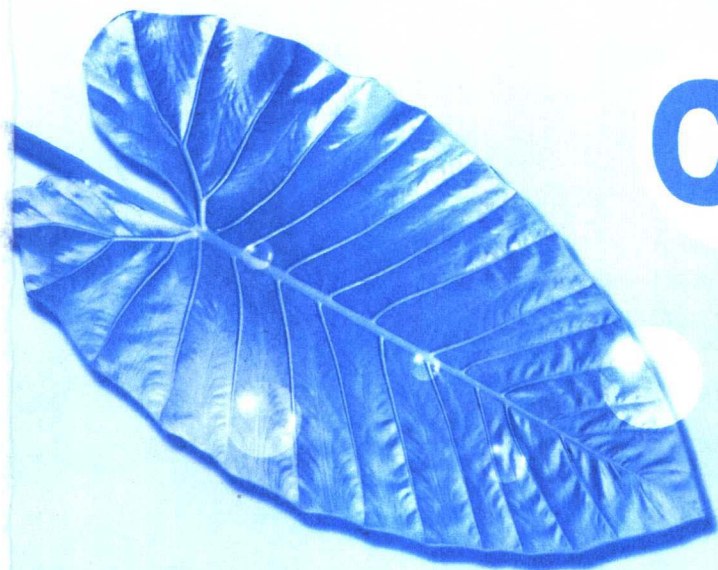
C#教程

北京希望电子出版社 总策划
沉 舟 崔培枝 何荣辉 王天章 编 写

本光盘内容包括：



1. 本版书程序源代码
2. 本版电子书



C#



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

内 容 简 介

本书是微软技术培训统编教材之一，对编程语言 C#进行了详细的介绍。

C#是基于微软下一代操作平台 .NET 的全新的面向对象的程序设计语言。它在保持了 C++中熟悉的语法的同时，添加了大量的高效代码和完全面向对象特性，以及更高的可靠性和安全性。C#语言将在保持 C/C++灵活性的基础上为程序员带来更高效的开发方式。它不仅能用于 Web 服务程序的开发，并且还能开发强大的系统级程序。

本书由五部分、18 章和 3 个附录组成。第一部分对 C#语言进行了概述，由 3 章组成：.NET 编程语言 C#、.NET 的运行环境及编写第一个应用程序；第二部分介绍了 C#程序设计基础，由 5 章组成：数据类型、变量和常量、类型转换、表达式及流程控制；第三部分介绍了面向对象的 C#语言，由 6 章组成：面向对象的程序设计、类、方法、域和属性、事件和索引指示器；第四部分对 C#进行了深入的介绍，由 4 章组成：接口、组织应用程序、文件操作及高级话题；第五部分由三个附录组成。

作者根据多年计算机教学经验，对全书体系进行了合理、严密地组织。本书内容简明扼要、概念清晰、例题丰富、实用性强。

本书不仅是广大程序设计爱好者、软件开发人员重要指导书，也是高等院校师生教学、自学参考书，也可作为各类计算机培训班教材使用。

本光盘内容包括本书程序源代码及本版电子书。

系 列 书： 微软技术培训统编教材

书 名： Microsoft .NET 编程语言 C#教程

总 策 划： 北京希望电子出版社

文本著作者： 沉舟 崔培枝 何荣辉 王天章编写

CD 制 作 者： 希望多媒体创作中心

CD 测 试 者： 希望多媒体测试部

责 任 编 辑： 郭淑珍 陆卫民

出版、发行者： 北京希望电子出版社

地 址： 北京中关村大街 26 号 100080

网址： www.bhp.com.cn E-mail: lwm@.hope.com.cn

电话： 010-62562329,62541992,62637101,62637102,62633308,62633309

(发行和技术支持)

010-62613322-215 (门市) 010-62547735、62629581 (编辑部)

经 销： 各地新华书店、软件连锁店

排 版： 希望图书输出中心

CD 生 产 者： 北京中新联光盘有限责任公司

文本印刷者： 北京媛明印刷厂

规格 / 开本： 787 毫米×1092 毫米 1/16 18.75 印张 438 千字

版次 / 印次： 2001 年 4 月第 1 版 2001 年 4 月第 1 次印刷

本 版 号： ISBN 7-900056-43-2/TP•42

定 价： 28.00 元 (1CD, 含配套书)

说明：凡我社光盘配套图书若有自然破损、缺页、倒页、脱页，本社负责调换。

前 言

有人说, .NET 将彻底改变软件的开发方式、发行方式和使用方式。 .NET 作为第三代 Internet 产品与 Java 有着极大不同, 用户数据生存于网络, 而不只是生存于不同平台。 .NET 允许不同应用程序之间能互相传递信息, 而 Java 的 beans 是不能和其他语言建立的构件共享数据的。 .NET 是一个中立语言, 基于 XML(Extensible Markup Language)和 SOAP(Simple Object Access Protocol), 允许开发者使用各自不同语言来实现它所有的功能, 所以可以肯定, 在网络飞速发展的今天, .NET 将很快成为主流。

C#(读作 C sharp) 作为 Microsoft 的下一代面向对象的语言产品, 能够让开发人员在 .NET 平台上快速地建立大量的应用程序。按照微软给出的定义, C#(读作“C Sharp”)是一种源自于 C 和 C++的, 简单的、现代的、面向对象的和类型安全的程序设计语言。C# 为程序员提供了开发飞速发展的 Web 应用程序所需的强大而灵活的功能。C#和 Java 的核心与 C++比较有着相同的优势和局限, 比起 C++, C#将更容易被人们理解和接受, 未来大量 .NET 平台的应用将由 C#开发。

C#将是未来开发企业级分布式应用程序的首选。当然这并不是说 C#只适用于网络开发。例如, 我们可以使用 C#开发任何古典风格的 Windows 程序。不仅如此, 不管是控制程序、图形程序、NT 服务程序, 还是普通组件, 乃至 Web 页面, 都可以用 C#开发出来。可以相信, 在阅读完本书之后, 不论您是一名 C 和 C++程序员, 或者您一直在使用 VB 或 Delphi 进行编程, 您都将欣赏 C#为我们所带来的一切。

全书分为几个部分来讲述 C#: 第一部分为 C#语言概述, 从一门新的软件语言的角度出发, 简要地介绍了 C#语言推出的目的、C#面向的开发环境、它所具有的特点以及为软件开发人员带来的巨大优势。第二部分全面细致地讲解了 C#语言的基本语法, 帮助读者由浅入深地掌握 C#程序设计的基本技术。第三部分系统地介绍了 C#面向对象的程序设计技术。第四部分是对 C#语言更为深入的探讨, 这将有助于读者深入了解 C#这门程序设计语言。

全书采用循序渐进的方式, 帮助读者从入门到真正精通 C#语言。在书中, 我们并不是把 C#仅仅作为一种简单的程序设计语言来对待, 而是侧重于分析 C#的语法现象以及它真正面向对象的特征。对于 C#程序设计涉及到的一些系统理论和方法学, 我们也简明扼要地进行了阐述。书中提供了大量的示例源代码, 通过这些实例, 避免了空洞的语法规则罗列和简单的说教。每一章在结束时还进行了小结, 对该章的内容进行了精辟的概括, 同时指出了难点和重点所在。每一章最后还提供了一定数量的复习题, 建议读者认真地去完成这些复习题, 通过它们, 您将得心应手地开发属于自己的 C#程序。

尽管 C#是从 C 和 C++中发展起来的, 但本书并不要求读者具有 C 或 C++语言的基础知识。无论您是其它程序设计语言的程序员, 或者刚刚开始编程生涯, 本书都将帮助您顺利进入 C#的大门。对已经熟练掌握 C++程序设计的专业程序员来说, 您将通过本书迅速过渡到新一代面向对象的语言——C#。

本书是全体编著人员集体合作的结晶。书中第一部分由沉舟负责编写, 第二部分由崔培枝、何荣辉和王天章负责编写, 第三部分和第四部分由沉舟负责编写。全书由沉舟负责统稿, 何荣辉对书中的例程进行了验证。在本书的编写过程中, 得到了希望电子出版社的陆卫民先生的热情帮助, 他对本书的内容编排提出了许多宝贵的意见, 使全书的结构更为完善, 在此向他和所有给予鼓励和支持的同志致以最衷心的感谢。

目 录

第一部分 C#语言概述

第一章 .NET 编程语言 C#	1
1.1 Microsoft.NET——一场新的革命	1
1.2 .NET 与 C#	3
1.3 C#语言的特点	5
1.4 小结	8
第二章 运行环境——全面了解 .NET	9
2.1 .NET 结构	9
2.2 公用语言运行时环境与公用语言规范	10
2.3 开发工具	14
2.4 小结	16
第三章 编写第一个应用程序	17
3.1 Welcome 程序	17
3.2 代码分析	17
3.3 运行程序	20
3.4 添加注释	22
3.5 小结	24

第二部分 C#程序设计基础

第四章 数据类型	25
4.1 值类型	25
4.2 引用类型	30
4.3 装箱和拆箱	36
4.4 小结	39
第五章 变量和常量	41
5.1 变量	41
5.2 常量	43
5.3 小结	44
第六章 类型转换	45
6.1 隐式类型转换	45
6.2 显式类型转换	50

6.3 小结	53
第七章 表达式	55
7.1 操作符	55
7.2 算术操作符和算术表达式	56
7.3 赋值操作符和赋值表达式	61
7.4 关系操作符和关系表达式	62
7.5 逻辑操作符和逻辑表达式	65
7.6 位运算	66
7.7 其它特殊操作符	69
7.8 小结	74
第八章 流程控制	76
8.1 条件语句	76
8.2 循环语句	83
8.3 条件编译	87
8.4 异常处理语句	92
8.5 小结	97

第三部分 面向对象的 C#

第九章 面向对象的程序设计	98
9.1 面向对象的基本概念	98
9.2 对象的模型技术	100
9.3 面向对象的分析	102
9.4 面向对象的设计	104
9.5 小结	107
第十章 类	109
10.1 类的声明	109
10.2 类的成员	110
10.3 构造函数和析构函数	116
10.4 小结	119
第十一章 方法	120
11.1 方法的声明	120
11.2 方法中的参数	121

11.3 静态和非静态的方法	125
11.4 方法的重载	126
11.5 操作符重载	130
11.6 小结	133
第十二章 域和属性	135
12.1 域	135
12.2 属性	139
12.3 小结	142
第十三章 事件和索引指示器	144
13.1 事件	144
13.2 索引指示器	147
13.3 小结	150
第十四章 继承	151
14.1 C#的继承机制	151
14.2 多态性	155
14.3 抽象与密封	159
14.4 继承中关于属性的一些问题	165
14.5 小结	168
 第四部分 深入了解 C#	
第十五章 接口	170
15.1 组件编程技术	170
15.2 接口定义	173
15.3 接口的成员	174
15.4 接口的实现	178
15.5 抽象类与接口	191
15.6 小结	191

第十六章 组织应用程序	194
16.1 基本概念	194
16.2 使用名字空间	196
16.3 使用指示符	199
16.4 程序示例	202
16.5 小结	209
第十七章 文件操作	210
17.1 .Net 框架结构提供的 I/O 方式	210
17.2 文件存储管理	212
17.3 读写文件	217
17.4 异步文件操作	222
17.5 小结	229
第十八章 高级话题	230
18.1 注册表编程	230
18.2 在 C# 代码中调用 C++ 和 VB 编写的组件	235
18.3 版本控制	244
18.4 代码优化	247
18.5 小结	249

第五部分 附录

附录 A 关键字	250
附录 B 错误码	251
附录 C .Net 名字空间成员速查	264

第一部分 C#语言概述

第一章 .NET 编程语言 C#

未来 5 年, 我们的目标就是超越今天各自为营的 Web 站点, 把 Internet 建成一个可以互相交换组件的地方。——比尔·盖茨

在本章中你将了解:

- Microsoft.NET 的概念
- .NET 框架
- C#语言在.NET 框架中的作用及其特性

1.1 Microsoft.NET——一场新的革命

1.1.1 什么是.NET

2000 年 6 月 22 日, 不论对 Microsoft 还是对整个 IT 业界都将成为值得纪念的一天。这一天, 微软公司正式推出了其下一代计算计划——Microsoft.NET(以下简称.NET)。这项计划将使微软现有的软件在 Web 时代不仅适用于传统的 PC, 而且也能够满足目前呈强劲增长势头的新设备, 诸如蜂窝电话以及个人数字助理(Personal Digital Assistant, PDA)等的需要。微软还计划通过创建新的工具来吸引软件开发人员和合作伙伴对 Microsoft.NET 的认同, 并且开发出其他基于 Internet 的服务。

那么, 你是否想知道: 究竟什么是.NET?

请听听微软官员的声音: “……因特网的革命……从微软的角度来讲, 我们就是要建设一个平台来创建并且支持新一代的应用。……我们必须有一套通用系统服务来支持这样的操作。这种观点就说明, 我们还有下一个层次的发展, 也就是说因特网下一步的发展, 它将使因特网的作用远远超越展现一个网站。”

.NET 首先是一个开发平台, 它定义了一种公用语言子集(Common Language Subset, CLS), 这是一种为符合其规范的语言与类库之间提供无缝集成的混合语。.NET 统一了编程类库, 提供了对下一代网络通信标准, 可扩展标记语言(Extensible Markup

Language, XML) 的完全支持, 使应用程序的开发变得更容易、更简单。Microsoft.NET 计划还将实现人机交互方面的革命, 微软将在其软件中添加手写和语音识别的功能, 让人们能够与计算机进行更好的交流, 并在此基础上继续扩展功能, 增加对各种用户终端的支持能力。最为重要的, .NET 将改变因特网的行为方式: 软件将变成服务。与 Microsoft 的其它产品一样, .NET 与 Windows 平台紧密集成, 并且与其它微软产品相比它更进一步: 由于其运行库已经与操作系统融合在了一起, 从广义上把它称为一个运行库也不为过。

简而言之, .NET 是一种面向网络、支持各种用户终端的开发平台环境。微软的宏伟目标是让 Microsoft.NET 彻底改变软件的开发方式、发行方式、使用方式等等, 并且不止是针对微软一家, 而是面向所有开发商与运营商! .NET 的核心内容之一就是要搭建第三代因特网平台, 这个网络平台将解决网站之间的协同合作问题, 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, 网站之间形成自动交流, 协同工作, 提供最全面的服务。

1.1.2 我们为什么需要.NET

某一天, 你出差到外地, 在机场租借手机电话。在向该终端插入自己的 IC 卡后, 自己的地址簿和计划簿被自动下载, 随即它就变成了你个人专用的 PDA。这不是梦境! 这是.NET 为我们描绘的一个未来生活的场景。

人们的需要总是无法满足, 我们不断地问自己: “我们还应该有些什么?” 需求推动着技术的进步。在二十一世纪, Internet 将成为商业活动的主要场所, B2B、B2C 等电子商务的运作方式, 一对一营销的经营概念将网络的服务功能提高到了前所未有的程度。微软公司在此时提出.NET 有其深远的战略考虑:

改革商务模型。微软公司感觉到只靠销售软件包的商务模型没有什么前途, 该公司打算今后将中心转移到可以在网络上使用“服务”型商务。这样, 首要的问题就是解决网络上用来开发并执行“服务”的平台, 这就是 Microsoft.NET。

提高软件开发生产效率, 并且试图使应用程序的发布更为容易 (再也不想因为 DLL 版本不同而烦恼, 希望不用重新启动电脑就能够安装应用软件)。

改进用户界面, 并能支持多种用户终端。用户界面演进的结果包括两方面的内容, 一是完成传统的 PC 界面与基于 XML 的浏览器界面间的过渡, 二是对自然语言和语音识别的支持, 从而使用户与各种终端之间的沟通更加透明, 真正达到网络互连的“3A”: Anywhere、Anytime、Any device。

今天, 许多的人时常问: “除了上网看新闻, 我们究竟还能干什么?” 这是因为今天的互联网与旧式的大型计算机的工作模式还有许多相似之处, 信息被储存在中央服务器内, 而用户的所有操作都要依靠它们。让不同的网址之间相互传递有意义的信息, 或者合作提供更广泛和更深层次的服务, 还是一件十分困难的事。

现代人时常有一种困惑, 感觉到如今生活在技术与机器架构的丛林中, 我们在努力地去适应机器, 适应技术, 而不是机器和技术适应人类。科技以人为本还只是一个美好的愿望。这是因为我们还不能将控制信息的权利交给那些需要信息的人们。 .NET 的出

现,意味着人们可以只用一种简单的界面就可以编写、浏览、编辑和分享信息,而且还可以得到功能强大的信息管理工具。由于使用的所有的文件都以符合网络协议的格式存在,所以所有的商业用户和个人用户都可以方便地查找和使用其中的信息,任何规模的公司都可以使用相同的工具与他们的供应商、商业伙伴和客户高效地沟通和分享信息,这样就创造出一种全新的协同工作模式。

总之,.NET 战略是一场软件革命:

- .NET 对最终用户来说非常重要,因为计算机的功能将会得到大幅度提升,同时计算机操作也会变得非常简单。特别地,用户将完全摆脱人为的硬件束缚:用户可以自由冲浪于因特网的多维时空,自由访问、自由查看、自由使用自己的数据,而不是束缚在便携式电脑的方寸空间——可通过任何桌面系统、任何便携式电脑、任何移动电话或 PDA 进行访问,并可对其进行跨应用程序的集成。

- .NET 对开发人员来说也十分重要,因为它不但会改变开发人员开发应用程序的方式,而且使得开发人员能创建出全新的各种应用程序,大幅提高软件生产率。.NET 将保证完全消除当今计算技术中的所有缺陷。.NET 定能实现确保用户从任何地点、任何设备都可访问其个人数据和应用程序的宏伟蓝图。

- .NET 把雇员、客户和商务应用程序整和成一个协调的、能进行智能交互的整体,而各公司无疑将是这场效率 and 生产力革命的最大受益者。.NET 承诺为人类创造一个消除任何鸿沟的商务世界。

1.1.3 .NET 的核心组件

.NET 的核心组件包括:

- 一组用于创建互联网操作系统的构建块,其中包括 Passport.NET (用于用户认证)以及用于文件存储服务、用户首选项管理、日历管理以及众多的其它任务。

- 构建和管理新一代服务的基本结构和工具,包括 Visual Studio.NET、.NET 企业服务器、.Net Framework 和 Windows.NET。

- 能够启用新型智能互联网设备的.NET 设备软件。

- .NET 用户体验。

1.2 .NET 与 C#

1.2.1 支持多种编程语言的.NET 结构框架

让我们翻开教科书,回顾一下近十年来软件开发的历史。

多年以前,当微软的组件对象模型(Component Object Model, COM)尚未推出时,软件的复用性对于开发人员仅仅是一种美好的憧憬。成千上万的程序员为了处理通信、接口和不同语言间的冲突而通宵达旦地艰辛劳动,但却收效甚微。COM 的出现改变了

这一切。通过将组件改变为通用、集成型的构件，开发人员正逐渐地从过去的繁复编程事务中解脱出来，可以选择自己最得心应手的编程语言进行编程。然而，软件组件与应用程序之间的联合仍然是松散的，不同的编程语言与开发平台限制了部件间的互用性，其结果是产生了日益庞大的应用程序与不断升级的软硬件系统。举个很简单的例子，只用五行 C 语言代码就能编写出的一个简单程序，若使用 COM 来编写，结果会是令人吃惊的：我们需要几百行代码。COM 在带来巨大价值的同时，也大大增加了开发开销。而 .NET Framework 的出现使得一切问题都迎刃而解。实际上，在 .NET Framework 中，所有的编程语言，从相对简单的 JScript 到复杂的 C++ 语言，一律是等同的。

Framework——框架，是开发人员对编程语言命令集的称呼。.Net 框架的意义就在于只用统一的命令集支持任何的编程语言。正如微软 Web 服务中心的成组产品经理 John Montgomery 所说：“只需简单地一用，.NET 框架便可消除各种异类框架之间的差异，将它们合并为一个整体。.NET 的作用不仅仅是将开发人员从必须掌握多种框架的束缚中解脱出来，通过创建跨编程语言的公共 API 集，.NET 框架可提供强大的跨语言继承性、错误处理和调试功能。现在，开发人员可以自由地选择他们喜欢的编程语言。.NET 平台欢迎所有人的垂顾。”.NET 将使编程人员梦想的语言互用性变成近在眼前的现实。想想看，一个在 Visual Basic (VB) 中定义类能够在另一种与它完全不同的语言环境中使用、调试，甚至继承，这是多么令人兴奋的事情！

.NET 框架是 .NET 平台的基础架构。其强大功能来自于公共语言运行时 (Common Language Runtime, CLR 将在第二章中进行详细的解释) 环境和类库。CLR 和类库 (包括: Windows Forms, ADO.NET 和 ASP.NET) 紧密结合在一起，提供了不同系统之间交叉与综合的解决方案和服务。.NET 框架创造了一个完全可操控的、安全的和特性丰富的应用执行环境。这不但使得应用程序的开发与发布更加简单，并且成就了众多种类语言间的无缝集成。

1.2.2 面向 .Net 的全新开发工具——C#

在最近的一段时间里，C 和 C++ 一直是最有生命力的程序设计语言。这两种语言为程序员提供了丰富的功能，高度的灵活性和强大的底层控制能力。而这一切都不得不在效率上作出不同程度的牺牲。如果你使用过包括 C 和 C++ 在内的多种程序设计语言，相信你会深刻体会到它们之间的区别。比如与 Visual Basic 相比，Visual C++ 程序员为实现同样的功能就要花费更长的开发周期。由于 C 和 C++ 即为我们带来了高度的灵活性，又使我们必须要忍受学习的艰苦和开发的长期性，许多 C 和 C++ 程序员一直在寻求一种新的语言，以图在开发能力和效率之间取得更好的平衡。

今天，人们改进、开发出了许多语言以提高软件生产率，但这些或多或少都以牺牲 C 和 C++ 程序员所需要的灵活性为代价。这样的解决方案在程序员身上套上了太多的枷锁，限制了他们能力的发挥。它们不能很好地与原有的系统兼容，更为令人头痛的是，它们并不总是与当前的 Web 应用结合得很好。

理想的解决方案，是将快速的应用开发与对底层平台所有功能的访问紧密结合在一起。程序员们需要一种环境：它与 Web 标准完全同步，并且具备与现存应用间方便地

进行集成的能力。除此之外，程序员们喜欢它允许自己在需要时使用底层代码。

针对该问题，微软的解决方案是一种称之为 C# 的程序语言。C# 是一种现代的面向对象的程序开发语言，它使得程序员能够在新的微软 .NET 平台上快速开发种类丰富的应用程序。.NET 平台提供了大量的工具和服务，能够最大限度地发掘和使用计算及通信能力。

由于其一流的面向对象的设计，从构建组件形式的高层商业对象到构造系统级应用程序，你都会发现，C# 将是最合适的选择。使用 C# 语言设计的组件能够用于 Web 服务，这样通过 Internet，可以被运行于任何操作系统上任何编程语言所调用。

不但如此，C# 还能为 C++ 程序员提供快捷的开发方式，又没有丢掉 C 和 C++ 的基本特征——强大的控制能力。C# 与 C 和 C++ 有着很大程度上的相似性，熟悉 C 和 C++ 的开发人员很快就能精通 C#。

1.3 C# 语言的特点

C# 在带来对应用程序的快速开发能力的同时，并没有牺牲 C 与 C++ 程序员所关心的各种特性。它忠实地继承了 C 和 C++ 的优点。如果你对 C 或 C++ 有所了解，你会发现它是那样的熟悉。即使你是一位新手，C# 也不会给你带来任何其它的麻烦，快速应用程序开发（Rapid Application Development, RAD）的思想与简洁的语法将会使你迅速成为一名熟练的开发人员。

正如前文所述，C# 是专门为 .NET 应用而开发出的语言。这从根本上保证了 C# 与 .NET 框架的完美结合。在 .NET 运行库的支持下，.NET 框架的各种优点在 C# 中表现得淋漓尽致。让我们先来看看 C# 的一些突出的特点，相信在以后的学习过程中，你将深深体会到“#”——“SHARP”的真正含义。

- 简洁的语法
- 精心地面向对象设计
- 与 Web 的紧密结合
- 完整的安全性及错误处理
- 版本处理技术
- 灵活性与兼容性

1.3.1 简洁的语法

请原谅，虽然我们一再强调学习本书不需要任何的编程基础，但在这里还不得不提到 C++。

在缺省的情况下，C# 的代码在 .NET 框架提供的“可操控”环境下运行，不允许直接地内存操作。它所带来的最大特色是没了指针。与此相关的，那些在 C++ 中被疯狂使用的操作符（例如：“::”、“->”和“.”）已经不再出现。C# 只支持一个“.”，对于我们来说，现在需要理解的一切仅仅是名字嵌套而已。

C#用真正的关键字换掉了那些把活动模板库（Active Template Library, ALT）和 COM 搞得乱糟糟的伪关键字,如 OLE_COLOR、BOOL、VARIANT_BOOL、DISPID_XXXXX 等等。每种 C#类型在.NET 类库中都有了新名字。

语法中的冗余是 C++中的常见的问题,比如“const”和“#define”、各种各样的字符类型等等。C#对此进行了简化,只保留了常见的形式,而别的冗余形式从它的语法结构中被清除了出去。

1.3.2 精心地面向对象设计

也许你会说,从 Smalltalk 开始,面向对象的话题就始终缠绕着任何一种现代程序设计语言。的确,C#具有面向对象的语言所应有的一切特性:封装、继承与多态,这并不出奇。然而,通过精心地面向对象设计,从高级商业对象到系统级应用,C#是建造广泛组件的绝对选择。

在 C#的类型系统中,每种类型都可以看作一个对象。C#提供了一个叫做装箱（boxing）与拆箱（unboxing）的机制来完成这种操作,而不给使用者带来麻烦,这在以后的章节中将进行更为详细的介绍。

C#只允许单继承,即一个类不会有多个基类,从而避免了类型定义的混乱。在后面的学习中你很快会发现,C#中没有了全局函数,没有了全局变量,也没有了全局常数。一切的一切,都必须封装在一个类之中。你的代码将具有更好的可读性,并且减少了发生命名冲突的可能。

整个 C#的类模型是建立在.NET 虚拟对象系统（Visual Object System, VOS）的基础之上,其对象模型是.NET 基础架构的一部分,而不再是其本身的组成成分。在下面将会谈到,这样做的另一个好处是兼容性。

借助于从 VB 中得来的丰富的 RAD 经验,C#具备了良好的开发环境。结合自身强大的面向对象功能,C#使得开发人员的生产效率得到极大的提高。对于公司而言,软件开发周期的缩短将能使它们更好地应付网络经济的竞争。在功能与效率的杠杆上人们终于找到了支点。

1.3.3 与 Web 的紧密结合

.NET 中新的应用程序开发模型意味着越来越多的解决方案需要与 Web 标准相统一,例如超文本标记语言（Hypertext Markup Language, HTML）和 XML。由于历史的原因,现存的一些开发工具不能与 Web 紧密地结合。SOAP 的使用使得 C#克服了这一缺陷,大规模深层次的分布式开发从此成为可能。

由于有了 Web 服务框架的帮助,对程序员来说,网络服务看起来就像是 C#的本地对象。程序员们能够利用他们已有的面向对象的知识与技巧开发 Web 服务。仅需要使用简单的 C#语言结构,C#组件将能够方便地为 Web 服务,并允许它们通过 Internet 被运行在任何操作系统上的任何语言所调用。举个例子,XML 已经成为网络中数据结构传送的标准,为了提高效率,C#允许直接将 XML 数据映射成为结构。这样就可以有效地处理各种数据。

1.3.4 完全的安全性与错误处理

语言的安全性与错误处理能力，是衡量一种语言是否优秀的重要依据。任何人都会犯错误，即使是最熟练的程序员也不例外：忘记变量的初始化，对不属于自己管理范围的内存空间进行修改，……。这些错误常常产生难以预见的后果。一旦这样的软件被投入使用，寻找与改正这些简单错误的代价将会是让人无法承受的。C#的先进设计思想可以消除软件开发中的许多常见错误，并提供了包括类型安全在内的完整的安全性能。为了减少开发中的错误，C#会帮助开发者通过更少的代码完成相同的功能，这不但减轻了编程人员的工作量，同时更有效地避免了错误发生。

.NET 运行库提供了代码访问安全特性，它允许管理员和用户根据代码的 ID 来配置安全等级。在缺省情况下，从 Internet 和 Intranet 下载的代码都不允许访问任何本地文件和资源。比方说，一个在网络上的共享目录中运行的程序，如果它要访问本地的一些资源，那么异常将被触发，它将会无情地被异常扔出去，若拷贝到本地硬盘上运行则一切正常。内存管理中的垃圾收集机制减轻了开发人员对内存管理的负担。.NET 平台提供的垃圾收集器（Garbage Collection, GC）将负责资源的释放与对象撤销时的内存清理工作。

变量是类型安全的。C#中不能使用未初始化的变量，对象的成员变量由编译器负责将其置为零，当局部变量未经初始化而被使用时，编译器将做出提醒；C#不支持不安全的指向，不能将整数指向引用类型，例如对象，当进行下行指向时，C#将自动验证指向的有效性；C#中提供了边界检查与溢出检查功能。

1.3.5 版本处理技术

C#提供内置的版本支持来减少开发费用，使用 C#将会使开发人员更加轻易地开发和维护各种商业应用。

升级软件系统中的组件（模块）是一件容易产生错误的工作。在代码修改过程中可能对现存的软件产生影响，很有可能导致程序的崩溃。为了帮助开发人员处理这些问题，C#在语言中内置了版本控制功能。例如：函数重载必须被显式地声明，而不会像在 C++ 或 Java 中经常发生的那样不经意地被进行，这可以防止代码级错误和保留版本化的特性。另一个相关的特性是接口和接口继承的支持。这些特性可以保证复杂的软件可以被方便地开发和升级。

1.3.6 灵活性和兼容性

在简化语法的同时，C#并没有失去灵活性。尽管它不是一种无限制的语言，比如：它不能用来开发硬件驱动程序，在默认的状态下没有指针等等，但是，在学习过程中你将发现。它仍然是那样的灵巧。

如果需要，C#允许你将某些类或者类的某些方法声明为非安全的。这样一来，你将能够使用指针、结构和静态数组，并且调用这些非安全的代码不会带来任何其它的问题。此外，它还提供了一个另外的东西（这样的称呼多少有些不敬）来模拟指针的功能——

delegates, 代表。再举一个例子: C#不支持类的多继承, 但是通过对接口的继承, 你将获得这一功能。

下面谈谈兼容性。

正是由于其灵活性, C#允许与 C 风格的需要传递指针型参数的 API 进行交互操作, DLL 的任何入口点都可以在程序中进行访问。C#遵守.NET 公用语言规范 (Common Language Specification, CLS), 从而保证了 C#组件与其它语言组件间的互操作性。元数据 (Metadata) 概念的引入既保证了兼容性, 又实现了类型安全。

1.4 小 结

Microsoft.NET 计划将彻底改变我们对因特网的认识, 从而在这样一个网络时代彻底改变我们的生活。软件是一种服务, 技术是我们的仆人, 时间与地点将不再是我们面前的障碍。建立在 CLR 与类库基础上的.NET 框架是.NET 平台的核心组件之一, 这为软件的可移植性与可扩展能力奠定了坚实的基础, 并为 C#语言的应用创造了良好的环境。

C#是.NET 平台的通用开发工具, 它能够建造所有的.NET 应用。其固有的特性保证了它是一种高效、安全、灵活的现代程序设计语言。从最普通的应用到大规模的商业开发, C#与.NET 平台的结合将为你提供完整的解决方案。

在本章中, 我们提出了与.NET 以及与 C#语言相关的一些概念, 例如 CLR、VOS 和 GC, 也许你是初次接触它们, 但不用担心, 在以后的各章中我们将详细地介绍这些相关的概念与知识, 相信通过学习, 你将能够迅速掌握它们, 并熟练地运用它们提供的各种特性。

复习题

- (1) 什么是.NET?
- (2) 简要说明.NET 战略的意义。
- (3) .NET 的核心组件包括哪些?
- (4) C#与其它语言相比有哪些突出特点?

第二章 运行环境——全面了解.NET

C#运行在.NET平台之上,其各种特性与.NET密切联系。它没有自己的运行库,许多强大的功能均来自.NET平台的支持。因此,要想真正掌握C#首先必须了解.NET。本章将向你介绍C#的运行环境,重点放在.NET公用语言运行时环境与公用语言规范上,最后介绍了.NET的开发工具。

2.1 .NET 结构

.NET包括四个组成部分:

- VOS 类型系统
- 元数据
- 公用语言规范
- 虚拟执行系统

下面分别对它们进行简要介绍。

2.1.1 虚拟对象系统

.NET跨语言集成的特性来自于虚拟对象系统(VOS)的支持。

在不同语言间进行代码复用和应用集成中所遇到的最大问题,是不同语言类型系统间的相容性问题。可以想象,不同的语言虽然语法结构大体相同,但数据类型与语言环境本身的各种特点联系紧密,很难想象一种解释性的语言所拥有的数据类型会与一种编译语言相同,而即使相同的数据类型在不同的语言环境中表示的意义也存在差别。例如,同样是整数类型,在MSSQL中的长度是32位,而在VB中却是16位,至于日期时间与字符串类型在这方面的区别就更加明显了。

VOS的建立就是为了改变这种状况。它既支持过程性语言也支持面向对象的语言,同时提供了一个类型丰富的系统来容纳它所支持的各种语言的特性。它在最大程度上屏蔽了不同语言类型系统间的转换,使程序员能够随心所欲地选择自己喜欢的语言(当然,这种语言必须支持.NET应用)从事开发,保证了不同语言间的集成。

对于过程性语言,它描述了值的类型并指定了类型的所有值必须遵守的规则;在面向对象的语言方面,它统一了不同编程语言的对象模型。每一个对象在VOS中都被唯一标识以与其它对象相区别。

2.1.2 元数据

元数据是对 VOS 中类型描述代码的一种称呼。在编译程序将源代码转换成为中间代码时，它将自动生成，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，这在一定程度上解决了版本问题：程序使用的类型描述与其自身绑定在一起。

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，将中间语言代码翻译成为本地代码，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 始终根据元数据建立并管理对应特定应用程序的类型，从而保证了类型安全性。

此外，元数据在解决方法的调用，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。

2.1.3 公用语言规范

公用语言规范（Common Language Specification, CLS），是 CLR 定义的语言特性集合，主要用来解决互操作问题。如果一个类库遵守 CLS，那么同样遵守 CLS 规范的其它编程语言将能够使用它的外部可见项。详细的内容见本章第二节。

2.1.4 虚拟执行系统

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

- 装入中间代码。
- 使用 JIT 将中间代码转换为本地码。
- 装入元数据。
- 代码管理服务——包括垃圾收集器和异常处理。
- 定制与调试服务。
- 线程和环境管理。

2.2 公用语言运行时环境与公用语言规范

了解了.NET 的结构之后，我们该看看.NET 利用其结构为我们创造的运行环境——公用语言运行时环境。它是 C#及其它支持.NET 平台的开发工具的运行基础。具体来说，它为我们的应用提供了以下益处：

- 跨语言集成的能力。
- 跨语言异常处理。
- 内存管理自动化。

- 强化的安全措施。
- 版本处理技术。
- 组件交互的简化模型。

2.2.1 理解 CLR

.NET 提供了一个运行时环境,叫做公用语言运行时,它管理着代码的执行,并使得开发过程变得更加简单。这是一种可操控的执行环境,其功能通过编译器与其它工具共同展现,你的代码将受益于这一环境。依靠一种以运行时为目标的(指完全支持运行时环境的)编译器所开发的代码叫做可操控代码。它得益于可操控环境的各种特性:跨语言集成、跨语言异常处理、增强的安全性、版本处理与开发支持、简单的组件交互模型以及调试服务。为了使运行时环境能够向可操控代码提供服务,语言编译器需要产生一种元数据,它将提供在你使用语言中的类型、成员、引用的信息。元数据与代码一起存储,每个可加载的 CLR 映像均包含了元数据。运行时环境使用元数据定位并载入类,在内存中展开对象实例,解决方法调用,产生本地代码,强制执行安全性,并建立运行时环境的边界。

运行时环境自动处理对象的展开与引用,当它们不再使用时负责它们的释放。被运行时环境进行这样的生命期管理的对象被称为可操控代码。自动内存管理消除了内存溢出,同时也解决了其它一些常见的语法错误。如果你的代码是可操控的,你仍然可以在需要的时候使用非可控代码,或者在你的.NET 应用中同时使用可控与非可控代码。由于语言编译器支持他们自己的类型,比如一些原始类型,你可能并不总是知道(也不必知道)你的数据是否是可控的。

CLR 使设计跨语言的组件与应用变得更加容易。以不同语言设计的对象能够彼此间进行通信,并且它们的行为能够紧密地综合与协调。举个例子,你定义了一个类,然后可以在另一种不同的语言中从该类中派生了一个类或者调用它其中的一个方法。你也可以向另一种语言中类的方法传递该类的一个实例。这种跨语言的集成之所以可能,因为以运行时间为目标的语言编译器与工具使用一种运行时间所定义的公用类型系统,他们遵守运行时的规则(公用语言规范)来定义新的类型,生成、使用、保持并绑定类型。

作为元数据的一部分,所有可控组件携带了关于它们所依赖的组件与资源的信息。运行时环境使用这些信息来保证你的组件或应用具有需要的所有东西的特定版本,其结果是你的代码将不会因为版本冲突而崩溃。注册信息与状态数据不再保存在难以建立与维护的注册表中,你所定义的类型及附属信息作为元数据被保存,这使得复制与移动组件的复杂程度得到降低。

编译工具用他们自己的方式向开发人员展现 CLR 的功能。这意味着运行时间的一些特性可能在不同的语言中的表现形式将会有所不同。你怎样体验运行时的特性将取决于你所使用的语言,比如说,如果你是一位 VB 开发人员,你可能注意到在运行时环境的帮助下,VB 语言比以前具有更多的面向对象的特性。