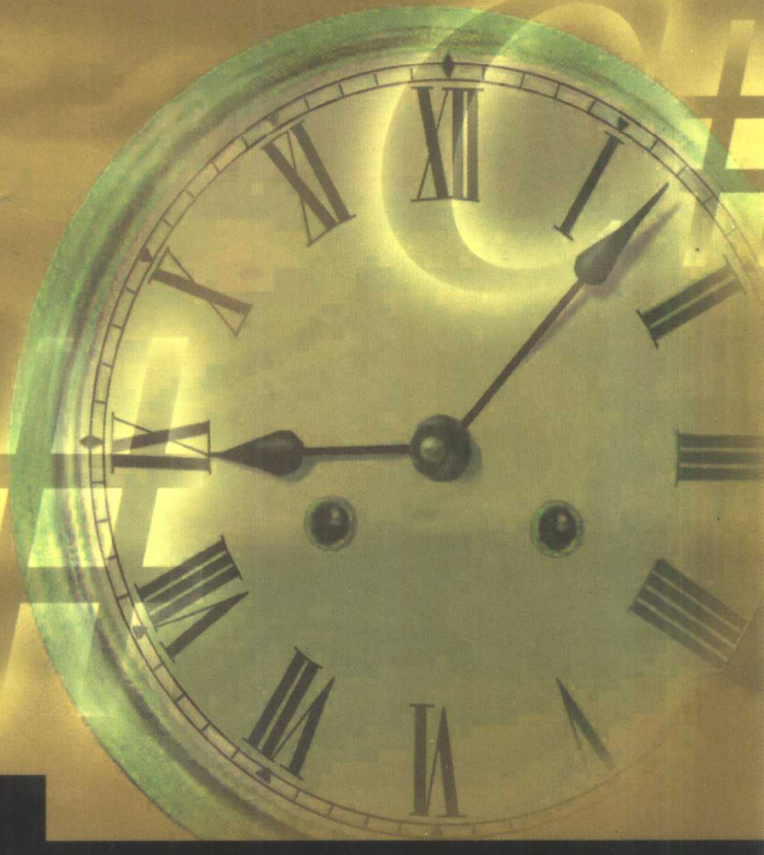
C#程序设计教程

● 余安萍　俞俊平　孙华志　编著



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http://www.phei.com.cn

C# 程序设计教程

余安萍　俞俊平　孙华志　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书对新一代编程语言 C# 进行了全面的介绍。C# 是基于微软的下一代开发平台 .Net 的全新的面向对象的程序设计语言。C# 不仅支持分布式应用的开发，还可以开发任何古典风格的 Windows 程序，包括控制程序、图形程序、服务程序、普通组件和 Web 页面等(硬件驱动程序除外)。C# 语言不仅保持了 C++ 语言的熟悉的语法，同时还集成了 Visual Basic 语言的快速应用开发功能以及类似于 Java 的与平台无关和即时编译特性，为程序的开发提供了更高的稳定性、可靠性和安全性。这就意味着学会它以后在应用开发中可以花费更少的时间和更低的培训费用，使源代码明了简洁，这一切都将使得开发者的开发过程变得轻松愉快。

作者根据多年的编程实践经验，对全书进行了合理、严密的组织。本书内容简明扼要，思路清晰，示例丰富，实用性强。通过本书的学习，读者可以在最短的时间内掌握 C# 语言。

本书适用于广大程序设计爱好者、软件开发人员。

图书在版编目(CIP)数据

C# 程序设计教程/余安萍等编著 . - 北京:电子工业出版社，2002.1
ISBN 7-5053-6920-2

Ⅰ.C… Ⅱ.余… Ⅲ.C 语言 - 程序设计 - 教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2001)第 058727 号

书　　名：C# 程序设计教程
编 著 者：余安萍　俞俊平　孙华志
责任编辑：黄志瑜
排版制作：电子工业出版社计算机排版室
印 刷 者：北京天竺颖华印刷厂
装 订 者：三河市金马印装有限公司
出版发行：电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787 × 1 092　1/16　印张：18　字数：461 千字
版　　次：2002 年 1 月第 1 版　2002 年 1 月第 1 次印刷
书　　号：ISBN 7-5053-6920-2/TP·3939
印　　数：5 000 册　定价：24.00 元

凡购买电子工业出版社的图书，如有缺页、倒页、脱页、所附磁盘或光盘有问题者，请向购买书店调换；
若书店售缺，请与本社发行部联系调换。电话　68279077

前　言

在软件开发界，因为 C++ 语言的强大功能，多年来它一直占据霸主的位置，几乎所有的操作系统和绝大多数的商品软件都用 C++ 作为主要开发语言。但是，在商品经济高速发展的今天，时间就是一切！开发人员已经没有时间，也没有耐心一遍遍重复“编写代码—编译—调试”这样一个令人厌烦的、枯燥的循环，开发人员需要一个更高效、更安全、更稳定的软件开发工具。

作为软件业的老大，微软公司不失时机地推出了其下一代网络计划——Microsoft .Net（以下简称 .Net 计划）。这项计划将彻底改变软件的开发方式、发行方式和使用方式等。该计划主要基于 XML 语言数据共享标准，其目的在于使微软现有的软件可以广泛地通过网络进入传统的 PC 电脑和逐渐流行的设备，如移动电话等。同时，微软也宣布了它的 .Net Framework（.Net 结构，也称为 .Net 平台），微软声称该结构将为所有编程语言提供一个通用平台。也就是说，该结构允许软件开发商可以使用多种编程语言联合开发应用程序。

接着，在 2000 年的 7 月份，微软又发布了它的针对 .Net 结构的下一代软件开发工具：Visual Studio .Net 7.0 测试版。该工具包包括对 Visual Basic、Visual C++ 和 Visual FoxPro 工具的升级以及一种新的编程工具——C#（发音为 C Sharp）语言的第一版。作为 .Net 结构的核心开发语言，C# 语言几乎集中了现今已有软件的所有优点：面向对象，类型安全，组件技术，自动内存管理，跨平台异常处理，版本控制和代码安全管理等。最重要的特点是，相对于 C++ 语言，C# 将使开发人员能够更快、更容易地创建应用，尤其是分布式应用。

本书采用循序渐进的方法，以简单实用、易学习和易掌握为宗旨，以清晰的思路、简练的文笔、丰富的示例和详尽的文档全面介绍了 C# 语言的基本知识以及如何利用 C# 语言进行开发应用。本书分为 5 部分，第一部分为 C# 语言的简介，包括第 1 章到第 3 章，主要介绍了 .Net 平台和 C# 语言的技术基础，以及 Visual Studio .Net 7.0 开发环境，并创建了第一个 C# 程序。第二部分是 C# 语言的基本概念，包括第 4 章到第 8 章，主要介绍了利用 C# 语言编程时需要的一些基本概念，如 C# 语言的类型、变量、表达式和语句以及 C# 语言的类和结构对象的声明、创建和使用。第三部分为 C# 语言的编程进阶，包括第 9 章到第 13 章，主要介绍了如何利用 C# 语言提供的新特性（如代理、属性以及名字空间）进行应用开发；另外，还介绍了在 C# 程序中如何处理异常以及调试 C# 应用的方法和步骤。第四部分是 C# 组件开发，包括第 14 章到第 15 章，主要讲述了 C# 对组件编程的支持，并通过实例比较了在 .Net 框架中对 COM 组件和 COM+ 组件的调用的不同区别。第五部分是 C# 高级应用，包括第 16 章到第 20 章，这一部分主要通过具体的例子来讲述利用 C# 语言如何进行更高级的应用开发，包括数据库编程、文件操作、线程应用、Web 应用以及其他一些非常有用的应用。对于本书中所有例子，作者均已在自己的机器上调试通过。只要读者掌握了本书的内容，就将能熟练地利用 C# 语言开发自己的应用。

本书由飞雪漫工作室的余安萍、俞俊平、孙华志主笔，另外，余显明、王正梅、俞俊军、俞晶樱、余小多等在本书的编写过程中也给予了大力帮助。特别感谢电子工业出版社为本书的内容编排提出了宝贵的意见。尽管在编写过程中，我们进行了认真细致的审改工作，但由于时间仓促，加上作者水平有限，书中难免存在不足之处，恳请读者批评指正。

作　者

目　录

第 1 章　.Net 简 介

在过去的十几年里，C/C++和 Visual C++以及 Visual Basic 是软件开发商开发商业软件常用的几种编程工具。这几种语言都有自己独有的特色，C/C++和 Visual C++语言功能强大，可用于开发一些功能强大的商业软件；而 Visual Basic 简单方便，开发周期短，容易上手，但对底层开发支持不够。由于这些语言都有各自的优缺点，所以一直以来，软件开发者都希望能有一种语言，既保留它们的优点，又摒弃了它们的不足，为我们提供更强的功能和更高的效率。于是微软的新一代开发工具 C#（发音为：C sharp）应运而生了。C#是一种现代、面向对象的语言，它结合了 C/C++和 Visual C++的强大功能和 Visual Basic 的易用性。另外，从 C #的语言规范可以看出，无论在语法方面还是在丰富的 Web 开发支持及自动化的内存管理上，C #都和 Java 非常相似。因此，如果读者曾经用过 C++或者 Java，再来学习 C #应该是相当轻松的。

本章是关于 C#语言理论方面的介绍，希望读者在读完本章以后，对 C#这个新语言能有一个初步的认识。

1.1　新一代编程语言——C#

2000 年 7 月，在佛罗里达州奥兰多市举行的软件开发商大会上，微软发布了它的第一批网络（.Net）计划工具。微软声称该 .Net 计划主要基于 XML 语言数据共享标准，其目的是使微软现有的软件可通过网络进入传统的 PC 电脑以及逐渐流行的设备，如移动电话和个人数字助理（PDA）。

微软的.Net 计划包括两大部分：.Net 平台和.Net 开发工具。前者为后者提供工具和服务平台支持。

在这次大会上，微软发布的 Microsoft.Net 平台提供了一种网络结构，该结构允许软件开发商使用多种编程语言混合编程。微软声称，此结构为当前流行的多种语言提供了一个通用平台。除了微软自身的语言外，17 种其他语言，包括流行的网络语言 Perl，也将能运行在.Net 框架上。这归功于.Net 框架中的运行环境引擎——微软的下一代 Windows 服务运行引擎（Next Generation Windows Services Runtime，NGWS Runtime）。

另外，微软发布了它的下一代开发工具：Visual Studio.Net 的测试版。该工具包括 Visual Basic，Visual C++和 Visual FoxPro 工具的升级版本，以及一种新的开发工具——C #的第一版。其中，C#语言能够快速开发基于新一代平台 Microsoft.Net 的应用软件。

C #语言是从 C/C++演变而来的，但它具有既先进又易学的特点，并完全面向对象和类型安全。C #最重要的特点之一：它是一种现代的编程语言。它简化了 C++在类、名字空间、方法重载和异常处理等方面的操作；它摒弃了 C++的复杂性，更易使用，更少出错。有人声称，在企业计算领域内，C #将会成为用于编写 NGWS 应用程序的主要语言。下面让我们看看 C#和 C++的不同之处。

（1）C++的指针不再是编程武器的一部分。缺省情况下，C #程序运行在受管理的代

码中，不允许如直接内存存取等不安全的操作。当然与指针相关的操作符，如::、.和->，一般都不再使用。但有时为了功能上的需要，会编制一些不安全代码，此时还会用到指针类型。

（2）不再支持模板、宏和多重继承。

（3）去掉 C++中一些多余无用的东西。这些无用的内容包括常数预定义，不同字符类型。

作为一种新语言，C#为我们提供了以下一些重要的新特性。

1. 简单

第一，C #之所以比 C++简单，一方面是因为它减少了 C++的一些特性，上面已经介绍过了；另一方面，前面也已经提到，C #不再支持指针，此时内存管理的工作不再是程序员的任务，而是.Net 平台的运行库的一部分。后者提供了垃圾收集器（Garbage Collector）用于自动管理程序的内存。可以认为，以上所述是 C #吸引人的一大特色。作者从多年的编程历程中体会到，内存管理是一件令程序员最为头痛的事情，稍不小心，程序就会因为内存的一点小问题而崩溃，并经常需要花大量的时间进行跟踪调试，以便定位导致内存泄漏的代码。现在，.Net 提供的垃圾收集器（GC）将为我们节省大量的时间和精力，提高产品的开发效率。

第二，C #使用统一类型系统，这种系统允许把各种类型作为一个对象查看。利用 C #提供的封装（Boxing）和取消封装（Unboxing）机制，可以把简单类型当做对象类型来处理，具体操作过程在后面的章节中会详细介绍。这里，我们应记住一点：使用 C #，再也不需要记住基于不同处理器的隐含类型以及各种类型的变化范围了。

第三，C++是一种功能强大的编程语言，这是众所周知的事实。但它在实现某些功能方面还是比较费劲的，在这种情况下使用 VB 就显得轻松容易。C #恰好结合了这两种语言的优点，即以前在 C++中很费力才能实现的功能，现在在 C #中只是一部分基本的功能而已。例如，对于 COM 来讲，用 C#来开发相同功能的组件会比 C++容易几个数量级。

第四，为了适应企业级软件开发的需求，C #新增了一种数据类型：金融数据类型。这是一种新的十进制数据类型，专用于金融业的计算。

2. 面向对象

与目前常用的高级编程语言一样，C #依然支持面向对象的功能，如封装、继承和多态性。整个 C #类模式构建在 NGWS 运行库的虚拟对象系统（Virtual Object System，VOS）的上层（VOS 将在第 1.3 节中介绍），对象模式只是基础的一部分，不再是编程语言的一部分。

从现在开始，读者必须记住一件事：在 C #中不再存在全局函数、全局变量或者全局常量。C#中所有内容都封装在类中，这使 C#代码更加易读，进一步避免命名冲突。

缺省情况下，C #类中定义的方法都是非虚拟的。也就是说，在进行 C #类继承操作时，基类的方法不能被重载。如果基类某方法需要重载，那么此方法必须有显式的重载标志（virtual）。这种行为不但缩减了虚拟函数表的大小，而且还能确保正确的版本控制。对于类的继承，C #与 C++一个最大的不同之处在于 C #不支持多重继承。如果必须多重继承，可以利用接口来实现。

3. 类型安全

指针在C++中扮演着重要的角色，它为我们带来了许多强大的功能，但随之而来的也有许多不安全隐患。例如，我们可以自由地将指针强制转换为任何类型，比如将 int * 转换为double *类型，就很可能会造成变量当前数值发生变化的错误。这种不安全性绝不是企业级编程语言应该包含的内容。

用 C #编制的代码都是受管理代码（Managed Code）。关于该术语的解释将在下面的通用语言运行时部分中介绍。这里，读者只须记住，利用 Managed Code 编制程序时，开发人员不需要负责内存的管理。这将使我们从最头痛的工作中解放出来。这全受益于.Net 平台提供的垃圾收集器，此新功能在前面已经提到过。但同时也要求 C #必须遵守一定的变量规则，如不能使用没有初始化的变量和对象的成员变量，并由编译器负责清零；局部变量则由程序内部负责清零。遵守这些规则之后的好处就是可以避免由于使用了没有初始化的变量，导致无法预料结果的错误。

其次，C #取消了不安全的类型转换操作。利用 C #编程时不能将一个整型强制转换成一个引用类型（如对象），但支持从基类向派生类的转换。

另外，C #支持边界检查，再也不会出现 C++中可能会发生的事情：当定义了拥有 n 个元素的一个数组，却越界访问了第 $n+1$ 个元素时，会造成内存访问错误。

4. 版本控制

相信读者在过去的 PC 生涯中一定遇到过众所周知的“DLL 灾难”，就是多个软件都需要相同的 DLL，但它们又具有不同的版本。特别是在安装软件时，经常会有提示：XXX.dll 已经存在，而且比将要安装的要新，是否覆盖？如果你选择了覆盖，那么也许你会痛苦地发现，以前的某个程序已经不能运行了。但在一般情况下，旧版本的应用程序可以很好地在新版本 DLL 上工作。

对于版本控制问题对现代软件开发商来说，是令人头痛也是最为必要的事情。因为现在的软件开发不再是单个个人就能完成的，经常必须由多人组成的一个团队集体合作实现。在团队开发过程中有许多共用的资源，一个模块可能由多人开发，那么正确地实施版本控制技术是绝对需要的。

对新一代编程语言，NGWS 运行库将对应用程序提供版本支持。相对于目前已有的高级编程语言而言，C #最好地支持了版本控制。虽然 C #不能确保正确的版本控制，但它可以对程序员担保版本控制是可能的。正是这种版本控制支持特性，当程序员开发的某类应用升级时，它将保持与已有客户应用程序的二进制级兼容。

5. 支持 XML 文档

C #语言为开发者提供了一种新机制：利用 XML 可以将程序代码文档化。在 Visual Studio.Net 7.0 提供的所有开发工具中只有 C #语言具有这种特征。在源代码文件中，以“///” 字符串开头的行，并且该行位于类、代理或接口这些用户定义的类型声明之前，或名字空间声明之前，那么该行的内容可以当做注释来处理，并放置到文档文件中。为了获得这种 XML 文档，编译源代码时需要加上/doc:file 编译选项，其中 file 就是目标 XML 的文件名。

6. 兼容

利用.Net 平台的通用语言规范（Common Language Specification，CLS），C #语言可以访问其他多种语言编制的 APIs。CLS 是一种标准，遵循这种标准的语言之间可以进行交互操作。

另外，.Net 平台提供对 COM 透明的访问，利用 C#可以方便地访问已有的旧版本的 COM 对象。

C #与 C++的兼容性还表现在：C #允许与 C 风格 APIs 交互操作，我们可以在 C #程序中访问 DLL 内以 C 风格定义的入口。

7. 灵活

上面提到 C #是类型安全的，但它又可以调用 C 风格的 APIs，这就难免会使用需要传递指针的 API。也就是说，C #在访问 Win32 代码时有可能导致对非安全类型指针的使用。为了实现这种功能，C #通过使用非安全代码使用它们。具体地说，C #允许程序员声明一些类或者仅声明类的方法是非安全类型的，这样的声明允许使用指针、结构和静态分配数组。此时，安全代码和非安全代码都在同一个管理空间里运行。这样，当从安全代码里调用非安全代码时并不需要进行装配过程。

也许读者此时会想到一个问题：非安全代码中变量的内存是如何处理的？在安全代码中，一旦检测到某对象不再被使用时，垃圾收集器立即释放此对象占用的内存。但是，垃圾收集器不能定位非安全代码中对象的内存，所以解决的方法就是 NGWS 运行环境引擎将非安全代码中的变量固定在垃圾收集器管理的内存内。

表 1.1 列出了 C#与 C++之间的区别。

表 1.1　C#与 C++的不同点

特　征	不同点描述
类的继承	一个类只能继承于一个基类
接口	一个类或接口可以实现多个接口
数组	C#声明数组时在数组类型之后紧接着的是“[]”标记
bool 类型	C#不支持 bool 类型与其他类型之间的转换
结构类型	C#中类和结构在语义上是不同的。结构是值类型，而类是引用类型
switch 语句	C#不支持多个 case 标号共用一个处理段，即 C#中每个 case 语句必须对应一个 break 语句
代理类型	C#的代理与 C++的指针很相似，但是代理是类型安全和可靠的
new 修饰符	C#利用 new 操作符显式隐藏一个继承成员
重载	在 C#中重载时必须指定 override 关键字
预处理指示器	C#中不用头文件，预处理指示器用于条件编辑
异常处理	C#中的异常处理利用 try-catch-finally 语句块
操作符	C#支持的新操作符：is 和 typeof
Main 方法	C#与 C++的 main 方法不同
方法参数	C#支持 ref 和 out 参数类型，它们代替 C++中的指针参数
非安全代码	C#允许使用指针，但仅能在非代码中使用
字符串类型	C#的字符串与 C++字符串不同

1.2 Microsoft.Net

C#程序在目前常用的 Windows 平台上还不能运行，因为它需要特定的运行库作为基础。目前最终用户必须安装整个.Net 的 SDK 包才能运行 C#程序。Microsoft.Net 是 Microsoft 的新一代 Internet 全面解决方案，开发者利用.Net 平台、产品和服务，可以轻松地建立真正分布式的、协作的新一代 Web 服务，让用户在任何设备、任何时间和任何地点，轻松地获取信息。.Net 框架是 Microsoft 继 DNA 之后最新推出的新一代 Internet 软件开发模型。如图 1.1 所示显示了 Microsoft.Net 框架的组成。

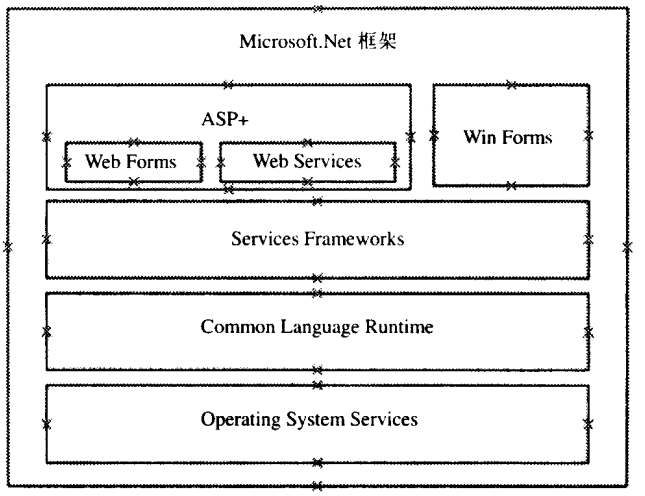


图 1.1 Microsoft .Net 框架组成

从图中可以看出，Microsoft.Net 平台的核心是通用语言运行环境引擎（Common Language Runtime，CLR）和服务框架（Services Frameworks），它们都建立在操作系统层之上。其中，通用语言运行环境引擎就是前面提到的下一代 Windows 服务运行环境引擎（NGWS Runtime），它的功能是管理代码的运行。代码可以采用任何现代编程语言编写，并以一种中间语言（Intermediate Language，IL）代码的形式出现。另外，运行环境引擎还提供了许多服务功能以简化代码的开发和应用配置，同时也改善了应用程序的可靠性。.Net 平台还包括一组类库，开发人员可以在任何一种编程语言环境中调用它们。最上一层是各种应用程序开发模型，并特别针对开发 Web 站点和 Web 服务提供了更高一级的组件和服务。

1.2.1 ASP.Net

Microsoft.Net 框架下的 Web 应用程序模型称做 ASP.Net（在.Net 出现以前，称为 ASP+）。ASP.Net 由 ASP 发展而来，但在许多方面，ASP.Net 与 ASP 有着本质的不同。ASP.Net 完全基于模块与组件，具有更好的可扩展性与可定制性，尤其是数据处理方面引入了许多激动人心的新技术。正是这些具有革新意义的新特性，让 ASP.Net 远远超越了 ASP。

ASP.Net 充分利用了通用语言运行环境和服务框架，为 Web 应用提供可靠的、坚固的和灵活的运行环境。另外，系统提供的服务使应用的开发、配置也变得很简单，有效地缩短了 Web 应用程序的开发周期。ASP.Net 的核心概念是 HTTP Runtime、模块管道和请求处理器。

其中，HTTP Runtime 是基于底层结构建立的高性能 HTTP 处理运行环境；模块管道和请求处理器则增强了系统的灵活性和可扩展性。

从图 1.1 中可以看到，在 ASP.Net 编程模型之上，还有以下两种更高级的编程模型。

（1）ASP.Net Web Forms

Web Forms 的开发风格非常类似于 VB 窗体的快速应用开发。Web Forms 支持传统的 ASP 语法，但它也提供了一个结构化的方法，可以将应用程序代码和用户界面内容分离开。代码与界面的分离使得 ASP.Net 页面可以动态编译成受管理的类，从而使性能有很大的提高。

另外，新引入的 Web Forms 控件提供了一种新机制，即可以将通用用户界面内容打包。这样，我们就可以使用类似 VB 的工具，以所见即所得的方式完成 Web 页面布局的开发。使用控件的一个重要优点就是允许程序能够自适应客户端的配置，这样，相同的页面就可以适应多种不同的客户端平台。

（2）ASP.Net Web Services

ASP.Net Web Services 编程模型的主要优点是简化了 Web Services 的开发。同时，其编程模型也非常类似于人们已熟悉的 ASP 或 VB 开发。

在.Net 框架中，Web Service 是由应用程序完成的服务，通过 Internet 标准可以和其他 Web Service 集成。Web Service 是一个 URL 服务资源，客户端可以通过编程方式请求得到它返回的信息。Web Services 的一个重要特点就是：客户端不需要知道它所请求的服务是怎样实现的。这一点与传统的分布式组件对象模型（DCOM 或 CORBA）完全不同。

.Net 系统之间的通信采用的是目前通行的 Web 协议和数据格式，例如 HTTP 和 XML。任何支持 Web 标准的系统都能支持 Web Services。XML 协议是定义一套可扩展的、标准化的语言的最好选择，它可以表示命令和类型数据定义。SOAP（简单对象访问协议）则是一套用 XML 表示数据和命令的国际标准。因此，在.Net 框架中，利用 SOAP 定义通信消息格式应该是很好的选择。

Web Services 定义了明确的接口，在.Net 框架中称为约定（contracts）。它描述了 Web Services 提供的服务，客户端应用程序根据约定就知道 Web Services 是否包含所需要的服务以及调用方法。开发人员可以通过组合调用远程服务、本地服务和自己编写代码来实现一个 Web 应用。

利用 ASP.Net 建立 Web Services 应用变得非常简单，通过编写一个扩展名为.ASMX 的文件，并且将其配置成 Web 应用的一部分即可。

1.2.2 WinForms

虽然.Net 框架的主要目标是开发 Web 服务和 Web 应用，但它也可以用来开发传统的 Windows 应用。同时，这些应用也可以使用 Web Services。

WinForms 是.Net 类库中的类集，其中封装了 Win32 类中的 Windows，Brushes 和 Pens 类等等。任何使用了.Net 运行库的语言（包括 C #）都可以建立这些类的实例并进行控制。所以，在编写 Windows 客户端应用程序时利用 WinForms 类库，可以调用 Windows 丰富的界面功能，包括现存的 ActiveX 控件和一些 Windows 2000 的新功能。不管选择传统的 Windows 方式还是新的 Web 方式，Win Forms 编程模型和设计都是非常直观的，与目前流行的 Windows 窗体方式很相似。

WinForms 同样也利用了.Net 框架的运行环境引擎，这样可以减少 Windows 客户端应用

程序的开发工作量。同时，.Net 框架的安全模型保证了在客户端机器上能够安全地执行应用程序和组件。

1.2.3 通用语言运行环境

.Net 框架的核心是通用语言运行环境引擎（CLR），也就是 NGWS Runtime。对 Runtime 这个概念读者一定不会陌生。我们常用的 C Runtime 库、标准模板库（STL）、Microsoft 基础类库（MFC）、活动模板库（ATL）和 VB 运行库等，它们的目的都是为应用程序提供通用服务，以节省编程时间和提高程序的可靠性。CLR 的功能也是如此，它负责管理代码的执行并提供服务以使开发过程更简单化。

如果某种语言的编译器是以运行时为目标的，那么利用该语言开发生成的代码在.Net 中称为受管理代码（Managed Code）。这样的代码具有平台无关，交叉语言集成，跨平台异常处理，自动内存管理，增强的安全性，版本控制及组件互操作简单化等特性。简单地说，术语“Managed Code”描述的是那些要求通用语言运行时支持的代码；而“Unmanaged Code”描述的是那些并不需要通用语言运行时引擎支持的代码。在 Visual Studio.Net 中，只有 VC++ 7.0 可以产生 Unmanaged Code，C#以及 Visual Basic 7.0 只能产生 Managed Code。当然，利用.Net 提供的 Managed Extension for C++，VC++ 7.0 也可以开发 Managed Code。正如以前我们使用 C/C++语言开发应用那样，Unmanaged Code 是针对特定 CPU 平台编译的，并且当它激活时，代码就可以简单地被执行。而在受管理环境下，代码的编译是分两步完成的：首先，编译器将源代码翻译成中间语言（MSIL）；然后，在运行时再将 MSIL 编译为本地的 CPU 指令。其实，受管理代码意味着在执行代码和运行时之间定义了一个接口，例如创建对象、方法调用等任务都可以交给 CLR 完成；同时，后者还向执行代码提供一些附加的服务功能。

下面详细介绍一下.Net 的通用语言运行时（CLR）的特征，以及受管理代码的具体执行过程。

1. 平台无关

如图 1.1 所示，最底层是操作系统层，但并没有指定它是 Windows 系统。因此，它可以是任何一种操作系统，即只要运行库被移植到其他系统上，就可以在该系统上运行.Net 应用。也就是说，一个受管理的.Net 应用（完全由受管理代码组成），只要编译一次，就能够在任何支持.Net 的平台上执行。

2. 交叉语言集成

过去，我们可以利用不同语言实现对 COM 的互操作；现在，.Net 实现了不同语言的相互集成。正如以前从很多种交叉语言功能所看到的，NGWS Runtime 主要是一种关于高度集成的交叉多异编程语言（tight integration across multiple different programming languages）。简单地举个例子，读者可以利用 VB 声明一个基类对象，在 C#代码中可直接创建此基类的派生类。.Net 之所以具有如此强大的功能，全得益于 CLR 定义并提供了一个对所有.Net 语言都通用的通用类型系统。

3. 通用类型系统

实际上，.Net 框架的语言集成是在框架的通用类型系统的基础上实现的。.Net 框架的通