YINGYONGCHENGXU

C#应用程序开发

标准教程

Microsoft®
Visual Studio.net
C

BIAOZHUNJIAOCHENG

主编 秦斌 曾斌

上海科学普及出版社

图书在版编目（CIP）数据

C#应用程序开发标准教程 / 秦斌，曾斌主编. —上海：上海科学普及出版社，2004.3
ISBN 7-5427-2643-9

Ⅰ.C… Ⅱ.①秦… ②曾… Ⅲ.C语言—程序设计—教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第120572号

策　　划 铭　政
责任编辑 徐丽萍

C#应用程序开发标准教程

秦　斌　曾　斌　编著

上海科学普及出版社出版发行

（上海中山北路832号　邮政编码200070）

各地新华书店经销　　北京市燕山印刷厂印刷

开本 787×1092　1/16　　印张 18　　字数 471000

2004年3月第1版　　2004年3月第1次印刷

ISBN 7-5427-2643-9 / TP · 508　　定价：23.00元

内 容 提 要

对开发人员而言，C#语言及相关的.NET Framework 开发环境是多年来最为成功的一项新技术。设计.NET 就是为了提供全新的开发环境，在此环境中，可以开发运行在Windows操作系统上的几乎所有的应用程序，而C#就是专门用于.NET的一个新型编程语言。

本书首先介绍了C#语言、.NET 的一些基本概念，C#的语法以及面向对象的编程思想；然后详细讲解了使用 C#语言在 Visual Studio.NET 开发环境中开发Windows 程序的各个方面，如界面、数据库、图形图像、文件以及网络等高级主体的方法；最后简单介绍了C#在 ASP.NET 中的应用。

本书既可以作为学习 C#和.NET 的入门教材，也可以作为具有一定编程基础开发人员的参考用书。

前　言

在过去的二十年内，C/C++已经成为在商用软件开发中广泛应用的开发语言。但是，C和C++都存在一些容易使开发者产生错误的特性，也可以说C/C++的灵活性牺牲了它们的开发效率。如果和其他的开发语言相比（如VB），相同功能的C/C++软件通常会需要更长的开发周期。正是由于C/C++程序开发的复杂性和较长的开发周期，所以许多C/C++开发人员都在寻找一种能够使功能和开发效率更加平衡的一种开发语言。

虽然目前有一些开发语言通过牺牲C/C++语言的灵活性（一些必要的灵活性）来换取开发效率的提高，但这些语言对开发人员产生了过多的限制（如限制使用底层控制代码等），并且所提供的对变量、函数的引用能力较差。因此，这些语言不能够轻易地与现存的系统相结合，并且不能够与当前的Web开发相结合。

一种合理的C/C++替代语言应该能够对现存和潜在的平台上的高效开发提供有效和有力的支持，并可以使Web开发非常方便地与现存的应用开发相结合。另外，C/C++开发人员都倾向于在必要的时候使用底层控制代码。在这种情况下，微软推出了一种名称为C#（发音为C Sharp）的开发语言。C#是一种先进的面向对象的开发语言，可以让开发人员快速地建立大范围的基于MS网络平台的应用，并且提供了大量的开发工具和服务以帮助开发人员开发基于计算和通信的各种应用。

由于C#是一种面向对象的开发语言，所以它非常适用于高层商业应用系统和底层系统的开发。即使是通过简单的C#构造也可将各种组件方便地转变为基于Web的应用，并且能够通过Internet被各种系统或者其他开发语言所开发的应用程序调用。

即使抛开上面所提到的优点，C#也可以为C/C++开发人员提供快捷的开发手段，而不需要牺牲任何C/C++语言的特性或优点。从继承角度来看，C#在更高层次上重新实现了C/C++，熟悉C/C++开发的人员可以很快地转变为C#开发人员。

.NET的出现，使传统的开发环境发生了重大改变。首先，在这个开发环境中集成了Visual Basic、Visual C++、C#和FoxPro。C#是.NET的默认语言，其优点使其成为.NET中的旗帜语言，而Visual Basic也第一次成为真正面向对象的语言。其次，.NET开发环境的适应性也大大增强了，它非常适用于Web应用软件的快速开发，用户可以轻松地用XML和Web服务进行跨平台计算，还可以快速开发中间层商务组件。最后，.NET的调试功能增强了，用户可以在.NET中端对端地调试Web应用软件，可以同时调试用不同语言开发的应用程序，还可以调试工程、进程和存储等。这些新特性都使用户开发程序的效率有了大幅度提高。

本书以Microsoft.NET Framework SDK为基础，完整地介绍了C#语言的语法以及各种特性，在后面还简单介绍了ASP.NET，让读者对Web开发也有一定的了解。

本书由秦斌、曾斌主编，同时参与本书编写和审校的还有任立功、董金波、杜同顺、李建慧、于晓利等，由于编者水平有限，书中疏漏之处在所难免，敬请广大读者批评指正。联系网址：http://www.china-ebooks.com。

编　者

2003年12月

目　录

目录

第1章 C#语言简介

在本章中用户将了解以下内容：

- Microsoft .NET 和.NET Framework 的概念。
- C#的由来、特点以及它在.NET 框架中的地位和作用。
- C#的开发环境 Visual Studio.NET。

1.1 C#的诞生与 Microsoft 的.NET 战略

在科技发展日新月异的今天，人们的身边随处可见“信息时代”的标志——计算机。不论是大到要好几个房间存放的大型服务器还是手机里面的小小芯片，都有着计算机的影子。计算机的发明，使人类社会的许多方面都有了长足的进步。而作为计算机灵魂的软件程序，在今天更是显得格外重要。因此编写程序所需要的编程语言与程序的设计方法也成了现在的热门学问。在学习 C#编程之前，如果能对编程语言的发展、演变以及 C#与其他编程语言的渊源有个大致了解，将有助于理解 C#的某些语言特点和编译机制。

1.1.1 编程语言的历史演变

目前世界上已有数百种编程语言，如 Basic、Pascal、C 和 C++等，这些名字人们都已经耳熟能详。每种编程语言都有自己独特的语法与规则，其中，有些编程语言是计算机直接就可以“看懂”的，有些则必须经过编译；有些语言只能在特定的计算机上才能工作，或者是为了特别的目的，如科学、研究或商业应用；然而，也有所谓的可以跨平台的编程语言，能够在多种计算机上执行。这些编程语言通常都遵循美国国家标准协会（American National Standards Institute，ANSI）制定的标准。

编程语言依据历史的演变与功能的不同，大致可以分为五类：机器语言、汇编语言、第三代语言、第四代语言和第五代语言。这些语言还可以划分为低级语言和高级语言两类。

低级语言是依赖于机器的程序语言。换句话说，这种语言只能在特定的计算机上执行，而且无法移植到其他计算机上，即使移植过去也不一定能够使用。机器语言和汇编语言就属于低级语言。

高级语言是独立于机器的程序语言。它不受计算机种类或是操作系统平台的限制，因此比较方便。第三代以后的编程语言都属于高级语言，但用高级语言编写的代码计算机并不能看懂，所以还需要编译程序将代码“翻译”成计算机能看得懂的机器语言才能被正确地执行，因此，高级语言编写的程序就执行效率而言低于低级语言，不过相对于高级语言的其他优点，这个缺点并不算太大，所以，目前大多数的编程场合都使用高级语言。

1.1.2 C家族

从名字上看，C语言与C++语言算得上是C#的前辈，这两个语言在编程语言的发展历史上有着不可磨灭的地位，即使到现在C++依然在主流编程语言中占据着一席之地。

C语言

C语言是由Brian Kernighan和Dennis Ritchie在20世纪70年代的Bell实验室开发的，最初是用来写系统软件，然而这个语言却十分简单和有弹性，并且很快地被应用在多种不同类型的程序中。今天，许多种类的软件都以C语言来编写，其中包括了操作系统与许多应用软件。

C语言也被称为一种"中级语言"，原因是它只提供给使用者一些最低限度的控制指令集，利用它们来定义一些更高层次的函数，以达到高级语言的效果。如此一来，它不但能处理低级语言擅长的位运算和地址寻址，再加上C语言的软件包中都提供了非常完整的链接库，使得它也可以像高级语言一样方便地使用。标准的C语言只有28个关键词，因此在很多平台上都有C语言的编译器。相同的程序，可以在许多不同的计算机上执行，只需重新编译一次即可，非常方便。

C++

尽管C语言有着许多的优点，但是C仍然有个大问题，那就是它是一种只适合进行结构化程序设计的语言。这意味着设计C程序时，程序员需要先从数据开始进行规划，然后写程序来处理这些数据。程序员最后发现如果能将数据和处理过程结合起来，将会使程序更为清晰易懂。这样的组合称为对象（object）和类（class）。用这种思路设计的程序称为面向对象设计（ODD，Object-oriented design）。

在20世纪80年代，Bjarne Stroustrup在Bell实验室开始设计一种新的语言，称为含有对象的C语言。这个语言以C为基础并增加了一些新的特点，其中最重要的就是面向对象的功能。经过多次改进与广泛讨论后，最后成为C++语言。

C++使程序员组织与处理信息时比其他程序语言更有效率，而且它是建立在C语言基础上的。事实上，大部分的C程序可以轻易转换成C++程序，这些程序通常不会用到C++的新特性，但是仍可正常执行。这一良好的兼容性使得C程序员能够方便地向C++过渡。

C#

C#在2000年9月推出，是Microsoft公司针对其.NET Framework所发展的新的程序语言。按Microsoft白皮书的说法，C#是一种简单的、现代化的、面向对象和类型安全的编程语言，它源自C和C++。C#旨在结合Microsoft Visual Basic的高效率和C++的原始功能。

但是，C#不是C++的升级版本，C#是专为.NET平台量身定做的程序语言，与.NET平台关系相当密切。C#是完全面向对象的语言，在C#中每一件东西都是对象，这几乎都与C++相同。C#的类型（Type）其实就是.NET Framework所提供的类型，而且C#本身并没有类库（Class Library），而是直接使用.NET Framework所提供的类库。此外，在C#中有关自动资源回收（Garbage Collection）、类别安全检查以及结构外的处理等都交由.NET Framework来处理。因此，C#程序语言非常适合但也仅仅适合开发.NET的应用程序。

尽管C#是Microsoft公司自己研发出来的全新的程序语言，但是仍然和C/C++有很多的相似之处，因此也算是C语言家族的一员。

1.1.3 C#产生的历史背景

C#这个名字从出现到成熟，其过程不过短短几年时间，但正如它的名字一样，已经给人留下了极为“锋利（Sharp）”的印象。那么，C#是如何产生的呢？

上一代编程语言的缺陷

在过去的二十年里，C/C++已经成为在商业软件开发领域中使用最广泛的语言。它们为程序员提供了十分灵活的操作，不过同时也牺牲了一定的效率。与Visual Basic 等语言相比，同等级别的C/C++应用程序往往需要更长时间来开发。由于C/C++语言的复杂性，许多程序员都试图寻找一种新的语言，希望能在功能与效率之间找到一个更为理想的平衡点。

目前有些语言，以牺牲灵活性的代价来提高效率，可是这些灵活性正是C/C++程序员所需要的。这些解决方案对编程人员的限制过多（如屏蔽一些底层代码控制的机制），其所提供的功能难以令人满意。这些语言无法方便地同早期的系统交互，也无法很好地和当前的网络编程相结合。

对于C/C++程序员来说，最理想的解决方案无疑是在快速开发程序的同时又可以调用底层平台的所有功能。他们想要一种和最新的网络标准保持同步并且能和已有的应用程序良好整合的环境。另外，一些C/C++开发人员还需要在必要的时候进行一些底层的编程。而对于VB程序员来说，当他们的程序要实现某些更复杂功能的时候，VB过于简单的特性就显得捉襟见肘了。

Java的兴起与Microsoft的战略转移

近年来，由于Internet的快速发展和普及，原本静态的网页已不能满足用户的需求。Sun公司推出的Java语言，就是为了使网页能和用户交互，从而达到互动的目的。Java语言改变了Internet和WWW的被动性。Java语言是一个纯面向对象的语言，初看之下，很多指令和C语言都是一样的。但在C语言中最重要但也最容易出问题的指针、函数和动态内存配置都没有了。这些特性使得Java语言写出来的程序，安全性大为提高，在网络上通过Web浏览器中的虚拟机来加以执行，能使网页更加生动。未来的计算机技术发展趋势就是以网络为基础的，用户的个人计算机中所有的资源都来自网上。

Java是一个优秀的产品，它以“一次编译，到处运行”的跨平台特性博得了广泛的支持。作为软件业界的龙头老大Microsoft公司通过Visual J++力图将Java限制到Windows平台之上，不可否认，Visual J++是迄今为止最高效的Java编译器和虚拟机，但Microsoft的这种捆绑行为遭到了Sun公司的起诉，理由是违反了Java的平台中立性。于是，Microsoft决定构建全新的.NET平台与之竞争。

Microsoft认为有必要设计一种最新的、面向对象的编程语言。它使得程序员可以快速地编写各种基于.NET平台的应用程序，集VC的灵活性和VB的易用性于一身，同时还具有与Java相同的跨平台特性，C#应运而生。

C#的诞生

应该说 Microsoft 还是非常敬业的，为了构建出一种基于.NET 平台的“完美的”编程语言来与 Java 竞争，Microsoft 动用了其最好的资源，包括世界上最优秀的语言专家来开发 C#。

Microsoft 为 C#制定了如下的目标，而在此之前没有其他任何一种语言能满足这些目标：

- 快速应用程序开发(RAD)：Microsoft 为 C#制定的最重要的目标之一就是支持 RAD。因特网应用程序必须以因特网节奏开发；一种新语言必须易于学习和调试，而且必须生成易于更新的代码。虽然 Delphi 和 VB 在这些方面很出色，但 C++却没有这么成功，因为 C++语言本身复杂而难以掌握，而且很少有有用的 C++库来提供简单的接口，此外，C++的手动内存管理和复杂的类型转换模型使它难以调试。
- 跨平台部署：根据定义，因特网语言应该支持跨平台的部署。因为因特网是不同系统的一个网络，所以必须将服务部署到各式各样的硬件和软件上。此外，客户端软件应该能够运行在多种类型的设备上，包括 PDA 和蜂窝式便携无线电话。这种灵活性事实上是对除 Java 之外所有语言的一种挑战。VB 只能生成在基于 Intel 微处理器的机器上运行的 Windows 应用程序；Delphi 也受到同样的限制，虽然 Borland 已经发布 Delphi for Linux，但它却不支持因特网设备。因此，VB 和 Delphi 都不符合跨平台部署的目标。
- 访问平台固有的资源：Microsoft 一直声称，开发人员需要访问平台固有的资源。虽然这个目标有维护 Windows 操作系统垄断地位之嫌，但对于编写强大的目标应用程序而言，这种访问有时是必不可少的。而 Java 通常不允许这种访问，Java 通过定义每种虚拟机实现的最小公分母标准来提供跨平台的部署。Java 开发人员是依据这种削弱的标准来编写代码，而不能利用只有某些平台才提供的服务。这样，Java 无法满足提供对平台固有资源的访问这一目标。

虽然这几个目标对于一种新语言来说要求是相当高的，但 Hejlsberg 交出了一份满意的答卷，C#产生了，从 C#的实现目标和 Hejlsberg 的个人经历来看，这种语言从 Java 和 Delphi 中借鉴东西甚至比从 C/C++借鉴的东西还要多。难怪 Mallard 公司的著名软件设计师 Michael L.Perry 这样评价 C#：“C#的父亲是 Java，母亲是 Delphi，而它出生在 C/C++家族”。这段评语精辟地指出了 C#与其他编程语言之间的继承关系。

2001 年年底，ECMA（欧洲计算机制造商协会）将 C#编程语言批准为一项标准（ECMA-334），经过标准化后的 C#可由任何厂商在任何平台上实现其开发工具及其支持软件。从此，带着 Microsoft“锋利”的梦想与雄心，C#正式成为编程语言大家庭中的一员。

1.1.4 Microsoft .NET 开发平台

从前面一节知道了 C#工作在.NET 平台之上，它的各种特性均和.NET 有着密切的联系，C#所能实现的许多强大功能都离不开.NET 平台的支持。那么什么是.NET 呢？

什么是.NET

.NET 基本上是已经历经三个重要阶段所产生出来的关键技术。第一个重要的阶段发生在 20 世纪 90 年代初期，这时候微软研发出一个技术，称为 OLE（Object Linking and Embedding），

OLE 让某个应用程序可以和另一个应用程序相互沟通。第二个重要的阶段是推出 COM 技术。随着程序功能的增加，程序代码量会以惊人的倍数增加。如果使用单一整体的做法，也就是只用一个庞大的可执行程序，开发或测试将会非常的困难与麻烦。因此解决方案就是使用组件式模型 COM（Component Object Model）。在组件式模型中，软件程序被分割成许多组件，而且每个组件提供不同的功能。因为每个组件都是独立的，因此对其他组件影响很小甚至没有影响。COM 通过将组件改变为通用、集成型的构件，使得采用不同编程语言进行合作编程成为现实。第三个阶段就是.NET，.NET 采用特别的方式编译及执行程序，先通过编译器编译，程序会被编译成微软中介语言 MSIL（Microsoft Intermediate Language）文件，MSIL 被启动时会启动 MSIL 编译器，将 MSIL 编译成机器码，然后载入 CPU 执行（如图 1-1 所示）。

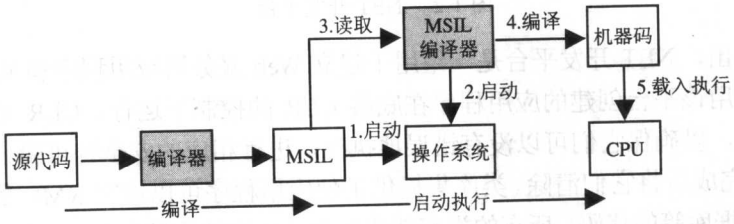

图 1-1　MSIL 原理

由于 MSIL 文件的成分并非机器码，因此每次执行都必须启动 MSIL 编译器，这样肯定会影响执行的效率，不过 MSIL 码十分接近机器码，从 MSIL 编译成机器码后再执行的速度就会很快（所以微软把 MSIL 编译器叫做 JIT 编译器，其中 JIT 是 Just In Time 的简称），因此对执行效能的影响相当有限。牺牲了执行效能，MSIL 到底能得到什么好处？首先是软件组件的通用，在.NET 平台下，不管哪一种程序语言，最终都要编译成 MSIL，因此.NET 就靠这个实现了统一平台下开发程序的目标。

.NET 开发平台的内容

.NET 开发平台包括以下内容：

- .NET Framework，它包括 Common Language Runtime（CLR），即通用语言运行环境，这是用于运行和加载应用程序的组件和新的类库，这些类库分级组织了开发者可以在他们的应用程序中用来显示图形用户界面、访问数据库和文件以及在 Web 上进行通信的代码集合。
- .NET 开发工具，包括：Visual Studio .NET Integrated Development Environment（IDE，Visual Studio .NET 集成开发环境），用来开发和测试应用程序；.NET 编程语言（例如 Visual Basic .NET 和本书讲述的 Visual C#），用来创建运行在 CLR 下并且使用类库的应用程序。
- ASP .NET，一个取代以前的 Active Server Pages（ASP）的特殊类库，用来创建动态的 Web 内容和 Web 服务器应用程序，这些都将采用诸如 HTML、XML 和 Simple Object Access Protocol（SOAP，简单对象访问协议）等 Internet 协议和数据格式。

有关.NET 平台组件的概貌，请看“.NET 开发平台”示意图（如图 1-2 所示）。

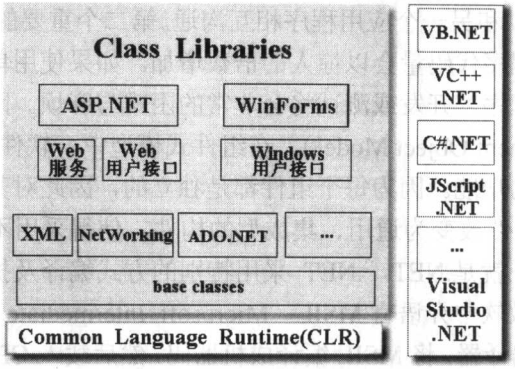

图 1-2 .NET 开发平台

由此可以看出，.NET 开发平台是一组用于建立 Web 服务器应用程序和 Windows 桌面应用程序的组件，用该平台创建的应用程序在底层 CLR 的控制下运行。CLR 是一个引擎，用来加载应用程序，以确保它们可以没有错误地执行，进行相应的安全许可验证，执行应用程序，然后在运行完成后将它们清除。类库集提供了使应用程序可以读写 XML 数据、在 Internet 上通信和访问数据库等的代码。所有的类库都建立在一个基础的类库之上，它提供管理使用最为频繁的数据类型（例如，数值或文本字符串）的功能，以及诸如文件输入/输出等底层功能。

Web 服务器应用程序通常依赖于 ASP.NET，一个处理 Web 请求的服务器端的库。ASP.NET 又依赖一个用于发送和接收 SOAP 信息的 Web Services 库，以及一个用于以浏览器接收用户输入并动态地生成 Web 页面以示响应的 Web 用户接口（UI，有时称作 Web 表单）。Windows 桌面应用程序通过使用 Win 表单库（也称作 Windows 表单）可以显示一个图形用户接口。

最后，Visual Studio.NET 提供了一个用于在该平台上创建应用程序的图形集成开发环境（Integrated Development Environment，IDE）。程序员可以使用一种或多种.NET 编程语言，来编写他们的代码，例如，Microsoft 自己的 Visual Basic .NET（VB.NET）、Visual C++、Visual C#和 JScript.NET 等。当然，用户也可以从第三方厂商获得其他的.NET 编程语言。

.NET 的架构改良并不是特别针对 C#设计的，然而作为一个.NET 的编程语言，C#从这些架构上获得了许多功能。因此了解.NET 也有助于帮助了解 C#。

.NET Framework

Framework（框架）是程序员对编程语言命令集的简称，.NET Framework 的意义在于用统一的命令集支持任何的编程语言。.NET Framework 是向程序员提供用来编写在 CLR 控制下运行的代码的组件。它们是单一、有序的分级组织，提供了一个庞大的功能集——从文件系统到对 XML 功能的网络访问的每一样功能。

.NET Framework 的目的是使开发人员更容易建立网络应用程序和网络服务。建立在操作系统最底层的服务，是管理运行代码需要的 CLR，这些代码可以用任何现代编程语言来编写。CLR 提供了许多服务，这些服务有助于简化代码开发和应用程序的开发，同时也将提高应用程序的可靠性。.NET Framework 包括一套可被开发者用于任何编程语言的类库。在此之上是许多应用程序模板，这些模板特定地为开发网络站点和网络服务提供高级组件和服务。

.NET Framework 引入了组合体的概念。一个组合体是一组资源和类型，并包括有关这些

资源和类型的元数据，也就是被作为一个单元配置的。元数据被称为组合体的名单，它包含像类型和资源表之类能被组合体外看得见的信息，这个名单也包括有关从属关系之类的信息，例如，组合体建立时的版本号。开发人员可以指定版本策略，以指示运行语言是否装入系统上已安装的依赖于组合体的最新版本，装入某一指定版本，或在编译时使用的版本。

组合体可以被一个应用程序私有，或被多个应用程序共享。一个组合体的多个版本可以同时配置在同一台机器上。应用程序配置信息定义了到何处去查找组合体，这样 runtime 就能为同时运行的两个不同的应用程序装入同一组合体的不同版本。这就消除了由组件版本的不兼容性引起的问题，提高了系统整体的稳定性。

.NET Framework 不仅规定代码访问的安全，还规定基于角色的安全。通过代码访问安全机制，开发人员能为应用程序指定完成工作所必需的权限。

总之，.NET Framework 是.NET 平台的基础架构，其强大功能来自于 CLR 环境和类库，CLR 和类库的紧密结合提供了不同系统之间交叉与综合的解决方案和服务。.NET Framework 创造了一个完全可操纵的、安全的应用执行环境，这不但使得应用程序的开发与发布更加简单，而且达成了多种编程语言之间的无缝集成。

专家指点

目前主流的 Windows 9x/2000/XP 操作系统并没有绑定.NET Framework，这意味着如果一台电脑上没有专门安装.NET Framework，那么除非它使用的是包含了.NET Framework 的 Windows.NET 系统，否则用 C#编写的很多程序将无法在这台电脑上运行。认识这一点很重要。

Microsoft 提供了用于开发基于.NET 应用程序的工具包——.NET Framework SDK，它包含了.NET Framework 的所有类库和编译器，也就是说，安装了.NET Framework SDK 之后，即使没有专用的 C#开发工具（如 Visual Studio.NET），也可以利用任何一个文本编辑工具（如记事本）进行 C#编程。

.NET Framework SDK 是免费的，可以从以下网址下载得到：

http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/000/976/msdncompositedoc.xml&frame=true。

.NET 的意义

.NET 平台的基本思想是：侧重点从连接到互联网的单一网站或设备上，转移到计算机、设备和服务群组上，使其通力合作，提供更广泛、更丰富的解决方案。用户将能够控制信息的传送方式、时间和内容。计算机、设备和服务将能够相辅相成，从而提供丰富的服务，而不是像孤岛那样，由用户提供惟一的集成。企业可以提供一种方式，允许用户将它们的产品和服务无缝地嵌入自己的电子架构中。这种思路将扩展对 20 世纪 80 年代 PC 功能的认识。

从这一基本思想来看，Microsoft.NET 平台将给 IT 产业和整个世界带来深远的影响。

1.1.5 C#的特点

C#可以说是 Microsoft 在不断的发展和实践过程中，锤炼出来的一个精品。它集众家之长，不仅安全，而且还非常的易于使用。在下面将详细介绍 C#的新特点。

简单

C#的简单性体现在以下几点：

- 在 C/C++语言中最遭非议的安全隐患——指针类型从 C#中消失了，当然，程序中还可以使用指针，但使用了指针的代码必须被声明为 unsafe，否则无法通过编译，这无疑大大增强了指针的安全性。既然禁用了指针，那么从前容易造成混淆的“::”或“->”操作符也就不存在了。
- C#从.NET 平台继承了自动内存管理和垃圾回收的特点，.NET 程序设计最吸引人的观点之一可能是内存的管理不再需要强制地当作程序代码的一部分。编写程序时，设计者不再需要自己编写代码来释放不需要的此前被占用掉的内存空间。这解决了 C++程序设计者过去编写 C++时所遭遇到的麻烦。通常，当程序配置内存，但是却没有用适当的方式释放它们时，会造成内存空间无法被还原成自由内存空间以被操作系统再利用，这种情况一般称为“内存漏泄”（memory leaks）。这种情况如果发生很多次，最后程序和操作系统将会因为内存不足而停止运行。而 C#的这项功能正好解决了程序员们常常提心吊胆的问题。
- 某些不安全的底层操作，比方说直接读写内存的操作不被允许了。
- 整型数值 0 和 1 不再作为布尔值出现。C#中的布尔值是纯粹的 true 和 false 值，而且没有更多的“=”和“==”操作符。“==”被用于进行比较操作，而“=”被用做赋值操作。

现代化

C#的现代化体现在以下两点：

- C#建立在目前的 Internet 潮流上，是一种新兴的编程语言，事物总是在不断发展进步的，编程语言也是一样，C#对于建立相互兼容的、可伸缩的和健壮的应用程序来说是非常强大和简单的。
- C#拥有内建的支持将任何组件转换成 Web Service 的服务，在任何平台上执行的任何应用程序都可以通过互联网来使用这个服务，从这个意义上来说已经实现了平台无关性。

面向对象

C#的面向对象的特性体现在以下几点：

- C#支持数据封装、继承、多态和对象接口，这些都是面向对象（OOP）最负盛名的特点。
- 一些在 Java 中都不是对象的元素，如 int、float 等简单数据类型，C#通过载入结构体（structs）来使原始数据类型变成对象。

类型安全

C#的类型安全特性体现在以下几点：

- 在 C#中，不能进行不安全的类型转换，如将 double 转换成 boolean，即使是强制性的显式类型转换也不行。

- 值类型被初始化为零值，而引用类型被编译器自动初始化为 Null 值。
- 数组类型下标从零开始而且进行越界检查。
- 类型的溢出情况将得到检查。

兼容性

C#的兼容性体现在以下几点：

- C#提供对 COM 和基于 Windows 的应用程序的原始的支持。
- 允许对原始指针的有限制的使用。
- 用户不再需要显式地实现 Unknown 以及其他 COM 接口，这些功能已经在 C#中内建。
- C#允许用户将指针作为不安全的代码段来操作老版本的代码。
- 由于.NET 平台的统一性，VB.NET 和其他中间代码语言中的组件可以在 C#中直接使用，这大大提高了代码的重复利用率。

可伸缩性和可升级性

C#的可伸缩性和可升级性体现在以下几点：

- .NET 引入了零部件的概念，它们通过其“手册”具有自描述的功能。手册确立了零部件的身份、版本、语言和数字签名等，零部件不需要在任何地方注册。
- 要扩展已编译的程序，只需要解除旧文件并用新的文件来升级它们。不需要注册动态链接库。
- 升级组件的过程只是一个错误探测的任务。对代码的修改能够影响现存的程序，C#在语言中支持版本修改。对接口和方法重载的支持使得复杂的程序框架能随着时间发展和进化。

1.1.6 C#与其他面向对象编程语言的比较

介绍了 C#的特点之后，不妨来看看它与其他面向对象编程语言之间的差别。

C#与 C++的比较

C#与 C++的差异主要有以下几点：

- C#对变量的声明顺序并不重要，而 C++需要在使用前事先声明。
- C#没有指针（可以通过 unsafe 来使用指针），而可以说没有指针就没有 C++。
- C#中对内存的直接操作被禁用。
- C#整型数值0和1不再作为Boolean值出现。C#中的Boolean值是纯粹的true和false，而且没有更多的“=”操作符和“==”操作符。“==”被用于进行比较操作，而“=”被用做赋值操作。
- C#没有前置作业，可以提高 C#编译器的速度。
- C#不支持多重继承。
- C#的环境会自动初始化程序中的变量。

C#与 Java 的比较

对于这两个竞争对手当然值得好好比较，表 1-1 列出了两者在各方面的不同。

表 1-1 Java 和 C#的比较

比较项目	Java	C#
开发公司	Sun	Microsoft
中间语言	JVM	MSIL
跨平台	Java 虚拟机	CLR 运行时
基础组件	Java 核心 API	.NET Framework SDK
Web 服务平台	Sun One	.NET
语言平台	Java runtime	CLR
操作重载	没有	有
结构	没有	有
Switch	只能用整数	可用表达式
Foreach 循环	For 或 While	有
使用者接口组件	Java Swing	Windows Forms 及 Web Forms
Web 服务	JDBC、EJB、JMS 以及 Java XML 链接库	ADO+以及 SOAP 为基础的 Web 服务
竞争特点	普及性的优势和制定产品兼容规格的公开标准	微软系列产品推广的策略

其实对许多程序员而言，C#最引人入胜的地方是它和 Java 不同的地方，而不是相似的地方，用户可以在实践中慢慢体会这一点。

三者的综合比较

C#的语法特点究竟是更像 C++还是更像 Java 呢？这是 C#问世以来就一直没有停止过争论的问题，通过表 1-2，用户心中也许会有一个自己的看法。

表 1-2 C++、C#和 Java 的综合比较

功　能	C++	C#	Java
平台无关	无	有	有
资源回收	无	有	有
Scalability	无	Namespace	Package
指针	无	无 但可通过 unsafe 使用	无
Template	有	无	无
结构	有	有	无
重载	有	有	无
学习难度	难	简单	简单