21世纪高等职业教育信息技术类规划教材
21 Shiji Gaodeng Zhiye Jiaoyu Xinxi Jishulei Guihua Jiaocai

C#面向对象基础教程

C# MIAN XIANG DUI XIANG JI CHU JIAO CHENG

宋楚平 周建辉 主编　王海峰 胡为民 副主编

- 夯实程序设计基础
- 贯彻面向对象思想
- 强调实际工作任务

人民邮电出版社
POSTS & TELECOM PRESS

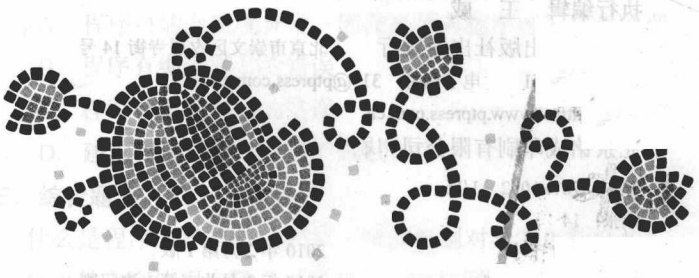
21世纪高等职业教育信息技术类规划教材

21 Shiji Gaodeng Zhiye Jiaoyu Xinxi Jishulei Guihua Jiaocai

C#面向对象基础教程

C# MIAN XIANG DUI XIANG JI CHU JIAO CHENG

宋楚平 周建辉 主编　王海峰 胡为民 副主编



人民邮电出版社

北京

图书在版编目（CIP）数据

C#面向对象基础教程 / 宋楚平，周建辉主编. --
北京 : 人民邮电出版社，2010.2
21世纪高等职业教育信息技术类规划教材
ISBN 978-7-115-21660-1

Ⅰ. ①C… Ⅱ. ①宋… ②周… Ⅲ. ①
C语言－程序设计－高等学校：技术学校－教材 Ⅳ.
①TP312

中国版本图书馆CIP数据核字(2009)第236400号

内 容 提 要

本书以 C# 语言面向对象的知识体系为主线，采用典型案例引导、任务驱动的模式进行编写。在内容的安排上遵循“实用、够用、应用”的原则，从基本知识、应用技能出发，介绍了使用 C# 语言开发控制台应用程序的基础知识、编程方法和实践技巧。

全书以案例和任务作为载体介绍 C# 语言的主要内容，并且配以大量的图表说明、解决思路提示和完成步骤介绍；在介绍 C# 语言的过程中，始终贯穿了面向对象的编程思想，力求使读者在学习 C# 语言的同时，深刻体会和理解面向对象编程的精髓和强大的功能。

本书可作为高职高专院校学生“C# 程序设计基础”课程的教材，也可供.NET 开发人员和程序设计爱好者参考使用。

21 世纪高等职业教育信息技术类规划教材

C#面向对象基础教程

◆ 主　　编　宋楚平　周建辉
副 主 编　王海峰　胡为民
责任编辑　潘春燕
执行编辑　王　威
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京铭成印刷有限公司印刷
◆ 开本：787×1092　1/16
印张：14
字数：357 千字　　2010 年 2 月第 1 版
印数：1－3 000 册　　2010 年 2 月北京第 1 次印刷
ISBN 978-7-115-21660-1

定价：26.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前言

C#语言是一种简单易用、面向对象且类型安全的编程语言，是微软将 Java 集成到.NET 中的产物，是整个.NET 平台的基础，也是未来主流的编程语言。现在，越来越多的高校将C#语言作为面向对象编程的课程教授，在高职院校基于项目、任务的教学课程改革开发的背景下，编写一本贯彻工学结合、突出实际应用，以典型任务承载教学内容的教材就变得迫切而具有现实意义。

我们对本书的体系结构和内容设置方面做了精心的设计，在章节的编排上以面向对象编程思想为脉络，以语言体系自然推进学习内容；在内容选取上充分考虑了工学结合的要求，重点选择在实际编程中广泛使用的知识点和程序设计方法，强调案例和任务的典型性；摒弃或者简单介绍那些相对比较落后，或者用得比较少的知识点和技术；在教学内容的安排上做到循序渐进、由浅入深，既考虑 C#语言解决问题的过程和步骤，又考虑学生的认知规律。

本书一共 12 章，分为 5 个单元。

第 1 单元：第 1 章～第 2 章，主要学习 C#简介、C#编程环境和 C#语言基础。

第 2 单元：第 3 章～第 6 章，主要学习面向对象编程基础、类型转换和命名空间、数组和方法、接口和抽象类。

第 3 单元：第 7 章～第 9 章，主要学习面向对象编程的核心思想：封装、继承和多态。

第 4 单元：第 10 章～第 11 章，主要学习集合、泛型和字符串。

第 5 单元：第 12 章，主要学习程序的异常处理，以及程序的调试方法和技术。

建议本教程的教学学时为 72 学时，学时分配如下表所示。当然可以根据学生实际情况对学时作必要的调整。

参考学时分配表

单元序号	授课内容	学时分配	
		理论	实践
1	C#简介、C#编程环境和 C#语言基础	8	8
2	面向对象编程基础、类型转换和命名空间、数组和方法、接口和抽象类	10	10
3	封装、继承和多态	8	8
4	集合、泛型和字符串	6	6
5	程序的异常处理、程序的调试方法和技术	4	4
合计		36	36

本书由宋楚平、周建辉任主编，王海峰、胡为民任副主编，其中王海峰编写了第 1 章～第 4 章，周建辉编写了第 5 章、第 8 章、第 9 章，胡为民编写了第 6 章、第 7 章，宋楚平编写了第 10 章～第 12 章。全书由宋楚平统稿，王路群审定。

本书得到江苏高校“青篮工程”的资助。

由于作者水平有限，书中难免有错漏之处，敬请读者批评指正。编者的 E-mail 为：ntscp@sina.com，欢迎联系和交流。

编 者

2009 年 12 月

目 录

第1章 C#及编程环境简介

学习目标

- 了解 C#的发展历史
- 认识.NET 框架的组成，能够描述.NET 框架各部分的功能
- 熟悉 C#编程环境，能够编写、编译、运行简单的控制台应用程序

1.1 C#历史

C 和 C++语言曾是被广泛使用的编程语言，尽管这两种语言为编程人员提供了丰富的控制软件功能的方法和灵活的程序结构，但是使用 C/C++语言开发 Windows 应用程序显然复杂了很多，特别是相对于 Microsoft 公司推出的 Visual Basic 语言来说，使用 C/C++语言开发具有 Windows 图形界面的软件不仅效率低，而且复杂程度高。

近年来，随着 Internet 的发展和普及，越来越多的应用程序基于网络运行，而以前的 C/C++应用程序在网络方面的功能不够强大，C/C++语言本身的发展已经远远跟不上网络技术的飞速发展了。

所以，无论是经验丰富的程序员，还是初涉编程语言的学习者都在寻找一种新的编程语言，希望这种编程语言简单、易学、易用，同时具有强大而丰富的功能。

对于已经具有 C/C++编程经验的人员而言，理想的解决方法：将 C/C++语言的能够利用开发平台底层的功能同 Visual Basic 语言的快速开发应用程序的特性结合起来。这个新的语言的应用程序开发环境最好能够将原有的应用程序较好地继承、发展，并且可以同步地生成基于 Internet 标准的应用程序。

对于初学者来说，新的语言要像 Basic 语言一样具有较少的关键字，又要像 C 语言一样具有松散简单的程序结构和灵活的编程语法，同时这门编程语言还要具有所有现代编程语言的特性，也就是面向对象的编程语言。简单地说，就是希望新的语言既像 Basic 语言一样简单，又像 C/C++语言一样具有很强大的编程能力。这样，学习者只需学习这

一种语言就可以了。

鉴于此，IT 行业巨头们纷纷推出了自己的解决方案，例如，Sun 公司推出了 Java 语言，Java 是一个开放、标准、通用的网络运算平台，由于其强大的兼容性和跨平台性，已经成为在 Internet 技术领域被广泛采用的一个成熟的技术平台。但由于纯 Java 编程的应用系统其运行速度太慢，而且基于 Java 开发的应用系统目前并没有完全实现跨平台运行，这使得 Java 仍旧未能完全取代 C/C++。行业的竞争使得 Microsoft 公司推出了其进军 Internet 的庞大计划：.NET 计划和该计划中催生的新的开发语言——C#。

Microsoft 公司的.NET 计划是一项非常庞大的工程，也是 Microsoft 公司今后几年发展的战略核心，以实现“在任何时间、任何地点，采用相应的设备获取所需的信息”的梦想。Visual Studio.NET 则是 Microsoft.NET 的技术开发平台，其重要性不言而喻，C#就集成在 Visual Studio.NET 中。

C#的出现给用户又多了一种选择，Microsoft 公司对 C#的定义：“C#是一种类型安全的、现代的、简单的、由 C 和 C++衍生出来的面向对象的编程语言。它是运行在.NET 框架之上的，并可立即被 C 和 C++的使用者所熟悉。C#的目的就是综合 Visual Basic 的高生产率和 C++的行动力”。从实际情况来看，这个定义应该是恰如其分的。

C#的发展经历了从最初的 1.2 版本到如今的 3.0 版本，具体如表 1.1 所示。

表 1.1　　C#的发展历程

时　间	C#版本	平台版本	集成开发工具
2003 年 4 月 25 日	C# 语言规范 1.2	.NET Framework 1.1	Visual Studio .NET 2003
2005 年 10 月 27 日	C# 语言规范 2.0	.NET Framework 2.0	Visual Studio .NET 2005
2006 年 11 月 6 日	C# 语言规范 2.0	.NET Framework 3.0	Visual Studio .NET 2005
2007 年 8 月 20 日	C# 语言规范 3.0	.NET Framework 3.0	Visual Studio .NET 2005
2007 年 11 月 19 日	C# 语言规范 3.0	.NET Framework 3.5	Visual Studio .NET 2008

C#在带来快速开发应用程序的能力的同时，并没有去掉 C 与 C++所具有的优良特性，它继承了 C 和 C++的优点。因为 C#是专门为.NET 应用而开发的语言，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，归纳起来，C#具有以下几个方面的特征。

（1）简洁的语法。C#取消了指针和相关操作符（例如，“::”、“->”等）。在默认的情况下，C#代码在.NET 框架提供的可操控环境下运行，不允许直接的内存操作。C#用真正的关键字换掉了伪关键字，每种 C#类型在.NET 类库中都有新名字。在 C#中简化了 C++语法中的一些冗余。

（2）完全的面向对象设计。C#具有面向对象语言所应有的一切特性：封装、继承和多态。

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）的机制来完成这些操作。

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。

C#中没有全局函数、全局变量、全局常数，一切都必须封装在一个类中。

整个 C#的类模型建立在.NET 虚拟对象系统（Visual Object System，VOS）的基础上，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。

（3）与 Web 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，SOAP(Simple Object Access Protocol)的使用使得大规模深层次的分布式开发成

为可能。由于有了 Web 服务框架的帮助，对用户来说，网络服务看起来就像是 C#的本地对象。用户能够利用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#语言结构即可。C#组件能够方便地为 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。

（4）完全的安全性与错误处理。安全性与错误处理能力是衡量一种编程语言是否优秀的重要依据。为了减少开发过程中的错误，C#会帮助用户通过更少的代码完成相同的功能。

.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。在默认情况下从 Internet 下载的代码都不允许访问任何本地文件和资源。

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）负责资源的释放与对象撤销时的内存清理工作。

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，当进行下行指向时，C#自动验证指向的有效性。C#中还提供了边界检查与溢出检查功能。

（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#会使开发人员更加轻易地开发和维护各种商业应用软件，从而保证软件系统中组件的升级。

（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。如果需要，C#允许用户将某些类或者类的某些方法声明为非安全的。这样一来，用户便能够使用指针、结构和静态数组，并且调用这些非安全的代码，而不会带来任何问题。

C#遵守.NET 公用语言规范（Common Language Specification，CLS），从而保证了 C#组件与其他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。

1.2　.NET 框架

如果脱离了.NET 平台而单纯学习 C#是没有太大意义的，所以用户必须先了解一些.NET 平台开发的基础知识，才能更深入了解 C#的特性。下面从软件开发模式不断发展、变革的原因中了解.NET 框架出现的必要性，随后进一步了解.NET 框架为软件开发带来的一些新的特性。

Windows 操作系统发展到今天，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，首先对这些变革做一个简要的回顾。

早先开发一个标准 Windows 应用程序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。那个时期操作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，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，所以，单纯从开发模式上讲，早先的软件开发还没有形成一个完善而科学的开发模式。

随着硬件的飞速发展，新一代操作系统的产生，软件开发方式的变革接踵而至。当这些强大的操作系统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。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。

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 Microsoft 公司推出并逐渐成为 Windows 操作系统的核心开发模式之一，这个方案就是组件对象模型（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。

最近几年来，随着 Internet 的高速发展，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，Microsoft 公司对 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。

通过上面的回顾，用户基本上可以对 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。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。这两种模式发展至今，都经历了长时间实际应用的考验，日趋成熟。

既然 Microsoft 公司拥有了成熟的软件开发模式，为什么还要提出全新的.NET 呢？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。在日趋成熟的现有应用程序架构中，仍然存在着许多的问题，主要有以下问题。

（1）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的共享动态链接库的版本很可能不是最新的。

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，便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”。

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，便有可能会影响到老版本软件的运行。

（2）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，而且要在注册表中进行更新工作，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复制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，因为很多与应用程序有密切关联的组件可能仍然存在于系统中。

（3）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，这个明显的缺陷使最终用户在安装

软件时，担心自己会受到伤害。因此，Windows 必须想办法解决此类安全问题。

针对上述列出的一些问题，Microsoft 公司提出了新的解决方案——.NET 框架（.NET Framework）。.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。

在基于.NET 框架的开发模式中，开发 Windows 应用程序的应用程序接口（API）被封装在各种“类”中，使用.NET 类库来开发应用程序。.NET 框架体系主要由两大部分组成，一部分是最基本的公共语言运行库（Common Language Runtime，CLR），另一部分是一些提供了具体功能的类库，例如，网络应用的 ASP.NET、数据库应用的 ADO.NET、Windows 窗口类等。运行库的类库之间以及它们同 Windows 操作系统之间的关系如图 1.1 所示。

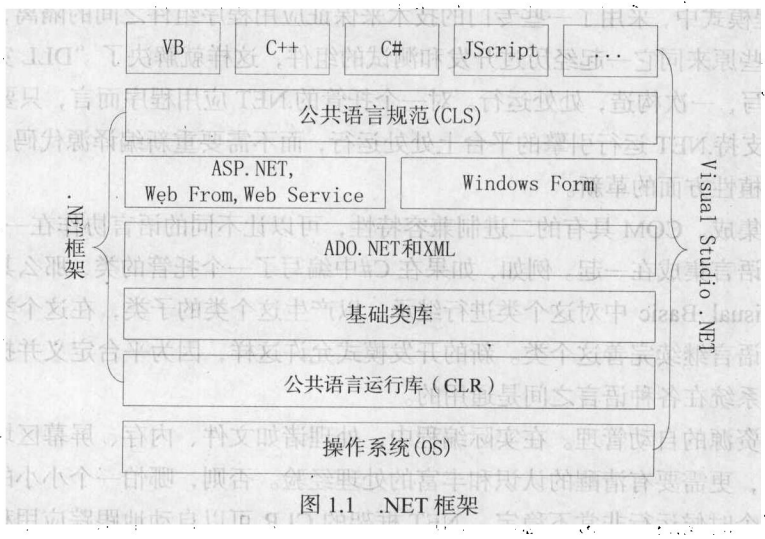


图 1.1　.NET 框架

.NET 框架由不同的组件组成，这些组件有助于创建和运行基于.NET 的应用程序。下面对这些组件做一些简单的介绍。

.NET 框架支持 3 种类型的用户界面：Web 窗体、Windows 窗体和控制台应用程序，Web 窗体利用 ASP.NET 和超文本传输协议来工作，Windows 窗体运行在 Win32 客户机上。

Web Service 是能被运行在 Internet 上的应用程序所共享的可编程 Web 组件。.NET 框架提供创建、测试和部署 Web Service 的工具和类。

ADO.NET 是新一代的数据库对象技术，它为无连接编程模型提供了更好的支持，也提供了 XML 扩展支持。

.NET Framework 类库体现了运行库特性，并且简化了.NET 应用程序的开发。.NET Framework 类库实现了.NET Framework 所有的应用程序（Web、Windows 和 Web Service）访问同一个.NET Framework 类库，并且这些类库被命名空间机制所控制。

公共语言运行库简化了应用程序的开发，提供了一个健壮的、安全的执行环境，支持多种语言，简化了应用程序的部署和管理。公共语言运行库也称为托管环境，在这个托管环境中自动提供诸如垃圾回收和安全性等公共服务。

符合公共语言规范（Common Language Specification，CLS）的任何语言都可以在公共语言运行库上运行。在.NET 框架中，支持 VB.NET、VC++.NET、C#.NET 和 JScript.NET 等几种语言。

使用.NET 框架有如下几点主要的优点。

（1）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。

（2）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，而在.NET 编程模式中，这些概念根本就不存在，更谈不上去理解。

（3）一次运行就会永远运行。前面提到过“DLL 灾难”这个术语，这个经典问题会时常给用户制造麻烦。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.NET 编程模式中，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。

（4）一次编写，一次构造，处处运行。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。

（5）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，如果在 C#中编写了一个托管的类，那么其他开发人员便可以轻松地在 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。

（6）内存和资源的自动管理。在实际编程中，处理诸如文件、内存、屏幕区域、网络连接和数据库等资源时，更需要有清醒的认识和丰富的处理经验。否则，哪怕一个小小的内存遗漏都有可能让软件在某个时候运行非常不稳定。.NET 框架的 CLR 可以自动地跟踪应用程序的资源使用情况，并保证不会出现资源泄漏。

（7）一致的异常处理。在 Windows 平台上进行编程，一件很烦人的事情就是提示错误信息的风格不一。某些函数提示错误的方式是返回标准 Win32 错误代码，而另一些函数是返回 HRESULTS 值（用于 COM），还有一些函数则会引发异常。在.NET 开发平台上，所有的错误处理都通过异常来报告，异常可以让开发人员从代码中去掉那些复杂的对可能错误的判断，使代码的编写、阅读以及维护工作大大简化。

（8）发布。在新平台上，安装软件意味着简单地将应用程序复制到目标机器上即可，而卸载应用程序则更为简单，只要删掉它们即可。

（9）安全。传统操作系统的安全机制在实现访问控制方面主要是面向个人用户账号的，在.NET 平台中，这种模式得到了改进，其核心内容是一种信任机制。如果应用程序是从物理介质（比如 CD-ROM）或其他可信任的服务器上安装的，那么这个应用程序是可以信任的。但是，对于那些来历不明的应用程序，诸如 Web 脚本、从因特网上下载的可执行文件或者可疑的 E-mail 附件，系统会使用代码访问安全机制对它们进行控制。

1.3 C#的编程环境

学习一种编程语言，首先要熟悉利用这种语言开发应用程序的运行环境。C#对编程环境的要求并不是很高，用户可以在文本编辑器中编写 C#代码，通过.NET 所提供的 C#编译器进行编译，就可以得到程序的输出结果；还可以在 VS.NET 的集成开发环境（IDE）中进行开发。

下面介绍 VS.NET 集成开发环境并且利用该环境开发一个简单的控制台应用程序。

1.3.1 VS.NET 集成开发环境

【学一学】

VS.NET 集成开发环境包括菜单栏、工具栏、类代码编辑窗口、工具箱、服务器资源管理器、解决方案资源管理器、“属性”窗口、“错误列表”窗口、“输出”窗口等，如图 1.2 所示。

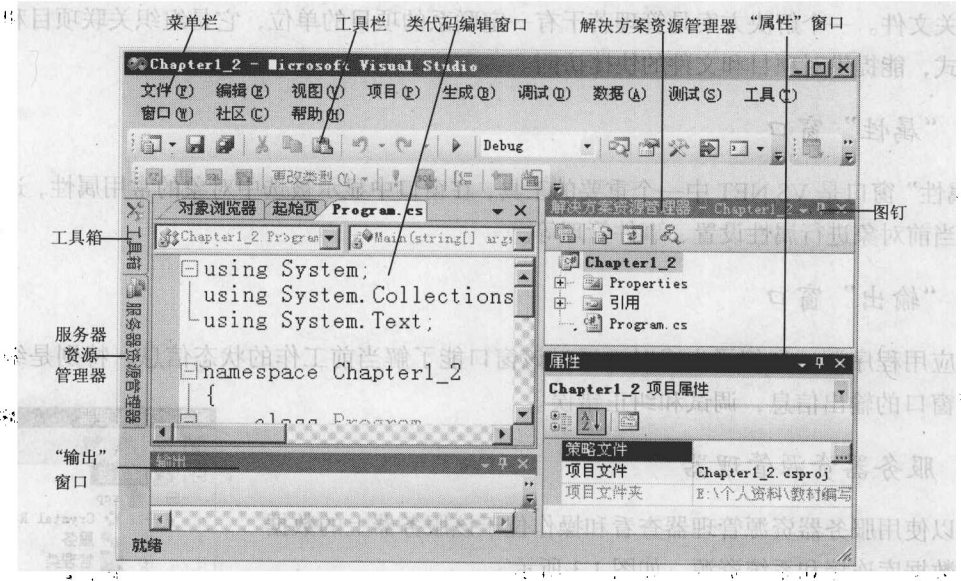


图 1.2　VS.NET 集成开发环境

1. 菜单栏

菜单栏包含丰富的菜单项，通过菜单项能实现程序开发中绝大部分的功能。菜单会随着不同项目和不同类型文件的变化而动态发生变化。

2. 工具栏

工具栏上包含了常见命令的快捷按钮，单击快捷按钮能执行相应的操作。工具栏上的快捷按钮也是智能变化的，它会随着当前任务的不同自动调整和改变命令按钮。

3. 类代码编辑窗口

类代码编辑窗口是编码人员必须熟练掌握的一个工具。该窗口主要用来输入、显示和编辑应用程序的代码。代码编辑器有以下几个方面的特点。

（1）智能缩进：自动为代码块设置合适的缩进量。

（2）自动语法检测：编辑器会自动检查编程人员书写的代码是否符合语法格式，并且在错误的语句下方加上红色的波浪线，同时将错误显示在错误列表窗口中。

（3）自动列出成员：当输入成员访问运算符“.”时，会在列表中显示所有有效的成员，供编程者选择。

（4）及时唤出：将光标定位在源代码某处，按“Ctrl+J”组合键，会及时唤出一个列表框，列表框中的内容与光标处的内容相同或最为接近，方便编程人员进行选择。

（5）快速浏览信息：将光标指向某个对象、方法、变量或常量时，就会显示光标所指内容的类型、原型、内容值等。

4. 解决方案资源管理器

在 VS.NET 开发环境中，项目是一个应用程序的编程单位，项目中主要包含类文件和其他的一些相关文件。一个解决方案是管理若干有一定联系的项目的单位，它是组织关联项目和文件的一种方式，能提供对项目和文件的快捷访问。

5. “属性”窗口

“属性”窗口是 VS.NET 中一个重要的工具，在窗口中显示被选中对象的常用属性，通过该窗口能对当前对象进行属性设置、事件管理等。

6. “输出”窗口

在应用程序调试和编译生成时，通过该窗口能了解当前工作的状态信息，特别是编码人员能利用窗口的输出信息，调试和纠正错误。

7. 服务器资源管理器

可以使用服务器资源管理器查看和操作位于该服务器上的数据链接、数据库连接和系统资源，如图 1.3 所示。

这样，就可以在不离开 VS.NET 的环境下，方便对所有的服务器和数据库资源进行管理和控制，以提高项目的开发效率。

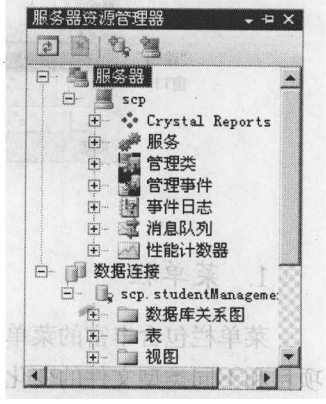


图 1.3 服务器资源管理器

8. 工具箱

在学习本教程时，可能还用不到工具箱。但在开发 Windows 应用程序、Web 应用程序时，工具箱的使用频率会非常高。工具箱提供了可视窗体或页面中可用的一些控件，对所有的控件按用途、以列表的形式进行分类，如图 1.4 所示。

图 1.4　工具箱

1.3.2　一个简单的控制台应用程序

【学一学】

在 Visual Studio .NET 2005 中创建一个控制台应用程序，来输出“Hello World!”。创建该程序的基本步骤如下。

（1）单击“开始”→“程序”→“Microsoft Visual Studio 2005”菜单项，打开 Microsoft Visual Studio 2005，如图 1.5 所示。

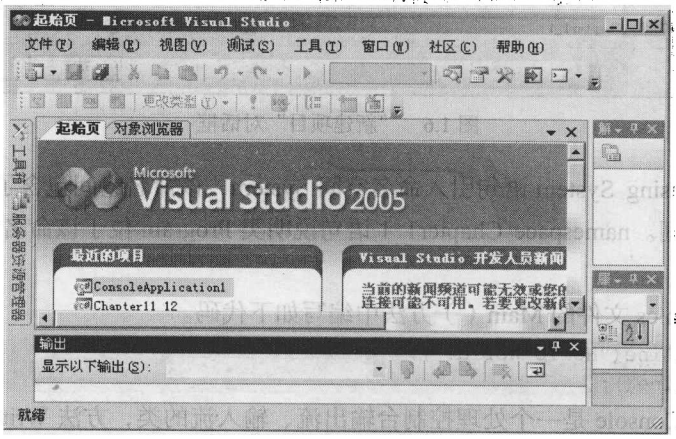


图 1.5　VS2005 主界面

（2）打开应用程序之后，单击“文件”→“新建”→“项目”菜单项，打开“新建项目”对话框，在该对话框中，“项目类型”选择“Visual C#”，“模板”选择“控制台应用程序”，“名称”命名为“Chapter01”，选择项目要保存的路径，操作后的结果如图 1.6 所示，然后单击“确定”按钮。

（3）在如图 1.7 所示的解决方案资源管理器中，有一个 Program.cs 文件，该文件是控制台应用程序的主文件，因为该文件包含一个 Main 主方法，程序从该方法入口，向下执行代码。

文件 Program.cs 的内容如下。

```
using System;
using System.Collections.Generic;
```