网络程序设计系列丛书

Visual C#.NET

网络核心编程

周存杰 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

网络程序设计系列丛书

Visual C#.NET 网络核心编程

周存杰　编著

清华大学出版社

（京）新登字 158 号

内 容 简 介

本书是关于 C#网络开发的教材，主要包括三个方面内容，首先简要介绍了有关 C#网络开发的基础知识；接着讲解基础服务器开发、基础客户端开发、FTP 开发、SMTP 开发、POP3 开发和远程控制开发；最后是高级应用，包括 Win32 网络组件开发、Web 数据库基础、Win32 异步套接字数据库开发、XML Web Services 开发以及一个完整的分布式网络应用程序开发实例。

本书适合于 C#开发人员进行网络开发，对 Visual C++.NET 和 Visual Basic.NET 开发人员也很有参考意义。

图书在版编目（CIP）数据

Visual C#.NET 网络核心编程/周存杰编著.-北京：清华大学出版社，2002.11
（网络程序设计系列丛书）
ISBN 7-302-05892-X
Ⅰ.V… Ⅱ.周… Ⅲ.C 语言-程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2002）第 070810 号

出 版 者：清华大学出版社（北京清华大学学研大厦，邮编：100084）
http://www.tup.tsinghua.edu.cn
责任编辑：龙啟铭
印 刷 者：北京市密云胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：24　**字数**：596 千字
版　　次：2002 年 11 月第 1 版　2002 年 11 月第 1 次印刷
书　　号：ISBN 7-302-05892-X/ TP・3498
印　　数：0001~5000
定　　价：35.00 元

前　言

C#是 Microsoft 为.NET 平台量身定做的语言。随着.NET 的推广，C#越来越成为一门重要的语言。许多企业已经开始尝试使用 C#开发应用软件。在.NET 平台上，C#开发者可以方便地扩展自己的应用。C#可以将大多数组件转变为 Web 服务，并且可以被任何在 Internet 上运行的应用程序调用，C#对这一特性提供内置的支持。更重要的是，Web 服务框架可以让任何 Web 服务看起来都类似于 C#的内置对象，从而使 Internet 上的站点变成一个可以互相交换组件的地方。这一特性使程序员们欣喜不已，这不但可以避免重复劳动、提高生产效率，而且可以使整个 Internet 变成一个大的操作系统。这无疑是下一代的开发理念。

本书共三部分：第一部分只有短短的一章，简明扼要地介绍有关网络开发的一些 C#基础知识；第二部分共六章，讲解基础服务器开发、基础客户端开发、FTP 开发、SMTP 开发、POP3 开发和远程控制开发；第三部分为高级应用，共五章，内容包括：Win32 网络组件开发、Web 数据库基础、Win32 异步套接字数据库开发、XML Web Services 开发和一个完整的分布式网络应用程序开发实例。

一门好的语言，仅仅开发出来还不够，更重要的是应用它、推广它、挖掘它。本书的所有内容都是编者两年来辛苦挖掘的结果，许多内容是编者的经验总结，希望本书能为读者今后从事 C#开发提供一臂之力。

本书由周存杰负责统稿。参加编写的有：周存杰、张芯娜、王丹丹、郭东、张显明、李飞、赵明信、梁越鸿、陈琳琳、王晓莉、林志东等。在书稿交出版社前，宋元时参加本书错字的校对工作。

在本书的编写过程中，宋元时先生给予了重要的帮助，在此表示感谢。清华大学出版社的龙啟铭先生在宏观上做了重要指导，使本书章节编排更加合理，提高了本书的质量，在此表示感谢。

由于时间仓促，本书肯定还有许多不足之处，望广大读者批评指正。联系信箱：zhou868@263.net。

编　者

目　录

第 1 章　C# 语法基础

本书是专门针对 C#网络高级开发而编写的教材，内容涉及常用网络协议和网络数据库的应用。由于本书篇幅的限制，在后面的章节里，往往是跳过语法知识而直接进入高级开发过程，这无疑增加了阅读本书的难度，会使许多不太熟悉网络的读者感到学习困难。为了帮助读者迅速掌握本书内容，本章首先介绍一下与网络有关的 C#基础知识，特别是与网络有关的 C#语法。如果读者已经对网络知识有了相当的了解，并且熟悉掌握了 C#的语法知识，则可跳过本章，直接阅读第 2 章。

1.1　C# 的特点

C#是 Microsoft 为.NET 平台量身定做的语言，它具有面向对象、面向 Web、语法简洁、功能强大、效率高、安全性强等特点。

1. 面向对象

C#是面向对象的语言。在 C#中，任何对象都会自动成为 COM 对象，开发者不再需要显式实现 IUnknown 和其他一些 COM 接口，同时可以方便而自然地使用现存的 COM 对象，而无需关心这些 COM 对象是否使用 C#开发。对于使用 C#的开发人员来讲，C#允许开发人员调用 OS 所提供的 API，在经过标记的代码区域内使用指针并手工管理内存分配，这样，以前的 C/C++代码依然可以重用，C/C++开发人员可以更快地熟悉和转向 C#，而无需放弃在以前开发中形成的开发习惯。由于 C#支持 COM，支持 API 调用，为开发人员提供了强大的开发控制功能。为了更好地实现公司的各种商业计划，在软件系统中，在商业流程和软件实现之间必须有紧密的联系。但大多数的开发语言都不能轻易地将各种应用逻辑与代码相联系。例如，开发人员会使用各种注释来标明各种类所代表的抽象商业对象。C#允许对任何对象使用预定义数据或是经过扩展的元数据，在系统结构中可以使用区域属性，并且将这些属性添加到类、接口或者其他元素上。开发者可以独立地测试各种元素上的属性。这使得收集区域中的对象属性或编写自动工具以保证区域中的类、接口是否被正确定义等工作变得十分简单。

2. 面向 Web

Microsoft 推出.NET 的主要目的是构建下一代 Internet，使 Internet 成为一个可以互相交换组件的地方。新的开发模式意味着需要更好地利用现有的各种 Web 标准，例如 XML，SOAP 等。C#之前的开发工具都是在 Internet 出现之前或是未得到充分应用之前出现的，所以都不能很好地适应目前 Web 技术的开发需要，而 C#开发者可以方便地在 Microsoft 网络平台上扩展自己的应用。C#可以将大多数组件转变为 Web 服务，并且可以被在 Internet 上

运行的各种平台的任何应用调用，C#对这一特性提供内置的支持。更重要的是，Web 服务框架使任何 Web 服务看起来都类似于 C#的内置对象，这样，开发人员在开发过程中可以继续使用他们已经具备的面向对象的开发方法和技巧。此外 C#还具有许多其他特性，使之成为最出色的 Internet 开发工具。例如，XML 目前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，以便更有效地处理各种数据。XML、SOAP 等是全球共有的协议和标准，不属于哪一家公司所有，对 XML、SOAP 的支持，意味着所有支持 XML 和 SOAP 的平台都可以访问 C#开发的 Web 资源。

3. 语法简洁

C#抛弃了许多旧的技术，如 ATL、MFC、C 运行库、标准模板库（STL）等类库。.NET 框架统一了编程类库，开发起来比使用这些旧技术容易得多。同时 C#将那些把 ATL 和 COM 搞得乱糟糟的伪关键字（如 OLE_COLOR，VARIANT_BOOL，DISPID_XXXXX 等）都抛弃了，代之以简明、清晰、易于理解的关键字。例如，在 C#中有一个 is 关键字，该关键字可以判断对象的类型，使用十分方便。下面的小程序演示了如何用 is 关键字判断对象的类型：

```
private  void button1_Click(object sender, System.EventArgs e)
{
    int i=100;
    object obj=i;
    if(obj is int)
    {
        textBox1.Text="对象是整型数值。";
    }
    else{textBox1.Text="对象不是整型数值。";}
}
```

执行该程序，textBox1 的文本框会出现“对象是整型数值。”一行字，如图 1-1 所示。

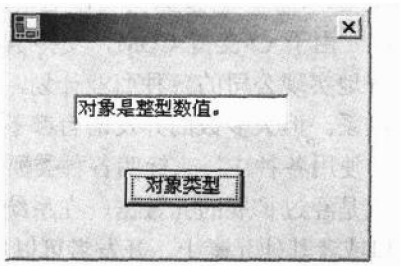

图 1-1 is 关键字

在 C#中对象的转换十分容易，任何对象都可以使用 ToString 方法转换成字符串类型；反之，使用特定对象的 Parse 方法可以把字符串类型转换成其他类型值。比如要把字符串“12345”转换成 Int32 的 12345，只要使用“Int32.Parse("12345")”即可。C#简洁的语法使得初学者非常容易入门，即使以前从来没有学过计算机语言的人也可以轻松入门。

4. 功能强大

C#可以开发任何传统风格的 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，甚至是 Web 页面、Web 服务、Web 组件，除了硬件驱动程

序，都可以用 C#开发出来。而且，C#的类可以从 VB、VC++中声明的类中派生出来。只要是使用运行库的语言，都可以做到在一种语言中声明类，而在另一种语言中派生类。Visual Studio 的调试器会完全支持跨语言的程序调试，在函数堆栈调试窗口的每个条目中都会显示堆栈中的函数是什么以及它们分别用何种语言写成；此外，开发人员甚至可以跨语言地处理程序中的异常错误。

5. 高效率

在 C#推出之前的 20 年里，C 和 C++已经成为商用软件开发的主流计算机语言，但是 C 和 C++都提供一些容易使开发者产生错误的特性，可以说，C 和 C++的灵活性是以牺牲开发效率为代价。如果和其他的开发语言相比（比如说 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++开发的复杂性和需要较长的开发周期，所以许多 C/C++开发人员都在寻找一种既灵活、又高效的开发语言。目前有些开发语言通过牺牲 C/C++语言的灵活性来换取开发效率，有些语言对开发人员限制过多，提供的通用命名能力很差。这些语言不能够轻易地与现存的系统相结合，并且不能与当前的 Web 开发相结合。合理的 C/C++替代语言应能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，使 Web 开发能非常方便地与现存的应用开发结合起来，并使 C/C++开发人员在必要时能使用底层代码。为了解决这个问题，Microsoft 推出了一种命名为 C#（发音为 C Sharp）的计算机语言。C#是一种先进的、面向对象的语言，它提供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，使开发人员可以快速地建立大范围的基于 MS 网络平台的应用。C#是一种面向对象的开发语言，可以大范围地适用于高层商业应用和底层系统的开发。C#可以方便地将各种组件转变为基于 Web 的应用，并且能够通过 Internet 被各种系统或其他开发语言所开发的应用调用。从开发效率讲，C#为开发人员提供了快速的开发手段而不牺牲任何 C/C++语言的特点或优点。从继承性讲，C#在更高层次上重新实现了 C/C++，使熟悉 C/C++的开发人员可以很快转变为 C#开发人员。

6. 安全性强

即使是优秀的 C/C++开发人员都难免在编码过程犯一些常见错误，比如错误地初始化一个变量，而这类错误有可能导致各种不可以预知的错误，并且难于发现。一旦这类错误在发现之前投入生产环境，为排除这些错误将会付出昂贵的代价。而 C#的先进设计思想可以消除 C/C++开发中的许多常见错误，比如：垃圾收集机制将减轻开发人员对内存的管理负担；C#中的变量将自动根据环境被初始化；变量是类型安全的；C#在语言中内置版本控制功能以减少更新组件时产生的错误等。

综上所述，C#是一门面向对象的、功能强大、高效、安全、语法简洁的崭新的计算机语言，它必将随着.NET 的推广而被广泛应用到各种软件开发中。

1.2　.NET 命名空间

要熟悉 C#语言，必须了解.NET 命名空间，这样才能在今后的网络高级开发中得心应手地使用各种空间下的资源。下面介绍.NET 命名空间。

- Microsoft.CSharp

包含支持用 C# 语言进行编译和代码生成的类。

- Microsoft.JScript

包含支持用 JScript 语言进行编译和代码生成的 JScript 运行库和类。

- Microsoft.VisualBasic

包含 Visual Basic .NET 运行库。此运行库与 Visual Basic .NET 语言一起使用。此命名空间还包含支持用 Visual Basic .NET 语言进行编译和代码生成的类。

- Microsoft.Vsa

包含将 .NET 框架脚本引擎的脚本集成到应用程序中以及在运行时编译和执行代码的接口。

- Microsoft.Win32

提供两种类型的类：处理由操作系统引发的事件的类和对系统注册表进行操作的类。

- System

最重要的类，包含用于定义常用值和引用数据类型、事件和事件处理程序、接口、属性和处理异常的基础类和基类。

- System.CodeDom

包含可用于表示源代码文档的元素和结构的类。

- System.CodeDom.Compiler

包含源代码模型的结构，管理源代码所生成和编译的类。

- System.Collections

包含定义各种对象集合（如列表、队列、位数组、散列表和词典）的接口和类。

- System.Collections.Specialized

包含专用的强类型集合；例如，链接表词典、位向量以及只包含字符串的集合。

- System.ComponentModel

提供用于实现组件和控件的运行时和设计时行为的类。此命名空间包括用于属性和类型转换器的实现、数据源绑定和组件授权的基类和接口。

- System.ComponentModel.Design

使开发人员可以生成自定义用户界面控件，并将这些控件包括在设计时环境中以便与供应商控件一起使用。

- System.ComponentModel.Design.Serialization

提供设计器所进行的组件序列化支持。此命名空间中的类可用于提供自定义序列化程序、管理特定类型的序列化、管理设计器加载和设计器序列化，以及优化设计器重新加载。

- System.Configuration

提供以编程方式访问 .NET 框架配置设置和处理配置文件（.config 文件）中的错误的类和接口。

- System.Configuration.Assemblies

包含用于配置程序集的类。

- System.Configuration.Install

提供为组件编写自定义安装程序的类。Installer 类是 .NET 框架中所有自定义安装程序的基类。

- System.Data

基本上由构成 ADO.NET 结构的类组成。使用 ADO.NET 结构可以生成用于有效管理多个数据源中的数据的组件。在断开连接的方案（如 Internet）中，ADO.NET 提供可以在多层系统中请求、更新和协调数据的工具。ADO.NET 结构也可以在客户端应用程序（如 Windows 窗体）或 ASP.NET 创建的 HTML 页中实现。

- System.Data.Common

包含由 .NET 数据提供程序共享的类。.NET 数据提供程序描述用于在托管空间中访问数据源（如数据库）的类的集合。

- System.Data.OleDb

封装 OLE DB .NET 数据提供程序。.NET 数据提供程序描述用于在托管空间中访问数据源（如数据库）的类的集合。

- System.Data.SqlClient

封装 SQL Server .NET 数据提供程序。.NET 数据提供程序描述用于在托管空间中访问数据源（如数据库）的类的集合。

- System.Data.SqlTypes

提供用于 SQL Server 中本机数据类型的类。这些类提供其他数据类型更安全、更快速的替代物。使用此命名空间中的类有助于防止在可能发生精度损失的情况中出现的类型转换错误。

- System.Diagnostics

提供允许与系统进程、事件日志和性能计数器进行交互的类。此命名空间还提供可以调试应用程序和跟踪代码执行的类。

- System.Diagnostics.SymbolStore

提供允许读取和写入调试符号信息的类。面向 .NET 框架的编译器可以将调试符号信息存储到程序员的数据库 (PDB) 文件中。调试器和代码分析器工具可以在运行时读取调试符号信息。

- System.DirectoryServices

提供从托管代码轻松访问 Active Directory 的方法。

- System.Drawing

提供对 GDI+ 基本形功能的访问。System.Drawing.Drawing2D，System.Drawing.Imaging 和 System.Drawing.Text 命名空间提供了更高级的功能。

- System.Drawing.Design

包含扩展设计时用户界面 (UI) 逻辑和绘制的类。可以进一步扩展此设计时功能，以创建自定义工具箱项、类型特定的值编辑器（可编辑和以图形方式表示所支持的类型的值）或类型转换器（可在特定类型之间转换值）。

- System.Drawing.Drawing2D

提供高级的二维和向量图形功能。此命名空间包括渐变画笔、Matrix 类（用于定义几何转换）和 GraphicsPath 类。

- System.Drawing.Imaging

提供高级的 GDI+ 图像处理功能。

- System.Drawing.Printing

提供与打印相关的服务。

- System.Drawing.Text

提供高级的 GDI+ 版式功能。此命名空间中的类使用户可以创建和使用字体集合。

- System.EnterpriseServices

为企业级应用程序提供重要的基础结构。COM+ 为企业级环境中部署的组件编程模型提供服务结构。此命名空间为 .NET 框架对象提供对 COM+ 服务的访问，使 .NET 框架对象更适用于企业级应用程序。

- System.EnterpriseServices.CompensatingResourceManager

提供在托管代码中使用补偿资源管理器 (CRM) 的类。CRM 是由 COM+ 提供的一项服务，使用户可以在 Microsoft 分布式事务处理协调器 (DTC) 事务中包括非事务性对象。虽然 CRM 不提供完整资源管理器的功能，但它们却通过恢复日志提供事务性原子性（全有或全无行为）和持久性。

- System.Globalization

包含定义区域性相关信息的类，这些信息包括语言、国家/地区、正在使用的日历、日期的格式模式、货币、数字以及字符串的排序顺序。

- System.IO

包含允许对数据流和文件进行同步和异步读写的类型。

- System.IO.IsolatedStorage

包含允许创建和使用独立存储区的类型。通过使用这些存储区，可以读写信任度较低的代码无法访问的数据，防止公开可保存在文件系统其他位置的敏感信息。数据存储在独立于当前用户和代码所在的程序集的数据舱中。

- System.Management

提供对一组丰富的管理信息和管理事件（有关符合 Windows 管理规范 (WMI) 基础结构的系统、设备和应用程序的）的访问。

- System.Management.Instrumentation

提供在规范应用程序管理并通过 WMI 向潜在用户公开管理信息和事件时必需的类。这样，Microsoft Application Center 或 Microsoft Operations Manager 等用户者就可以轻松地管理您的应用程序，而管理员脚本或其他应用程序（托管应用程序和非托管应用程序）也可以监视和配置您的应用程序。

- System.Messaging

提供用户连接、监视和管理网络上的消息队列以及发送、接收或查看消息的类。

- System.Net

为当前网络采用的多种协议提供简单的编程接口。WebRequest 和 WebResponse 类构成所谓的可插接式协议的基础，该协议是一种网络服务的实现，它使您可以开发使用 Internet 资源的应用程序，而不必考虑各个协议的具体细节。

- System.Net.Sockets

为需要严格控制网络访问的开发人员提供 Windows 套接字 (Winsock) 接口的托管实现。

- System.Reflection

包含提供已加载类型、方法和字段的托管视图的类和接口，并具有动态创建和调用类型的能力。

- System.Reflection.Emit

包含允许编译器或工具发出元数据和 Microsoft 中间语言 (MSIL) 并在磁盘上生成 PE 文件（可选）的类。这些类的主要客户端是脚本引擎和编译器。

- System.Resources

提供允许开发人员创建、存储和管理应用程序中使用的各种区域性特定资源的类和接口。

- System.Runtime.CompilerServices

为使用托管代码的编译器编写器提供功能，以影响在公共语言运行库运行时行为的元数据中指定的属性。此命名空间中的类只用于编译器编写器。

- System.Runtime.InteropServices

提供用于通过 .NET 访问 COM 对象和本机 API 的类的集合。此命名空间中的类型分为以下功能区：属性、异常、COM 类型的托管定义、包装、类型转换器和 Marshal 类。

- System.Runtime.InteropServices.Expando

包含 IExpando 接口，此接口允许通过添加或移除对象的成员来修改对象。

- System.Runtime.Remoting

提供允许开发人员创建和配置分布式应用程序的类和接口。

- System.Runtime.Remoting.Activation

提供支持服务器和客户端远程对象激活的类和对象。

- System.Runtime.Remoting.Channels

包含支持和处理信道和信道接收器的类，这些信道和信道接收器在客户端对远程对象调用方法时用作传输媒介。

- System.Runtime.Remoting.Channels.Http

包含使用 HTTP 协议与远程位置之间相互传输消息和对象的信道。默认情况下，HTTP 信道以 SOAP 格式对对象和方法调用进行编码以便传输，但在信道的配置属性中也可以指定其他编码和解码格式化程序接收器。

- System.Runtime.Remoting.Channels.Tcp

包含使用 TCP 协议与远程位置之间相互传输消息和对象的信道。默认情况下，TCP 信道以二进制格式对对象和方法调用进行编码以便传输，但在信道的配置属性中也可以指定其他编码和解码格式化程序接收器。

- System.Runtime.Remoting.Contexts

包含定义所有对象所驻留的上下文的对象。上下文是一个有序的属性序列，用于定义其中的对象所处的环境。上下文是在对象的激活过程中创建的，这些对象被配置为要求某些自动服务，如同步、事务、实时 (JIT) 激活、安全性等。多个对象可以存留在一个上下文内。