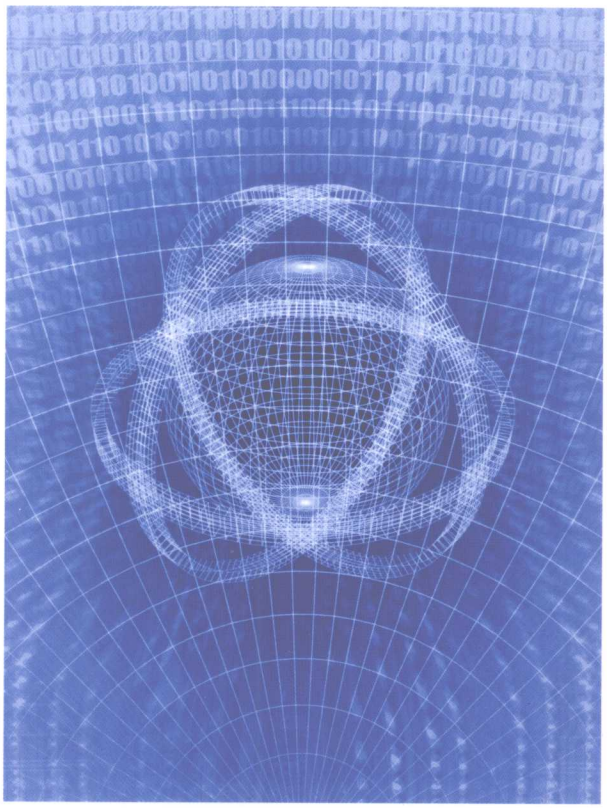
高等学校计算机应用规划教材

Visual C# 2005 程序设计教程

- Visual Studio 2005开发环境
- C# 2005语法基础
- 程序流程控制
- 数组与集合
- C#面向对象程序设计
- 域、属性、事件与方法
- 继承与多态
- C# 2005范型编程
- C#数据库编程与ADO.NET
- C# Web应用程序开发及ASP.NET



金雪云　周新伟　王雷　编著

清华大学出版社

高等学校计算机应用规划教材

Visual C# 2005 程序设计教程

金雪云　周新伟　王　雷　编著

清华大学出版社
北　京

内 容 简 介

本书详细介绍了 C# 程序设计的方方面面，并针对各章知识点附以大量的示例程序。通过本书的学习，读者可以由浅入深，逐步掌握 C#程序设计。

本书共 12 章，主要介绍了.NET Framework 及 Visual Studio 开发环境、C#语言基础及面向对象机制、C# Windows 程序设计、C# Web 程序设计、ADO.NET 及 C#数据库程序设计、C#泛型编程等内容。

本书难度适中，实例丰富，既适合 C# 的初学者阅读，也适合有一定开发经验的读者阅读，书中包含的大量实例对应用程序开发人员具有一定的参考价值。本书在各个章节的结尾附有不同类型的练习题，这些习题难易适中，有助于读者对所学知识点进行巩固、掌握，并能启发读者进行深层次的思考。

本书可作为各大中专院校计算机相关专业的教材或参考用书，也可作为读者的自学材料。

图书在版编目(CIP)数据

Visual C# 2005 程序设计教程/金雪云，周新伟，王雷 编著. —北京：清华大学出版社，2009.2

(高等学校计算机应用规划教材)

ISBN 978-7-302-19477-4

Ⅰ.V…　Ⅱ. ①金…②周…③王…　Ⅲ.C 语言—程序设计—高等学校—教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2009)第 016292 号

责任编辑：王　定　刘金喜
装帧设计：康　博
责任校对：胡雁翎
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市清华园胶印厂
装 订 者：三河市兴旺装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：20.75　**字　数**：479 千字
版　　次：2009 年 2 月第 1 版　　**印　次**：2009 年 2 月第1 次印刷
印　　数：1～5000
定　　价：32.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：025479-01

前　言

Microsoft .NET 是微软公司以服务方式递交软件的一种策略。它是微软公司的新战略，所有微软的产品都将围绕这个战略开发。微软为了推行.NET 战略，特别为.NET 平台设计了一种新语言——C#。

C#是由 C 和 C++发展而来的一种“简单、高效、面向对象、类型安全”的程序设计语言，其综合了 Visual Basic 的高效率和 C++的强大功能。C#是.NET 的关键语言，是整个.NET 平台的依托。设计 C# 是为了建立运行于.NET 平台上的、范围广泛的企业级应用程序。用 Visual C# 编写的代码被编译为托管代码，这意味着它将受益于公共语言运行库的服务。这些服务包括：语言互操作性、垃圾回收、增强的安全性以及改进的版本支持。

本书介绍了 C#语言编程的方方面面，共分为 12 章，首先介绍了.NET Framework 的相关概念、Visual Studio 开发环境以及 C#编程基础，接下来详细介绍了 C#面向对象机制以及如何利用 C#进行 Windows 及 Web 应用程序开发，并对 C#泛型编程给予简要说明，同时简要介绍了 ADO.NET 的相关知识。具体包括的内容如下。

第 1 章介绍了.NET Framework、Visual Studio 集成开发环境、MSDN 的安装与使用以及如何使用 Visual Studio 2005 集成开发环境创建基于 C#语言的 Windows 及 Web 应用程序。

第 2 章介绍了 C# 语言的相关基础知识和基本语法。

第 3 章介绍了 C# 中常见的程序结构。

第 4 章介绍了 C# 中数组与集合的使用。

第 5 章介绍了 C# 面向对象程序设计的基础。

第 6 章详细介绍了 C# 面向对象程序设计中的域、属性与事件。

第 7 章主要介绍了 C# 面向对象程序设计中的方法。

第 8 章介绍了 C# 语言中的继承与多态机制。

第 9 章简要介绍了 C# 中泛型编程的机制。

第 10 章详细介绍了利用 Visual Studio 2005 开发环境进行 Windows 应用程序开发的常用元素，包括常用控件、菜单、工具栏、对话框编程、GDI+编程等。

第 11 章简要介绍了 ADO.NET 与 C#数据库编程。

第 12 章介绍了 ASP.NET 及 C# Web 应用程序开发。

本书的特点在于理论与实际应用相结合，克服了理论型书籍难以动手实践和示例型图书难以理解和下手的不足。在理论方面，本书全面介绍了与 C#语言相关的知识点，使读者能够对 C#编程有一个完整的认识与把握；实践方面，本书各个章节中均附有难易度适中的示例，通过给出源代码，读者可在学习各章节知识点的基础上按照示例源代码进行实际操作。通过实际动手操作，巩固对知识点的理解。

本书由金雪云、周新伟、王雷负责编写，刘玉领参编了本书第 2 章和第 5 章的内容。

参加本书编写的还有杨卫、廖建军、付永华、叶明、崔宁、卢宏、汪昔玉、卫平峰、程冬丁、王勤、张锐、汪小锋、李葵、叶浩、肖飞、宋海剑、林勇及朱衡等人。全书由周新伟负责修改、定稿。在此对所有在本书编写过程中给予帮助的人一并表示感谢。

由于时间仓促，加之作者水平有限，书中不足和错误之处在所难免，敬请读者批评指正。

编　者

2008 年 12 月

目　录

第1章 .NET平台与Visual Studio 2005开发工具

Microsoft .NET 平台自从 2000 年 6 月推出以来，逐步获得了广大开发人员的认同与支持，目前已成为主流的开发平台。.NET 平台包含了 Microsoft 与软件开发相关的绝大部分产品，Microsoft 还为该平台设计了新的开发语言——C#。C#是从 C 和 C++派生来的一种简单、现代、面向对象和类型安全的编程语言。它保持了 C++中熟悉的语法和面向对象的特征，同时摒弃了 C++中复杂、易于出错的部分。C#语言综合了 C/C++的灵活性和 RAD 开发工具的高效率。不仅能适用于 Web 服务程序的开发与部署，更能高效地完成桌面应用系统的开发。

本章重点内容：

- .NET 平台与 C#语言
- C#开发工具 Visual Studio 2005 的使用
- SharpDevelop 和 UltraEdit 的使用

1.1 Microsoft.NET 平台

Microsoft.NET 是基于 Internet 的新一代开发平台，借助于.NET 平台，可以创建和使用基于 XML 的应用程序、进程和 Web 站点以及服务，它们之间可以按设计在任何平台或智能设备上共享和组合信息与功能，以向单位和个人提供定制好的解决方案。

.NET 的最终目的就是让用户能在任何地方、任何时间，以及利用任何设备都能够获取所需要的信息、文件和程序。而用户不需要知道这些东西存放在什么地方，甚至连如何获取等具体细节都不需要知道。他们只需要发出请求，然后等着接收结果就行了，而所有后台的复杂操作是被完全屏蔽起来的。

.NET 可以被认为是一个“商标”，在该商标下可以包含 Microsoft 的所有产品和服务，目前常被用到的有.NET Framework、.NET Framework SDK、Visual Studio.NET 2005、ADO.NET、ASP.NET 以及专门为.NET 平台设计的 C#语言等。

1.1.1　.NET Framework 2.0 概述

.NET Framework 是支持生成和运行下一代应用程序和 XML Web Services 的内部 Windows 组件，是.NET 战略的核心。.NET Framework 的目标是：

- 提供一个一致的面向对象的编程环境，而无论对象代码是在本地存储和执行，还是在本地执行但在 Internet 上分布，或者是在远程执行的。
- 提供一个将软件部署和版本控制冲突最小化的代码执行环境。
- 提供一个可提高代码(包括由未知的或不完全受信任的第三方创建的代码)执行安全性的代码执行环境。
- 提供一个可消除脚本环境或解释环境的性能问题的代码执行环境。
- 使开发人员的经验在面对类型大不相同的应用程序(如基于 Windows 的应用程序和基于 Web 的应用程序)时保持一致。
- 按照工业标准生成所有通信，以确保基于.NET Framework 的代码可与任何其他代码集成。

.NET Framework 具有两个主要组件：

- 公共语言运行库；
- .NET Framework 类库。

.NET Framework 目前的版本为 2.0，本书即以该版本为基础。最新的 3.0 版本将在稍后推出，.NET Framework 3.0 是 2.0 的附加版本，并且使用.NET Framework 2.0 的核心运行时组件，所以它完全向后兼容其早期版本。基于.NET Framework 2.0 的现有应用程序可继续运行，不需要任何修改。实质上，.NET 3.0 可以描述为：

.NET 3.0 = .NET 2.0 + WCF + WPF + WCS + WWF

用图形的方式描述.NET 3.0 的架构如图 1-1 所示。

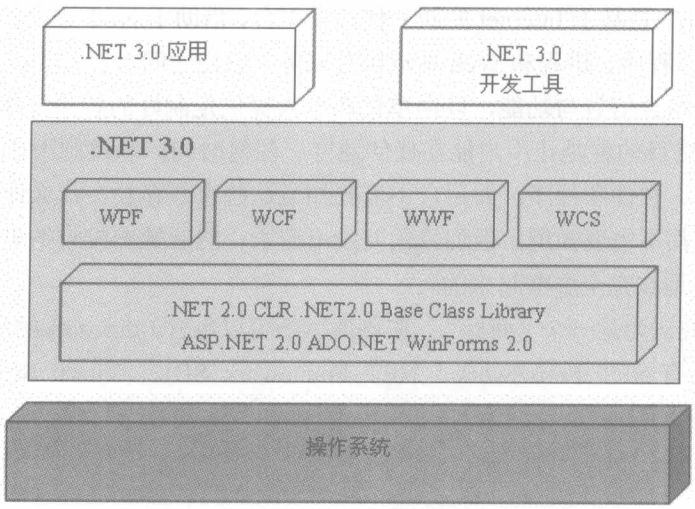


图 1-1　.NET 3.0 架构

.NET Framework 3.0 包含以下几个部分。

(1) Windows Presentation Foundation(WPF)

Windows 描述基础(WPF)提供了一种一致的方案来构建编程模型，并且支持使用更为丰富的控件和设计技术来开发 Windows 程序。一个开发出来的 WPF 程序最终能够被发行到桌面、Web 以及智能设备上。

WPF 已经提供了一些工具使开发者用来构建方案。其中，微软的 Expression 系列工具就包含了一些新的设计工具，它们允许你创建演示程序、网站以及交互式描述。

(2) Windows Communication Foundation(WCF)

Windows 通信基础(WCF)的核心目的是，允许程序与同一台计算机或网络上的或跨越互联网的其他程序实现通信。WCF 编程模型把 Web 服务、.NET 远程技术、分布式事务和消息队列统一到单个面向服务的编程模型中，从而实现真正意义上的分布式计算。

(3) Windows Workflow Foundation (WWF)

Windows 工作流基础(WWF)是一种定义、执行和管理工作流的微软技术。工作流由一系列的活动组成，开发者能够编写他们自己的域特定的活动，然后把它们应用于工作流中。.NET 框架 3.0/Windows 工作流基础还提供了一组涉及若干控制流构建方面的通用目的的活动。

Windows 工作流基础中还包括了 Visual Studio 2005 扩展。这些扩展包含一个允许用户设计工作流的可视化工作流设计器，一个支持用户调试工作流的可视化调试器，还有一个支持用户在 Visual Studio 2005 内编译工作流的工程系统。

(4) Windows CardSpace(WCS)

Windows CardSpace(WCS)实际上是更大的标识元系统的一部分。标识元系统完全基于开放的公共协议，它定义了一种全新的方式，能够使不同的数字标识技术在各个不同的平台(包括 Windows 以外的操作系统)和应用程序(包括 Internet Explorer 以外的 Web 浏览器)上使用。CardSpace 采用通用的方法来选择标识和其他 Windows 信息，因而在元系统中扮演着重要角色。并且，由于解决了基本的标识问题，CardSpace 也已经成为.NET Framework 3.0 的重要组成部分。

1.1.2　.NET Framework 类库

.NET Framework 类库是一个由 Microsoft .NET Framework SDK 中包含的类、接口和值类型组成的库。该库提供对系统功能的访问，是建立 .NET Framework 应用程序、组件和控件的基础。

该类库是面向对象的，这不但使.NET Framework 类库易于使用，而且还减少了学习.NET Framework 的新功能所需要的时间。.NET Framework 类库使开发人员能够完成一系列常见的编程任务，如字符串管理、数据收集、数据库连接以及文件访问等任务。除这些常见任务之外，类库还包括支持多种专用类型的开发方案。例如，可使用.NET Framework 开发下列类型的应用程序和服务：

- 控制台应用程序。
- Windows GUI 应用程序(Windows 窗体)。
- ASP.NET 应用程序。
- XML Web Services。
- Windows 服务。

例如，Windows 窗体类是一组综合性的可重用的类型，它们大大简化了 Windows GUI 的开发。如果要编写 ASP.NET Web 窗体应用程序，可使用 Web 窗体类。

.NET Framework 以命名空间的形式组织类库中的类，具有相似或关联功能的类被组织到一个特定的命名空间中，如：System、System.IO、System.Collections、System.Data、System.Xml 等，这些命名空间包含了与系统、系统输入输出、集合、数据以及 XML 等操作相关的类，编程时可以通过引用这些命名空间来使用相关类。

1.1.3 公共语言运行库

公共语言运行库(Common Language Runtime，CLR)也称为.NET 运行库，为.NET Framework 提供的运行时环境。C#中根据代码受 CLR 控制与否，将代码分为托管代码(managed code)和非托管代码，故托管代码是由公共语言运行库环境(而不是直接由操作系统)执行的代码。

托管代码是可以使用 20 多种支持 Microsoft .NET Framework 的高级语言编写的代码，它们包括：C#，J#，Microsoft Visual Basic .NET，Microsoft JScript .NET 以及 C++等。所有的语言共享统一的类库集合，并能被编码成为中间语言(IL)。运行库编译器(runtime-aware ompiler)在托管执行环境下编译中间语言(IL)使之成为本地可执行的代码，并使用数组边界和索引检查、异常处理、垃圾回收等手段确保类型的安全。

在托管执行环境中使用托管代码及其编译，可以避免许多典型的导致安全漏洞和不稳定程序的编程错误。同时，许多不可靠的设计也被自动地增强了安全性，例如类型安全检查、内存管理和释放无效对象。程序员可以花更多的精力关注程序的应用逻辑设计并可以减少代码的编写量。这就意味着更短的开发时间和更健壮的程序。

实质上，Microsoft 中间语言(IL)与 Java 字节代码共享一种理念：它们都是低级语言，语法很简单(使用数字代码，而不是文本代码)，可以非常快速地转换为内部机器码。正是 IL 具有的这些特征，保证了公共语言运行库具有平台无关性、高性能、语言互操作等优点。

1.1.4 C#语言

C#(发音为 C-Sharp)语言，是由微软推出的最新编程语言。它是针对.NET 平台而开发的一种面向对象编程语言。C#保持了 C++ 中熟悉的语法和面向对象的特征，同时摒弃了 C++中复杂、易于出错的部分。C#语言综合了 C/C++的灵活性和 RAD 开发工具的高效率。不仅能适用于 Web 服务程序的开发与部署，更能高效地完成桌面应用系统的开发。

作为一种针对.NET 平台开发的语言，C#继承了 C++强大的功能又兼顾 VB 等语言的易用性，同时也吸取了目前绝大多数开发平台的可以借鉴的优点。有关 C#语言的特点，请参阅 2.1.1 节。

1.1.5　理解命名空间

.NET Framework 提供了丰富的类资源，为了能在程序中引用这些类，必须先引用这些类所在的命名空间。一个命名空间是一个逻辑的命名系统，用来组织庞大的系统类资源，使开发者使用起来结构清晰、层次分明、使用简单，同时，开发者可以使用自定义的命名空间以解决大型应用中可能出现的名称冲突。事实上将命名空间看作是虚拟的组织架构更容易被理解和接受。如全国名称为张三的人很多，为了区分和引用某个特定、具体的某一位张三，现实生活中就通过指定他所属的不同的省、市、县等来唯一标识，.NET 中的命名空间类似与现实生活中的省、市、县等。

首先，名称同为张三的不同的个体才是程序设计中真正需要使用的，类似于命名空间中的一个个不同的类；

其次，张三所属的省、市、县等只是逻辑上的表示，就相当于程序中的不同层次的命名空间一样；

最后，省、市、县等不同层次的标识构成了一个树状结构，同样程序中的不同层次的命名空间也构成了类似的树状结构。

1. 定义命名空间

在 C#中定义命名空间的语法格式如下：

```
namespace SpaceName
{
    ......
}
```

其中 namespace 为声明命名空间的关键字，SpaceName 为命名空间的名称，在整个{}内的内容都属于名称为 SpaceName 的命名空间的范围。其中可以包含类、结构、枚举、委托和接口等可在程序中使用的类型。

请参考代码清单 1.1 所示例子(只需理解命名空间，不必细究语法内容)。

代码清单 1.1

```
using System;
namespace Space1
{
    class Test
    {
        static void Main()
        {
```

```
            Console.WriteLine("My NameSpace Space1");
        }
    }
}
```

至此，就声明了一个命名空间，在此可以对代码清单 1.1 所示例子进行修改，将输出内容写入另一个命名空间里的一个类中，在 Main 函数中引用该类完成输出功能。例子如代码清单 1.2 所示。

代码清单 1.2

```
using System;
namespace Space1
{
    class Test
    {
        static void Main()
        {
            A.Print a = new A.Print();
            a.DoPrint();
        }
    }
}

namespace A
{
    public class Print
    {
        public void DoPrint()
        {
            Console.WriteLine("My NameSpace A");
        }
    }
}
```

在开发大型应用程序的时候，通常会遇到在同一个文件中定义相同的类或其他可编程元素，但是它们执行的内容却是不一样的，这就很容易造成命名冲突。利用命名空间就可以很好地解决这种冲突，假定在同一范围内需要定义两个包含 DoPrint 接口的 Print 类，但是在 DoPrint 方法中所做的操作不同，这时就可以通过声明另外一个命名空间来包含另一个 Print 类，修改后的内容如代码清单 1.3 所示。

代码清单 1.3

```
using System;
namespace Space1
{
    class Test
```

```
    {
        static void Main()
        {
            A.Print a = new A.Print();
            a.DoPrint();
            B.Print b = new B.Print();
            b.DoPrint();
        }
    }
}

namespace A
{
    public class Print
    {
        public void DoPrint()
        {
            Console.WriteLine("My NameSpace A");
        }
    }
}
namespace B
{
    public class Print
    {
        public void DoPrint()
        {
            Console.WriteLine("My NameSpace B");
        }
    }
}
```

这样，有相同短名的不同的类就可以在同一个程序段中使用了。

2. 嵌套命名空间

命名空间内包含的可以是一个类、结构、枚举、委托和接口，同时也可以在命名空间中嵌套其他命名空间，从而构成树状层次结构。

```
namespace Wrox
{
  namespace ProCSharp
  {
    namespace Basics
    {
      class NamespaceExample
      {
```