高职高专计算机实用规划教材
——案例驱动与项目实践


本书特色：

- 以应用能力培养为基础，以提高职业素养为目的
- 将职业能力培养与课程学习相结合，精心设计案例与实践项目

# Visual C#
# 程序设计与项目实践

刘 伟 陈显珊
崔业勤 刘培涛 编著

NLIC 2970690153

■案例驱动

■项目实践

提供代码、电子教案下载
http://www.tup.com.cn

清华大学出版社

高职高专计算机实用规划教材——案例驱动与项目实践

# Visual C#程序设计与项目实践

刘　伟　陈显珊
崔业勤　刘培涛　编　著

清华大学出版社
北　京

## 内 容 简 介

本书以 Visual Studio.NET 为开发平台，以实例为中心，对 C#程序设计技术进行了深入浅出、循序渐进的论述，全面、详细、生动地介绍了 Visual C#的各种编程技巧。全书共分 12 章。具体包含 C#语言及其开发环境、C#语言基础、C#面向对象程序设计、接口和异常处理、Windows 应用程序设计、GDI+与图形图像编程、多线程程序设计、流和文件、ADO.NET 数据库访问技术、Web 应用程序开发等内容，最后详细描述了设备管理系统、网络计费系统两个典型实例的开发过程，使读者对使用 Visual C#进行应用程序开发有一个更加感性的认识。

本书从教学实践的角度出发，立足于提高学生的程序设计能力。全书理论分析透彻，实例丰富生动，内容由浅入深，能快速引导学生进入 Visual C#的编程世界。本书非常适合作为高职高专院校的程序设计课程教材，也可作为广大希望掌握 C#编程的程序设计人员的参考用书。



**图书在版编目(CIP)数据**

Visual C#程序设计与项目实践/刘伟，陈显珊，崔业勤，刘培涛编著. --北京：清华大学出版社，2011.1
(高职高专计算机实用规划教材——案例驱动与项目实践)
ISBN 978-7-302-24152-2
Ⅰ. V… Ⅱ. ①刘… ②陈… ③崔… ④刘… Ⅲ. 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(2010)第 219793 号

**责任编辑**：黄　飞
**装帧设计**：杨玉兰
**责任校对**：李玉萍
**责任印制**：何　芊
**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**：北京宏伟双华印刷有限公司
**装 订 者**：北京国马印刷厂
**经　　销**：全国新华书店
**开　　本**：185×260　**印　张**：22.25　**字　数**：539 千字
**版　　次**：2011 年 1 月第 1 版　　**印　次**：2011 年 1 月第1 次印刷
**印　　数**：1～4000
**定　　价**：35.00 元

产品编号：035228-01

# 前　言

在微软公司推出的 Visual Studio.NET 中文版开发工具中，Visual C#占有重要位置。C#语言并不是 C++或者 Visual Basic 的变形，它是一种全新的编程语言，它集合了 C++的强大功能以及 Visual Basic 的易用性等优点，主要面向需要使用 Microsoft .Net Framework 来创建应用程序的开发者。Visual Studio.NET 一经问世，就受到计算机应用开发者和广大师生的欢迎。

本书指导读者学习 C#的各种特性，并利用它们来构建运行于 Microsoft Windows 操作系统上的应用程序。学完本书之后，读者会对 C#有一个全面的理解，并能用它来构建 Windows 窗体应用程序，访问 Microsoft SQL Server 数据库，开发 ASP .NET Web 应用程序，以及创建并使用 Web 服务。具体内容如下。

第 1 章讲解 C#语言基础常识，包含 C#简介、.NET 框架概述、Visual C#集成开发环境简介、命令行工具的使用、MSDN 的使用、开发一个简单的应用程序、C#程序结构分析及 Visual Studio 2008 的安装。

第 2 章讲解 C#语言的基本理论知识，包含数据类型、变量和常量、运算符和表达式、字符串、程序流程控制语句、数组等内容。

第 3 章讲解 C#面向对象程序设计，内容包含类和对象、构造函数和析构函数、重载、继承和多态、访问修饰符和 this 关键字。

第 4 章讲解接口以及异常处理的知识，具体涉及接口的应用、异常处理、捕获异常等内容。

第 5 章讲解 Windows 应用程序设计的知识，内容包含 Windows 应用程序结构、窗体控件、常用控件介绍、一个“记事本”的开发实例。

第 6 章讲解 GDI+与图形图像编程技术，内容包含.NET 中的图形图像处理、GDI+中的图形图像处理以及绘制各种图形实例。

第 7 章讲解多线程程序设计的知识，内容涉及 C#多线程编程、线程的生命周期、线程同步、线程池等内容。

第 8 章讲解流和文件的知识，主要内容包含 I/O 流概述、文本处理和文件 I/O、文件目录管理。

第 9 章学习 ADO.NET 数据库访问技术，主要内容包含 ADO.NET 概述、ADO.NET 组件和对象、DataGridView 控件介绍及 ADO.NET 可视化编程。

第 10 章讲解 Web 应用程序开发，主要内容包含 Web 应用程序概述、Web 应用程序的代码模型、Web 应用程序的事件模型、ASP.NET 服务器控件及 Web 服务的应用。

第 11 章、第 12 章分别讲解设备管理系统、网络计费系统的完整开发过程，将本书所学的知识点贯穿起来，使读者对 C#应用程序开发有一个更加感性的认识。

本书从教学实践的角度出发，立足于提高学生的程序设计能力，理论分析透彻，实例丰富，内容由浅入深，能快速引导学生进入C#的编程世界，既可作为高职高专院校的程序设计课程教材，也可作为计算机软件培训教材，并可作为广大C#开发人员的参考用书。

本书由刘伟、陈显珊、崔业勤、刘培涛等编写。因编者水平有限，书中难免出现差错之处，恳请广大读者提出批评和建议。

# 目　录

# 第 1 章　C#语言及其开发环境

**教学提示**

本章主要介绍 C#语言的基本理论知识，这也是学习一种语言之前必须要做的一件事。了解 C#语言，会对将来的学习带来很大的帮助。本章还介绍了.NET 框架的概念以及开发.NET 应用程序的运行环境，最后介绍如何安装、部署 Visual Studio 2008 环境。

**教学目标**

了解 C#语言与其他语言的异同，为什么要学习 C#语言？C#语言为编程领域带来了什么？初步了解 C#语言结构，并通过上机操作，完成开发环境的安装。

## 1.1　C#语言简介

C#是一种简洁、类型安全的面向对象的程序设计语言，开发人员可以使用它来构建在.NET Framework 上运行的各种安全、可靠的应用程序。使用 C#，用户可以创建传统的 Windows 客户端应用程序、XML Web Services、分布式组件、客户端/服务器应用程序、数据库应用程序以及很多其他类型的程序。Microsoft Visual C#.NET 2008 提供了高级代码编辑器、方便的用户界面设计器、集成调试器和许多其他工具，以便在 C# 2.0 和.NET Framework 的基础上加快应用程序的开发。

C#语法表现力强，关键字不到 90 个，而且简单易学。C#的大括号语法使得任何熟悉 C、C++或 Java 的人都可以立即上手。了解上述任何一种语言的开发人员，通常在很短的时间内就可以使用 C#高效地工作。C#语法简化了 C++的诸多复杂性，同时提供了很多强大的功能。例如，可以为空的值类型、枚举、委托以及匿名方法提供直接内存访问，这些都是 Java 所不具备的。C#还支持泛型方法和类型，从而提供了更加出色的类型安全和性能。C#还提供了迭代器，允许对数组或集合类执行自定义的迭代行为，以简化客户端代码对它的使用。

作为一种面向对象的语言，C#支持封装、继承和多态性概念。所有的变量和方法，包括 Main 方法(应用程序的入口点)，都封装在类定义中。类可能直接从一个父类继承，但它可以实现任意数量的接口。重写父类中虚方法的各种方法，要求 override 关键字作为一种避免意外重定义的方式。在 C#中，结构类似于一个轻量类，它是一种堆栈分配的类型，可以实现接口，但不支持继承。

除了这些基本的面向对象的原理外，C#还通过几种创新的语言结构，加快了软件组件的开发，其中包括：

- 封装的方法签名(称为委托)，它实现了类型安全的事件通知。
- 属性(Property)，充当私有成员变量的访问器。
- 自定义属性(Attribute)，提供关于运行时类型的声明性元数据。

在C#中，如果需要与其他Windows软件(如COM对象或本机Win32 DLL)交互，可以通过一个称为Interop的过程来实现。互操作性使C#程序能够完成本机C++应用程序可以完成的任何任务。在直接内存访问必不可少的情况下，C#甚至支持指针和“不安全”代码的概念。

C#的生成过程比C/C++简单，比Java更为灵活。没有单独的头文件，也不要求按照特定顺序声明方法和类型。C#源文件可以定义任意数量的类、结构、接口和事件。

## 1.2 .NET框架概述

.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。.NET Framework旨在实现下列目标。

- 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运行环境。
- 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)执行安全性的运行环境。
- 提供一个可消除脚本环境或解释环境的性能问题的代码运行环境。
- 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，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：公共语言运行库和.NET Framework类库。公共语言运行库是.NET Framework的基础。可以将运行库看作一个在执行时管理代码的代理，它提供内存管理、线程管理和远程处理等核心服务，并且还强制实施严格的类型安全以及可提高安全性和可靠性的其他形式的代码准确性。事实上，代码管理的概念是运行库的基本原则。以运行库为目标的代码称为托管代码，而不以运行库为目标的代码则称为非托管代码。.NET Framework的另一个主要组件是类库，它是一个综合性的面向对象的可重用类型集合，可以使用它开发多种应用程序，这些应用程序包括传统的命令行或图形用户界面(GUI)应用程序，也包括基于ASP.NET所提供的最新的应用程序(如Web窗体和XML Web Services)。

.NET Framework可由非托管组件承载，这些组件将公共语言运行库加载到它们的进程中并启动托管代码的执行，从而创建一个可以同时利用托管和非托管功能的软件环境。.NET Framework不但提供若干个运行库宿主，而且还支持第三方运行库宿主的开发。

例如，ASP.NET承载运行库来为托管代码提供可伸缩的服务器端环境。ASP.NET直接使用运行库，以启用ASP.NET应用程序和XML Web Services。

Internet Explorer是承载运行库(以MIME类型扩展的形式)的非托管应用程序的一个示例。使用Internet Explorer承载运行库，用户能够在HTML文档中嵌入托管组件或Windows窗体控件。以这种方式承载运行库，使得托管移动代码(类似于Microsoft ActiveX控件)成为

可能，不过它需要进行重大改进(如不完全受信任的执行和独立的文件存储)，而这种改进只有托管代码才能提供。

图 1.1 显示了公共语言运行库和类库与应用程序之间以及与整个系统之间的关系。该插图还显示了托管代码如何在更大的结构内运行。

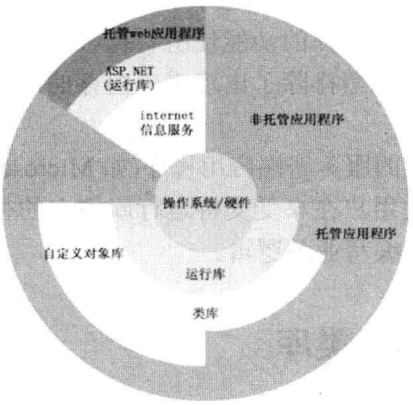


图 1.1　.NET Framework 环境

下面的章节将更加详细地描述.NET Framework 的主要组件和功能。

## 1.2.1　公共语言运行库的功能

公共语言运行库管理内存、线程执行、代码执行、代码安全验证、编译以及其他系统服务。这些功能是在公共语言运行库上运行的托管代码所固有的。

至于安全性，取决于包括托管组件的来源(如 Internet、企业网络或本地计算机)在内的一些因素，托管组件被赋予不同程度的信任。这意味着即使用在同一个活动应用程序中，托管组件既可能执行文件访问操作、注册表访问操作或其他须小心使用的功能，也可能不执行这些功能。

运行库强制实施代码访问安全。例如，用户可以相信嵌入网页中的可执行文件能够在屏幕上播放动画或唱歌，但不能访问他们的个人数据、文件系统或网络。这样，运行库的安全性功能就使通过 Internet 部署的合法软件能够具有特别丰富的功能。

运行库还通过通用类型系统(CTS)的严格类型验证和代码验证来加强代码可靠性。CTS 确保所有托管代码都是可以自我描述的。各种 Microsoft 和第三方语言编译器生成符合 CTS 的托管代码，这意味着托管代码可在严格实施类型保真和类型安全的同时使用其他托管类型和实例。

此外，运行库的托管环境还消除了许多常见的软件问题。例如，运行库自动处理对象布局并管理对对象的引用，不再使用它们时则将它们释放。这种自动内存管理解决了两个最常见的应用程序错误：内存泄漏和无效内存引用。

运行库还提高了开发人员的工作效率。例如，程序员可以用他们选择的开发语言来编写应用程序，却仍能充分利用其他开发人员使用其他语言编写的运行库、类库和组件。任何选择以运行库为目标的编译器供应商都可以这样做。以.NET Framework 为目标的语言编

译器使得用该语言编写的现有代码可以使用.NET Framework 的功能，这大大减轻了迁移现有应用程序的工作负担。

尽管运行库是为未来的软件设计的，但是它也支持现在和以前的软件。托管和非托管代码之间的互操作性使开发人员能够继续使用所需的 COM 组件和 DLL。

运行库旨在增强性能。尽管公共语言运行库提供许多标准运行库服务，但是它从不解释托管代码。一种称为实时(JIT)编译的功能使所有托管代码能够以它在其上执行的系统的本机语言运行。同时，内存管理器排除了出现零碎内存的可能性，增大了内存引用区域，进一步了提高性能。

最后，运行库可由高性能的服务器端应用程序(如 Microsoft SQL Server 和 Internet 信息服务(IIS)承载。此基础结构使用户在享受支持运行库宿主的行业最佳企业服务器的优越性能的同时，能够使用托管代码编写业务逻辑。

## 1.2.2 .NET Framework 类库

.NET Framework 类库作为一个与公共语言运行库紧密集成的可重用的类型集合，该类库是面向对象的。这不但使.NET Framework 类型易于使用，而且还减少了学习.NET Framework 的新功能所需要的时间。此外，第三方组件可与.NET Framework 中的类无缝集成。

例如，.NET Framework 集合类实现一组可用于开发用户自己的集合类的接口。用户的集合类将与.NET Framework 中的类无缝地混合。

正如用户对面向对象的类库所希望的那样，.NET Framework 类型使用户能够完成一系列常见编程任务(包括字符串管理、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开发方案的类型。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。

## 1.2.3 客户端应用程序开发

客户端应用程序在基于 Windows 的编程中最接近于传统风格的应用程序，即在桌面上显示窗口或窗体，从而使用户能够执行任务的应用程序。客户端应用程序不仅包括诸如字处理程序和电子表格等应用程序，还包括自定义的业务应用程序(如数据输入工具、报告工具等)。客户端应用程序通常使用窗口、菜单、按钮和其他 GUI 元素，并且它们可以访问本地资源(如文件系统)和外围设备(如打印机)。

另一种客户端应用程序是作为网页通过 Internet 部署的传统 ActiveX 控件(现在被托管 Windows 窗体控件所替代)。此类应用程序非常类似于其他客户端应用程序：它在本机执行，可以访问本地资源，并包含图形元素。

## 1.2.4　服务器应用程序开发

在托管领域，服务器端应用程序是通过运行库宿主实现的。非托管应用程序承载公共语言运行库，后者使用户的自定义托管代码可以控制服务器的行为。此模型在获得主服务器的性能和可伸缩性的同时提供给用户公共语言运行库和类库的所有功能。

图 1.2 显示了在不同服务器环境中运行托管代码的基本网络架构。应用程序逻辑通过托管代码执行时，服务器(如 IIS 和 SQL Server)可执行标准操作。

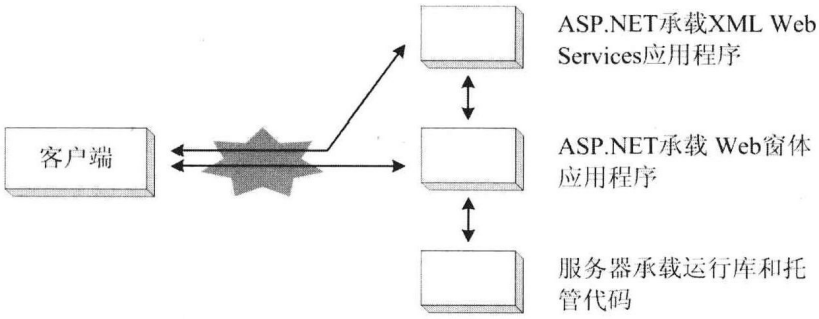


**图 1.2　在不同服务器环境中运行托管代码**

ASP.NET 是使开发人员能够使用.NET Framework 开发基于 Web 的应用程序的宿主环境。但是，ASP.NET 不止是一个运行库宿主，它还使用托管代码开发网站并通过 Internet 发布对象。Web 窗体和 XML Web Services 都将 IIS 和 ASP.NET 用作应用程序的发布机制，并且两者在.NET Framework 中都具有支持类集合。

XML Web Services 作为基于 Web 技术的重要发展，类似于常见网站的分布式服务器端应用程序组件。但是，与基于 Web 的应用程序不同，XML Web Services 组件不具有 UI 并且不以浏览器为目标。XML Web Services 由旨在供其他应用程序使用的可重用的软件组件组成，这些其他应用程序包括：传统的客户端应用程序，基于 Web 的应用程序，甚至是其他 XML Web Services。因此，XML Web Services 技术正迅速地将应用程序开发和部署推向高度分布式 Internet 环境。

## 1.2.5　编译.NET 代码

在.NET 框架内，应用程序可以用多种高级程序语言编写，如 VB.NET、C#乃至 COBOL.NET 等。而通过每种遵守.NET 规范的编程语言所编写的程序代码，首先都得通过一种初始编译步骤从源代码变成.NET 的公共标准语言：MSIL(Microsoft Intermediate Language，微软中介语言)。MSIL 是一种完整的和对象相关的语言，只有它才可能创建应用程序。.NET 应用程序是以 MSIL 的形式出现的，只有在程序执行的时候才通过即时编译器

(JIT)被编译为本机代码。

在编译过程中，JIT 编译器首次遭遇对象的索引就会装载匹配对象各个方法声明的对应程序。以后调用方法的时候就会编译其 IL，而方法的对应程序则被方法的编译后代码的地址所取代。这一过程在每次方法被首次调用的时候进行，产生的本机代码则被缓冲，以便在会话过程中下次装载 assembly 代码的时候可以使用。显然，这样的指令系统相比传统的编译语言需要更大的处理能力。

JIT 编译器则是默认的运行时配置，它会对其产生的代码进行即时优化。这样做无形中使.NET 具有超出传统预编译语言的一个优点：预编译语言只能对其处理的代码将要运行于其上的平台做一番大致的事前估计。JIT 编译器可以经过准确调节达到当前运行时的状态，结果可以完成一些预编译语言无法完成的工作。

- 更高效地利用和分配 CPU 寄存器。
- 在适当情况下实施低级代码优化，比如常量重叠、拷贝复制、取消范围检查、取消常规副表达式以及方法内联等。
- 在代码执行期间监控当前的物理和虚拟内存需求，从而更高效地利用内存。
- 产生特定的平台指令，以准确、充分地利用实际的处理器模式。

## 1.2.6 通用类型系统

通用类型系统定义了如何在运行库中声明、使用和管理类型，同时它也是运行库支持跨语言集成的一个重要组成部分。通用类型系统执行以下功能。

- 建立一个支持跨语言集成、类型安全和高性能代码执行的框架。
- 提供一个支持完整实现多种编程语言的面向对象的模型。
- 定义各语言必须遵守的规则，有助于确保用不同语言编写的对象能够交互作用。

通用类型系统支持两种一般类型，每种类都可细分成子类型。

### 1. 值类型

值类型直接包含它们的数据，值类型的实例要么在堆栈上，要么内联在结构中。值类型可以是内联的(由运行库实现)、用户定义的或枚举的。有关内联值类型的列表，请参见.NET Framework 类库。

### 2. 引用类型

引用类型存储对值的内存地址的引用，位于堆上。引用类型可以是自描述类型、指针类型或接口类型。引用类型的类型可以由自描述类型的值来确定。自描述类型进一步细分成数组和类类型。类类型是用户定义的类、装箱的值类型和委托。

作为值类型的变量，每个都有自己的数据副本，因此对一个变量的操作不会影响其他变量。作为引用类型的变量，可以引用同一对象，对一个变量的操作会影响另一个变量所引用的同一对象。

值是数据的二进制表示形式，类型提供了一种解释该数据的方式。值类型直接以类型数据的二进制表示形式存储。引用类型的值表示该类型的数据的位序列的位置。

每个值都有一个准确的类型，完全定义了值的表示形式和针对该值定义的操作。自描述类型的值称为对象。通过检查值，可以确定对象的准确类型，但不能处理值类型和指针类型。值可以有多种类型：一种实现某--接口的类型，其值也是该接口类型的值；另一种是从某一基本类型派生的类型，其值也是该基本类型的值。

运行库使用程序集来定位和加载类型。程序集清单包含运行库，用来解析在程序集范围内进行的所有类型引用的信息。

运行库中的类型名称有两个逻辑部分：程序集名称和程序集内类型的名称。有着相同名称但位于不同程序集内的类型被定义为两种不同的类型。

程序集在开发人员所看到的名称范围与运行库系统所看到的名称范围之间提供了一致性。开发人员在程序集的上下文中创作类型。开发人员正在构建的程序集内容确立了在运行时的可用名称范围。

运行库允许定义类型的成员：事件、字段、嵌套类型、方法和属性。每个成员都有一个签名。表 1.1 说明了.NET Framework 中使用的类型成员。

表 1.1　.NET Framework 中使用的类型成员

| 成　员 | 说　明 |
| --- | --- |
| 事件 | 定义可以响应的事件，并定义订阅、取消订阅及引发事件的方法。事件通常用于通知其他类型的状态改变 |
| 字段 | 描述并包含类型状态的一部分。字段可以是运行库支持的任何类型 |
| 嵌套类型 | 在封闭类型范围内定义类型 |
| 方法 | 描述可用于类型的操作。方法的签名指定其所有参数和返回值的允许类型。<br>构造函数是一种特殊类型的方法，可创建类型的新实例 |
| 属性 | 命名类型的值或状态，并定义获得或设置属性值的方法。属性可以是基元类型、基元类型的集合、用户定义的类型或用户定义类型的集合。属性通常用于使类型的公共接口独立于类型的实际表示形式 |

## 1.2.7　程序集

程序集是.NET Framework 应用程序的构造块。程序集构成了部署、版本控制、重复使用、激活范围控制和安全权限的基本单元。程序集是为协同工作而生成的类型和资源的集合，这些类型和资源构成了一个逻辑功能单元。程序集向公共语言运行库提供了类型实现所需要的信息。

程序集是.NET Framework 编程的基本组成部分。程序集执行以下功能。

- 包含公共语言运行库执行的代码。如果移植可执行(PE)文件没有关联的程序集清单，则不执行该文件中的 Microsoft 中间语言(MSIL)代码。
- 程序集形成安全边界。程序集就是在其中请求和授予权限的单元。
- 程序集形成类型边界。每一种类型的标识均包括该类型所驻留的程序集的名称。在一个程序集范围内加载的 MyType 类型，不同于在其他程序集范围内加载的 MyType 类型。

- 程序集形成引用范围边界。程序集的清单包含用于解析类型和满足资源请求的程序集元数据，它指定在该程序集之外公开的类型和资源。该清单还枚举它所依赖的其他程序集。
- 程序集形成版本边界。程序集是公共语言运行库中最小的可版本化单元，同一程序集中的所有类型和资源均会被版本化为一个单元。程序集的清单描述用户为任何依赖项程序集所指定的版本依赖性。
- 程序集形成部署单元。当一个应用程序启动时，只有该应用程序最初调用的程序集必须存在，其他程序集(例如本地化资源和包含实用工具类的程序集)才可以按需检索。这就使应用程序在第一次下载时保持精简。
- 程序集是支持并行执行的单元。

程序集可以是静态的或动态的。可以使用过去用来创建.dll 或.exe 文件的开发工具，例如 Visual Studio 2008。用户可以使用在.NET Framework SDK 中提供的工具来创建带有在其他开发环境中创建的模块的程序集，还可以使用公共语言运行库 API 来创建动态程序集。

## 1.3 Visual C#集成开发环境

在日常开发中，为了编写数据的增加、更新、修改和删除等功能而不得不面对枯燥的代码做重复的工作。Visual Studio 2008 的一些新的增强功能和 ADO.NET 2.0 的新特性让开发人员开发高可伸缩的多层数据库应用程序更加简单和快捷。

### 1.3.1 集成开发环境

Visual Studio 2008 是用于开发和维护托管的、本机的和混合模式的应用程序的集成开发环境(integrated development environment，IDE)。它提供了用于创建不同类别应用程序的多种项目模板，这些模板包括 Microsoft Windows 窗体、控制台、ASP.NET 网站、ASP.NET Web 服务、SmartPhone 2003、Windows CE 5.0 以及其他类型的应用程序。此外，它还提供了针对特定设备(如移动设备)的模板。而且，开发人员还可以根据需要选择不同的编程语言，包括 C#、Microsoft Visual Basic .NET 和托管的 C++等。

### 1.3.2 起始页

起始页是进入 Visual Studio 的入口屏幕，如图 1.3 所示。对于很多开发人员来说，起始页是 Visual Studio 呈现的第一个窗口。Visual Studio 2008 完全重新设计了起始页。Visual Studio 通常是从起始页开始的，如果起始页不可见，可以从【视图】菜单项中打开起始页，方法是选择【其他窗口】子菜单，然后选择【起始页】命令。

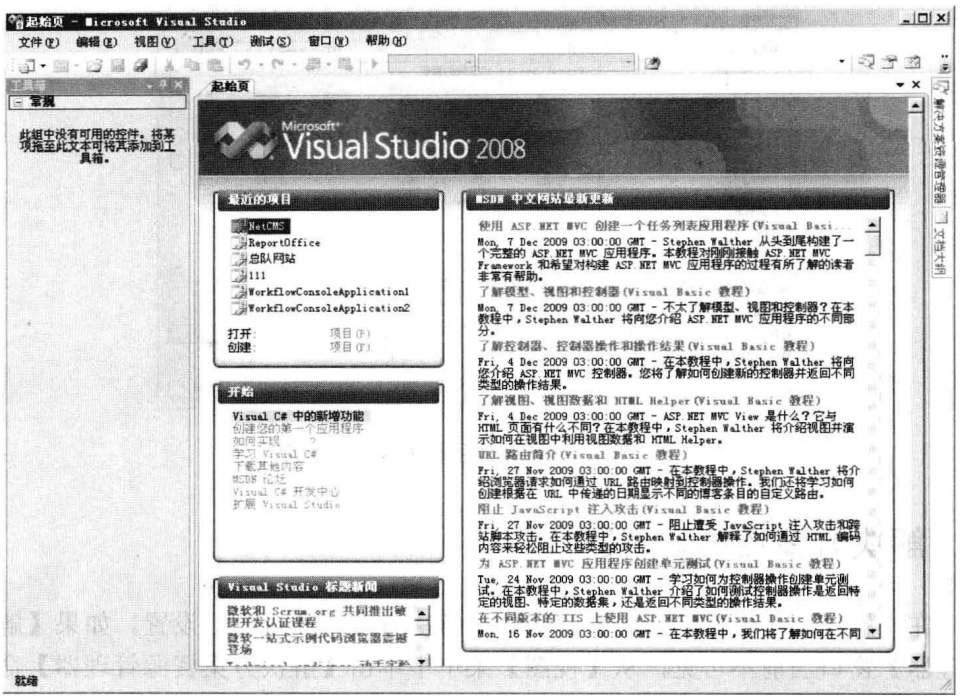


图 1.3　【起始页-Visual Studio 2008】窗口

重新设计的起始页包括以下 4 个窗格。

- 最近的项目：该窗格列出最近打开过的项目，在项目列表中进行选择，可以打开相应的项目。在该窗格底部的【打开】和【创建】按钮，分别用于打开和创建一个新项目。
- MSDN 中文网站：该窗格包含关于 Visual Studio 和 C#的最新新闻的链接。每个主题都是以预览的方式显示，单击相关链接可以观看完整的文章。
- 开始：该窗格包含对 Visual Studio 新手很有用的链接。
- Visual Studio 标题新闻：该窗格允许开发人员向 Microsoft 直接提交反馈。

## 1.3.3　创建项目

在 Visual Studio 中开发一个应用程序通常是从创建一个新项目开始。项目是 Visual Studio 的基本组织构件，文件、资源、引用以及其他应用程序构件在这里被组织起来。相关项目有时组合成解决方案，解决方案可以包含不同类型和语言的项目。例如，在开发周期中，通常将应用程序和相关类库组织到同一个解决方案中。添加一个新项目到一个当前打开的解决方案，还是在一个新解决方案里创建项目，这样的选择方式已经改变。图 1.4 显示的是一个【新建项目】对话框。