崔群法　唐有明　王俊伟　编著

Visual C# 2008

从入门到精通

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Visual C# 2008
从入门到精通

崔群法　唐有明　王俊伟　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

.NET已经成为Microsoft的支撑战略技术，Microsoft的很多产品现在都围绕.NET而展开。而C#是.NET的核心——.NET框架的“母语”，因此使用C#能够有效地开发基于.NET的应用程序。

本书以最新的.NET 3.5和Visual C# 2008为基础循序渐进地介绍了从入门到深入掌握Visual C# 2008所需的各个方面，包括开发环境的配置、C#语法、Windows应用程序开发、处理文件和注册表、创建基于Web的应用程序以及.NET 3.5的新增特性等。

本书同时还介绍了大量Visual C# 2008的开发经验，对使用中的重点、难点进行了专门的讲解，是一本有效、实用的从入门到精通级的学习指南。

本书适合于学习Visual C# 2008技术的初、中级读者使用，如果是初学者，本书将带你顺利进入Visual C# 2008开发领域；如果已有Visual C# 2008技术基础，本书将是你提高开发技能和更新开发理念的必备工具。

图书在版编目（CIP）数据

Visual C# 2008从入门到精通/崔群法等编著.—北京：电子工业出版社，2009.4
ISBN 978-7-121-08366-2

Ⅰ. V… Ⅱ. 崔… Ⅲ. C语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2009）第024847号

责任编辑：易 昆
印 刷：北京天竺颖华印刷厂
装 订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路173信箱 邮编：100036
北京市海淀区翠微东里甲2号 邮编：100036
开 本：787×1092 1/16 印张：29.5 字数：750千字
印 次：2009年4月第1次印刷
定 价：52.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

Visual C# 2008是开发C#的工具（类似VC++与C++的关系），即使用Visual Studio 2008集成开发环境进行C#应用程序的开发。C#本身是一门语言，是随.NET Framework一起由Microsoft开发和发布的。目前.NET Framework的最新版本是3.5，因此也被称为C# 3.5。本书以最新的Visual Studio 2008和C# 3.5为基础，循序渐进地介绍了从入门到深入掌握Visual C# 2008所需的各个方面。全书分5篇，共19章。

第一篇为C#入门篇，本篇首先向读者介绍了什么是.NET Framework和C#、开发环境的配置以及创建第一个示例程序所需要执行的操作。然后，针对C#的语法进行了详细介绍，包括表达式、控制语句、类型转换、C#结构及面向对象的内容，等等。

第二篇继续介绍C#语法，讨论更多C#特性，像声明和初始化数组、数组比较、使用集合、操作字符串和正则表达式、捕捉错误及异常的处理、C#的委托和事件，等等。

第三篇是本书的重点之一，主要介绍如何使用前面介绍的C#语法来开发各种应用程序，像创建Windows应用程序、使用MDI窗体、加载/写入/显示XML。还介绍了在Visual C# 2008中对文件、注册和数据库的各种操作，另外介绍了基于.NET Framework的各种编程实现，像创建Windows服务、使用线程和程序集，等等。

第四篇主要介绍使用Visual C# 2008创建Web应用程序方面的内容，包括ASP.NET的工作原理、ASP.NET的内置对象、数据控件和实现Ajax技术，等等。

第五篇针对.NET Framework 3.5而编写，主要向读者介绍了WCF、WPF及LINQ三个重要新增特性。

本书注重从初学者的认知规律出发，强调实用性、可操作性。使用通俗的语言对Visual C# 2008的基本概念和基本设计方法进行讲解，并且安排了大量典型和实用的例题供读者学习。

参加本书编写与制作的人员除封面署名者以外，还有祝红涛、唐晓阳、郑东方、李玺、刘小丹、郝春雨、王伟平、张水波、陈军红、李振、赵俊昌、刘海松、朱俊成、方宁、郑千忠、郭二鹏等人。在此，对他们表示衷心的感谢。由于编写时间仓促，加之作者水平有限，书中难免会有错误和疏漏指出，恳请广大读者批评和指正。

为了方便读者阅读，本书配套资料请登录“华信教育资源网”（http://www.hxedu.com.cn），在“资源下载”频道的“图书资源”栏目下载。

目　录

第一篇　入门必备

第二篇 编 程 基 础

第三篇 高 级 课 题

第四篇 Web 开 发

第五篇　使用.NET Framework 3.5

第一篇

入 门 必 备

第1章 .NET Framework和C#简介

Microsoft发布的.NET Framework简称为.NET，是支持生成和运行下一代应用程序和Web服务的内部Windows组件，提供托管执行环境、简化开发和部署以及与各种编程语言的集成。本章以最新的.NET Framework 3.5版本为例向读者介绍什么是.NET Framework及其重要组成部分、.NET Framework 3.5的新增特性，还包括如何配置.NET Framework的开发环境、使用C#创建.NET应用程序及编译器的知识等。

- 了解C#语言
- 理解.NET Framework概念
- 理解并熟悉公共语言运行时的概念及组成
- 熟悉CTS和CLS
- 理解中间语言的概述
- 了解CLR的执行过程和内存管理机制
- 熟悉.NET Framework类库的结构
- 了解.NET Framework中程序集的概念
- 掌握CLR命名空间的定义及引用方法
- 掌握.NET环境的配置方法
- 熟悉.NET程序的创建过程
- 熟悉C#命令行编译器的使用

1.1 .NET Framework与C#

C#是随.NET Framework一起发布的一种新语言，是一种崭新的面向对象的编程语言，强调以组件为基础的软件开发。C#不但结合了Visual Basic的简单易用性，同时也提供像Java和C++语言一样的灵活性和强大功能。C#在.NET Framework构架中扮演着一个重要角色，可以说是Microsoft公司面向下一代互联网软件和服务战略的重要内容，也是编写.NET Framework应用程序的首选。

1.1.1 C#概述

C#是用于创建运行在.NET公共语言运行时上的应用程序的语言之一，是Microsoft专门为使用.NET平台而创建的，它从C语言和C++语言演化而来，并且考虑了其他语言的许多优点，例如Visual Basic的易用性。

C#本身是面向对象的语言，然而C#进一步提供了对面向组件（Component Oriented）编程的支持。现代软件设计日益依赖于包含和描述功能包形式的软件组件。这种组件的关键在于，通过属性（Property）、方法（Method）和事件（Event）来提供编程模型；提供了关于组件的声明信息的属性（Attribute）；同时，还编入了自己的文档。C#提供的语言构造直接支持这些概述，这使得C#语言自然而然成为创建和使用软件组件的首选。

C#具有几个非常优秀的用于构造健壮和持久应用程序的特性，如下所示：

- 垃圾回收将自动回收不再使用的对象所占用的内存。
- 异常处理提供了结构化和错误检测的恢复方法。
- 类型安全的语言设计则避免了读取未初始化的变量、数组索引超出边界或执行未经检查的类型强制转换等情形。

此外，C#还具有一个统一的类型系统，所有C#类型（包括像int和string之类的基础数据类型）都继承于一个唯一的基类型：Object。因此，所有类型都共享一组通用操作，并且任何类型的值都能够以一致的方式进行存储、传递和操作。另外，C#同时支持用户定义的引用类型和值类型，既允许对象的动态分析，也允许轻量结构的内联存储。

为了确保C#程序和库能够以兼容的方式逐步演进，C#的设计中充分强调了版本控制。许多语言都不太重视这一点，导致采用那些语言编写的程序，常常因为其所依赖的库的更新而无法正常工作。C#的设计在某些方面直接考虑到了版本控制的需要，其中包括单独使用的virtual和override修改、方法重载决定规则以及对显式接口成员声明的支持。

总之，C#是一个易于使用的，能够开发出功能强大、安全、稳定应用程序的语言，在本书的第2章中将详细讲述C#的这些特性。

1.1.2 .NET Framework 3.5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，或者在远程执行。
- 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。

1. .NET Framework组件

.NET Framework主要有两个组件：公共语言运行时和.NET Framework类库。公共语言运行时是.NET Framework的基础。读者可以将运行时看做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时的基本原则。以运行时为目标的代码称为托管代码，不以运行时为目标的代码称为非托管代码。

.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。读者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所提供的最新创新的应用程序（如Web窗体和XML Web Services）。

在图1-1所示的.NET Framework平台上显示了运行时和类库与应用程序以及与整个系统之间的关系。

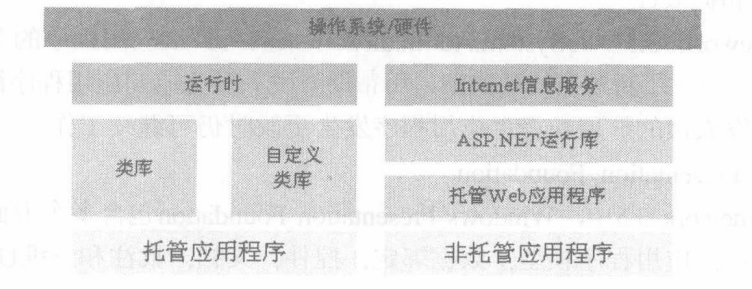

图1-1 .NET Framework平台

2. .NET Framework 3.5简介

.NET Framework 3.5版以.NET Framework 2.0版和.NET Framework 3.0版为基础，包括.NET Framework 2.0和3.0版的Service Pack。其中主要包括如下的组件：

- .NET Framework 2.0。
- .NET Framework 2.0 Service Pack 1，更新包含在.NET Framework 2.0中的程序集。
- .NET Framework 3.0，使用.NET Framework 2.0或.NET Framework 2.0 SP1（如果已安装）中存在的程序集，并且包含.NET Framework 3.0中引入的技术所必需的程序集。例如，Windows Presentation Foundation（WPF）所必需的PresentationFramework.dll和PresentationCore.dll就随.NET Framework 3.0一起安装。
- .NET Framework 3.0 Service Pack 1，更新在.NET Framework 3.0中引入的程序集。
- 一些新程序集，它们为.NET Framework 2.0和3.0提供附加功能，同时还提供.NET Framework 3.5中新采用的技术。

如果在计算机上安装 .NET Framework 3.5时缺少上述任何组件，则会自动安装上它们。应用程序无论针对的是.NET Framework 2.0、3.0还是3.5版，都使用相同的程序集。例如，对于使用WPF并针对.NET Framework 3.0的应用程序，其所使用的mscorlib程序集实例与使用Windows窗体并针对.NET Framework 2.0的应用程序是相同的。如果.NET Framework 2.0 SP1已安装在计算机上，则mscorlib.dll已更新，并且两个应用程序都将使用mscorlib.dll的更新版本。

.NET Framework 2.0、3.0和3.5版之间的关系不同于1.0、1.1和2.0版之间的关系。.NET Framework 1.0、1.1和2.0版是彼此完全独立的，对于其中任何一个版本来说，无论计算机上是否存在其他版本，自己都可以存在于该计算机上。当1.0、1.1和2.0版位于同一台计算机上时，每个版本都有自己的公共语言运行时、类库和编译器，等等。应用程序可以选择是针对1.0、1.1还是2.0版。

3. .NET Framework 3.5重要新功能

.NET Framework 3.5为2.0和3.0中的技术引入了新功能，并以新程序集的形式引入了其他技术。下列技术是随.NET Framework 3.5引入的：

• LINQ

LINQ（Language Integrate Query，语言集成查询）是Visual Studio 2008和.NET Framework 3.5中的新功能。LINQ将强大的查询功能扩展到C#和Visual Basic的语法中，并采用标准的、易于学习的查询模式。可以对此技术进行扩展以支持几乎任何类型的数据存储。

• 外接程序和扩展性

.NET Framework 3.5中的System.AddIn.dll程序集向可扩展应用程序的开发人员提供了强大而灵活的支持。引入了新的结构和模型，可帮助开发人员完成向应用程序添加扩展性的初始工作，并确保开发人员的扩展在宿主应用程序发生更改时仍可继续工作。

• Windows Presentation Foundation

在.NET Framework 3.5中，Windows Presentation Foundation包含多个方面的更改和改进，其中包括版本控制、应用程序模型、数据绑定、控件、文档、批注和三维UI元素。

• WCF和ASP.NET Ajax集成

WCF与ASP.NET中的异步JavaScript和XML（Ajax）功能的集成提供了一个端对端的编程模型，可用于构建可以使用WCF服务的Web应用程序。在Ajax样式的Web应用程序中，客户端（例如，Web应用程序中的浏览器）通过使用异步请求来与服务器交换少量的数据。在ASP.NET中集成Ajax功能可提供一种生成WCF Web服务的简单方法，通过使用浏览器中的客户端JavaScript可以访问这些服务。

• ClickOnce清单

新增了一些密码类，用于验证和获取有关ClickOnce应用程序的清单签名的信息。

在这里仅列举了.NET 3.5中的重要新功能和新特性，但不是全部。读者如果需要了解更多，可到网站http://www.microsoft.com上查找。

1.1.3 公共语言运行时（CLR）

.NET Framework提供一个称为公共语言运行时（Common Language Runtime）的运行环境，用于运行代码并提供使开发过程更轻松的服务。作为.NET Framework的核心组件，它是执行时管理代码的代理，提供内存管理、线程管理和远程处理等核心服务。如图1-2所示为公共语言运行时环境中的各个部分及其提供的重要服务。

公共语言运行时通过公共类型系统（Common Type System，CTS）和公共语言规范（Common Language Specification，CLS）定义了标准数据类型和语言间互操作性的规则。Just-

In-Time编辑器在运行应用程序之前把中间语言（Intermediate Language，IL）代码转换为可执行代码。CLR还管理应用程序，在应用程序运行时为其分配内存和解除分配内存。

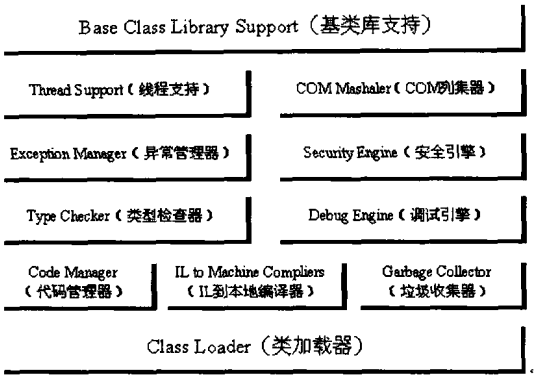

图1-2 公共语言运行时环境

1.1.4 .NET Framework类库

.NET Framework类型是生成.NET应用程序、组件和控件的基础。.NET Framework包括的类型可执行下列功能：

- 表示基础数据类型和异常。
- 封装数据结构。
- 执行I/O。
- 访问关于加载类型的信息。
- 调用.NET Framework安全检查。
- 提供数据访问、多客户端GUI和服务器控制的客户端GUI。

.NET Framework类库是一个由.NET Framework SDK中包含的类、接口和值类型组成的库，提供了对系统功能的访问，是建立.NET Framework应用程序、组件和控件的基础。.NET Framework类库中还包含了.NET Framework中定义的所有类型，如图1-3所示为.NET Framework类库与.NET Framework之间的关系，以及一些.NET Framework类库中的成员。

类通过继承从其他类创建。通过继承，一个类可以访问另一个类定义的方法和属性。另外，除了继承一个类的属性和方法之外，还可以修改已有方法的动作或者属性的行为，这称为重写（Overriding）。

.NET Framework中的所有类和用户创建的类都组织成层次结构，.NET Framework层次结构的基类为System.Object，也就是说System.Object类位于层次结构的顶端，称为超类（Super Class），它提供.NET Framework中所有类的基本功能。图1-4演示了.NET Framework类库中定义的一些类之间的关系。

如图1-4所示，Int16、Int32、Int64、Single、Double结构都派生自System.ValueType，System.Enum类也派生自System.ValueType，而System.ValueType派生自System.Object。Array和String类都直接派生自System.Object。这说明不管是什么类都位于.NET Framework类层次结构的某个位置，最终都派生自基类System.Object，这也是理解.NET Framework类库的关键。

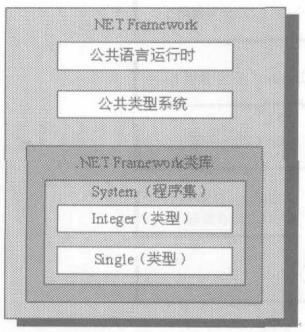

图1-3 .NET Framework类库与.NET Framework的关系

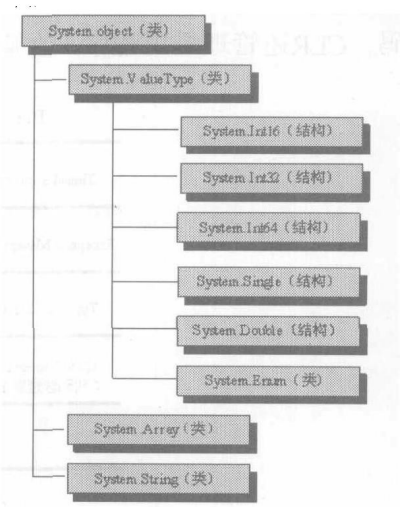

图1-4 .NET Framework类层次结构

1.1.5 程序集

程序集是.NET Framework的生成块，它们构成了部署、版本控制、重复使用、激活范围控制和安全权限等基本单元。程序集可以是静态的或动态的。静态程序集可以包括.NET Framework类型（接口和类），以及该程序集的资源（位图、JPEG文件、资源文件等），存储在磁盘上的可移植的可执行文件中。还可以使用.NET Framework来创建动态程序集，直接从内存运行并且在执行前不存储到磁盘上。但可以在执行动态程序集后将它们保存在磁盘上。

.NET Framework类库由许多程序集组成，用于读取和写入文件，从数据库保存和检索信息，及提供窗体的功能。如表1-1所示为.NET Framework类库的部分程序集及其作用。

表1-1 常见程序集

名称	说明
System.dll	该程序集用于定义主要数据类型，如Integer和Long，另外还定义了最基本的类，如System.Object
System.Windows.Forms.dll	该程序集包含用来实现桌面应用程序使用的窗体的组件，以及创建这些窗体的控件
System.XML.dll	该程序集包含处理XML文档所必需的组件，XML是ADO.NET与Internet相关服务的主要的传输协议
System.Drawing.dll	该程序集包含用于向输出设置（如屏幕、打印机等）绘制各种图形（直线、圆形等）的组件
System.Data.dll	该程序集用于定义组成ADO.NET的组件，而ADO.NET提供了Visual Studio.NET使用的数据库处理服务

1. 程序集内容

通常程序集的内容是由4个元素组成的：程序集清单，包括程序集元数据；类型元数据；

实现这些类型的MSIL代码；程序资源。其中只有程序集清单是必需的，但也需要类型或资源来向程序集提供有意义的功能。程序集中的这些元素有两种分组方法。开发人员可以将所有元素分组到单个物理文件中，如图1-5所示为单文件程序集的结构。

在图1-6所示的多文件程序集结构中，假设应用程序的开发人员已选择将一些实用工具代码单独放入另一个模块中，同时在其源文件中保留一个较大的资源文件（在此例中为一个.bmp图像）。.NET Framework只在文件被引用时下载该文件，通过将很少引用的代码保留在独立于应用程序的文件中来优化代码下载。

图1-5 单文件程序集

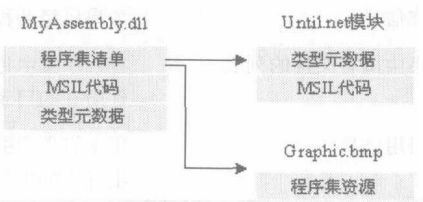

图1-6 多文件程序集

在图1-6中，所有的文件均属于一个程序集，如MyAssembly.dll所包含的程序集清单文件中所述。对于该文件系统，其中的3个文件是3个独立的文件。但是要注意，文件Util.net被编译为一个模块，因为它不包含任何程序集信息。而且在创建了程序集后，程序集清单就被添加到MyAssembly.dll，指示程序集与Util.net模块和Graphic.bmp的关系。

2. 程序集清单

每一程序集，无论是静态的还是动态的，均包含描述该程序集中各元素彼此如何关联的数据集合。程序集清单就包含这些程序集元数据。程序集清单包含指定该程序集的版本要求和安全标识所需的所有元数据，以及定义该程序集的范围和解析对资源和类的引用所需的全部元数据。程序集清单可以存储在具有MSIL代码的PE文件（.exe或.dll）中，也可存储在只包含程序集清单信息的独立PE文件中，如图1-7所示。

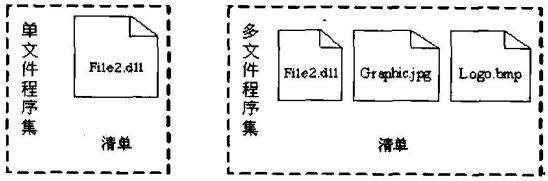

图1-7 单和多文件程序集清单

对于有一个关联文件的程序集，程序集清单将被合并到PE文件中以构成单文件程序集。也可以创建有独立的清单文件，或清单被合并到同一多文件程序集中某一PE文件的多文件程序集。

表1-2显示了在程序集清单中包含的信息，其中前4项（程序集名称、版本号、区域性和强名称信息）构成了程序集的标识。如果依赖的程序集具有强名称，则每一引用均包括该依赖程序集的名称、程序集元数据（版本号、区域性等）和公钥。

表1-2 程序集清单信息

信息	说明
程序集名称	指定程序集名称的文本字符串
版本号	主版本号和次版本号，以及修订号和内部版本号。公共语言运行时使用这些编号来强制实施版本策略
区域性	有关该程序集支持的区域性或语言的信息。此信息只应用于将一个程序集指定为包含特定区域性或特定语言信息的附属程序集（具有区域性信息的程序集被自动假定为附属程序集）
强名称信息	如果已经为程序集提供了一个强名称，则为来自发行者的公钥
程序集中所有文件的列表	在程序集中包含的每一文件的散列及文件名。请注意，构成程序集的所有文件所在的目录必须是包含该程序集清单的文件所在的目录
类型引用信息	用来将类型引用映射到包含其声明和实现的文件的信息。该信息用于从程序集导出的类型

1.1.6 命名空间

命名空间提供了一个组织相关类和其他类型的方式。与文件或者组件不同，命名空间是一种逻辑组合，而不是物理组合。不在同一个文件中的多个类可以共同包含在一个命名空间中，这样就创建了一个逻辑结构。

程序集是.NET Framework的构造块，一个物理程序集可以包含一个或者多个命名空间，例如System和System.IO命名空间都保存在System.dll程序集中。而一个命名空间也可能保存在两个程序集中。一个命名空间又可以包含一个或多个类型（Type）。在.NET Framework中，Integer或者String被认为是类型，每一个类也被视为类型。例如，System命名空间包含一个名为Int16的类型，它代表一个值。

1. *命名空间组成*

命名空间还包含类、委托、结构、枚举和接口，它们都是类型。这些类型既有值类型又有引用类型，如下所示为这些类型的简单介绍。

• 类

类是引用类型，表示具有某个类类型的变量保存了内存地址。该内存地址指向分配给实际对象的内存。像工具箱中的控件都是类，窗体也是类。

• 委托

委托也是引用类型，委托提供了一种机制让程序向Windows操作系统注册事件，以便Windows在运行时调用这些事件，进而调用合适的事件处理程序。委托（事件处理程序）也可以在运行时动态创建。

• 结构

结构是值类型，表示数据直接保存在变量中。结构可以拥有属性和方法。主要的数据类型，如Int（Int16）或者Single在.NET Framework中都是结构。

• 枚举

枚举是一种受限形式的值类型，枚举没有属性和方法，而有域。枚举提供了一种机制把帮助记忆的名称与一个常量值相关联。因此，从概念上讲，枚举类似于常量。许多控件属性都是枚举值。