郝军启　方　宁　朱俊成　编著

Visual Basic 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 Basic 2008
从入门到精通

郝军启　方　宁　朱俊成　编著

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

内 容 简 介

众所周知，Visual Basic是基本、简单、易学的可视编程语言，Visual Basic的优势在于其易用性，使用它可以快捷地编写Windows操作系统下的各种应用程序（像窗体、MDI和Web等）。而Visual Basic 2008是Microsoft公司推出的Visual Basic的最新版本。

本书注重从初学者的认识规律出发，介绍了从入门了解到深入掌握Visual Basic 2008所需的各个方面的知识，包括开发环境的配置、Visual Basic语法、使用面向对象特性、处理字符串和正则表达式、构建Windows和Web应用程序、使用对话框和数据库，以及部署应用程序等。

图书在版编目（CIP）数据

Visual Basic 2008从入门到精通/郝军启，方宁，朱俊成编著.—北京：电子工业出版社，2009.5
ISBN 978-7-121-08548-2

Ⅰ. V… Ⅱ. ①郝…②方…③朱… Ⅲ. BASIC语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2009）第041656号

责任编辑：李红玉 李 荣
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：26.25 字数：670千字
印 次：2009年5月第1次印刷
定 价：47.00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 Basic是世界上使用最广泛的编程语言。随着.NET应用的增多，VB的重要性也日益突出。Visual Basic.NET 2008是Microsoft公司推出的Visual Basic的最新版本。众所周知，Visual Basic是基本、简单、易学的Basic语言，Visual Basic的优势在于其易用性，使用它可以快捷地编写Windows操作系统下的应用程序。

随着Microsoft公司的.NET Framework的推出，Visual Basic也过渡到了Visual Basic.NET，Visual Basic.NET与先前的Visual Basic相比较，具有较大的变化，其中最重要的变化是Visual Basic.NET也实现了完全面向对象，使得人们通过Visual Basic.NET不仅可以开发非常复杂的用户界面，而且可以使用面向对象技术对业务系统建模，这就大大方便了应用系统的开发。

本书将全面介绍Visual Basic.NET编程的有关知识，从该语言本身，到Windows编程、数据访问和Web编程。另外，针对Visual Studio 2008的一些重要的新特性，也会做相应的介绍。全书分5篇，共21章。

第一篇为Visual Basic 2008语言基础。本篇首先向读者介绍了什么是.NET Framework及.NET Framework的重要组成部分，像公共语言运行时、类库、命名空间和程序集。接下来对如何配置Visual Studio 2008的开发环境进行了介绍，还创建了第一个示例程序Hello World。最后，针对Visual Basic 2008的语法进行了详细介绍，包括变量、运算符、控制语句，以及一些面向对象的概念，像什么是对象、类、继承、构造函数、成员、事件，以及如何在Visual Basic中实现等。

第二篇继续介绍Visual Basic 2008语法，本篇讨论了更多特性，像定义和实现接口、使用委托和多态、操作字符串和正则表达式、使用集合、捕捉错误及异常的处理，等等。

第三篇是本书的重点之一，主要介绍了如何进行Windows编程，包括：使用内置控件构建Windows窗体应用程序，在MDI应用程序中管理子窗体、使用菜单栏、工具栏和状态栏，使用Windows的文本弹出、文件、字体、颜色和打印对话框，还对创建自定义控件的知识进行了简单介绍。

第四篇为Visual Basic 2008的高级应用篇。主要介绍了如何使用前面介绍的知识来处理各种应用，像读写文件和注册表、操作XML数据，以及使用GDI+绘制图形，还包括对数据库操作的ADO.NET命名空间、DataSet类、执行存储过程、显示数据的DataGridView控件和LINQ。最后，介绍了如何使用ClickOnce和Windows Installer部署创建的这些程序。

本书最后的第五篇向读者介绍了如何使用Visual Basic 2008构建ASP.NET Web应用程序。首先向读者阐述了ASP.NET的概念，然后讲解如何使用Web控件、管理状态、验证身份，以及在Web页面中访问数据库。还对如何创建和调用Web服务、部署Web应用程序有所介绍。

参加本书编写与制作的作者除封面署名者以外，还有董志鹏、祝红涛、赵俊昌、田广增、刘新龙、肖新峰、郝安林、唐晓阳、郑东方、董红伟、李玺、刘小丹、王龙才、王红敏、韩林林、岳培元、张小素等人。在此，编者对他们表示衷心的感谢。由于编写时间仓促，作者水平有限，书中难免会有错误和疏漏之处，请广大读者给予批评和指正。

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

目　录

第一篇　Visual Basic 2008语言基础

第二篇 编 程 基 础

第三篇 Windows编程

第四篇 高 级 应 用

第五篇 构建Web应用程序

第一篇　Visual Basic 2008语言基础

第1章　.NET Framework

Microsoft公司发布的.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，它提供了托管执行环境、简化的开发和部署，以及与各种编程语言的集成。简而言之，如果想要开发和运行.NET应用程序，就必须首先安装.NET Framework。.NET Framework基至包含了编译器，使开发人员不必使用Visual Studio.NET就可以创建应用程序。

本章以最新的.NET Framework 3.5版本为例向读者介绍.NET Framework及其重要组成部分，以及.NET Framework 3.5的新增功能。

- 了解.NET Framework
- 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
- 命名空间

1.1　.NET Framework概述

现在的计算机编程语言的执行方式分为两种，一种是编译执行，一种是解释执行。编译执行是指源程序代码先由编译器编译成可执行的机器码，然后再执行；解释执行是指源代码程序被解释器直接读取执行。

上面这些都是比较传统的程序代码执行方式，从Java语言开始，一种新的程序语言执行方式产生了，这就是“中间码+虚拟机”执行机制。在这种执行方式中，程序语言源代码需要被编译成一种特殊的中间码，这种中间码是不能直接执行的，它需要一个叫“虚拟机”的装置来管理和执行，可以是解释执行，也可以是编译执行。由于“虚拟机”可以参与和管理程序代码的执行，因此解决了很多传统编译语言一些致命的缺点，如垃圾内存回收、安全性检查等。也正因为如此，.NET框架也采用了此种语言执行方式，在.NET框架中，.NET Framework类似于

管理和执行中间码的“虚拟机”。

在整个执行过程中，首先开发人员在Visual Studio.NET开发环境中编写C#或Visual Basic等源代码，然后这些源代码被Visual Studio.NET中内置的编译器编译成中间语言代码，最后中间语言代码由操作系统中的.NET Framework执行。

需要注意，.NET Framework和Java里面的虚拟机JVM是不同的，Java的虚拟机一般是解释执行的，而.NET Framework是编译执行。另外，.NET Framework作为一个架构，它覆盖了在操作系统上开发软件的所有方面，为集成Microsoft或任意平台上的显示技术、组件技术和数据技术提供了最大的支持。最重要的是，.NET Framework创建出来的应用程序系统可以使Internet程序的开发就像桌面程序的开发一样简单。

实际上，.NET Framework封装了操作系统，将使用.NET开发的应用程序与操作系统特性隔离开来，例如文件处理和内存分配。这样，为.NET开发的应用程序就可以移植到许多不同的硬件和操作系统上。

.NET Framework框架是一个多语言组件开发和执行环境，使开发人员更容易建立网络应用程序和网络服务。.NET Framework框架主要包含三个主要部分：

- 公共语言运行时（Common Language Runtime，CLR）。
- .NET Framework类库集合。
- 用户和程序界面。

.NET Framework的主要组件如图1-1所示。

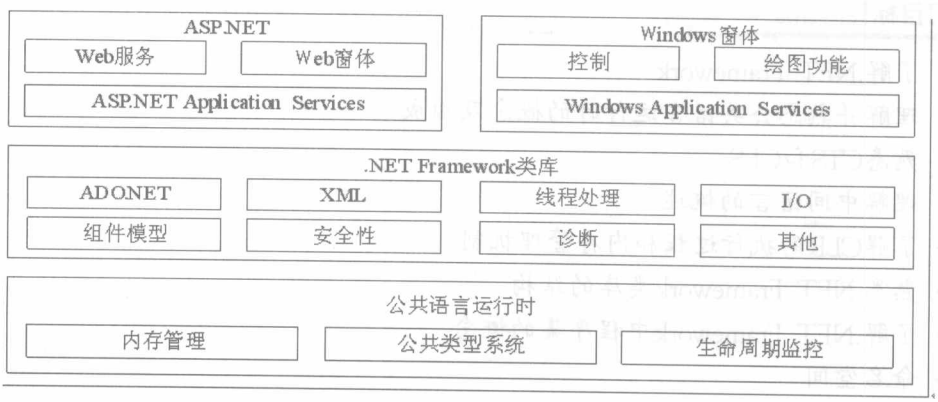

图1-1 .NET Framework的主要组件

在.NET Framework框架的底层是公共语言运行时CLR。这是.NET Framework的核心，是驱动关键功能的引擎。它包括数据类型的公共系统等，这些公共类型和标准接口约定使跨语言继承成为可能。除了内存的分配和管理之外，CLR还负责对象的跟踪，处理垃圾回收。

中间层是.NET Framework类库集合，例如管理数据和XML的类。这些服务在架构的控制之下，可以在各处通用，而且在各种语言中的用法也一致。

顶层为用户和程序界面。Windows窗体为实现客户端的智能程序提供了一种更高级的新方式；Web窗体提供了基于Web的新用户界面。最具代表性的是Web服务，它和Web窗体组成了.NET的Internet接口部分，.NET Framework实现的一部分为ASP.NET。ASP.NET是使用.NET框架提供的类库构建而成的，它提供了Web应用程序模型，该模型由一组控件和一个基本结构组成。使用ASP.NET，开发人员可以直接使用ASP.NET控件集构建Web应用程序。实际上，

ASP.NET控件是运行在Web服务器上的，它们将用户界面转换成HTML格式后再送给浏览器。

需要注意，VS 2008.NET中使用的.NET Framework版本为3.5。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，都可以独立存在于计算机上。而.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。

.NET Framework 3.5主要包括如下的组件：

- .NET Framework 2.0。
- .NET Framework 2.0 Service Pack 1，它用于更新包含在.NET Framework 2.0中的程序集。
- .NET Framework 3.0，它使用.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，它用于更新在.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，则系统会自动将安装它们。另外，.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。.NET Framework 3.5引入的新技术包括：

- 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Visual Basic和C#的语言语法中，并采用标准且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技术支持。它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，并确保开发人员所做的对程序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包含多个方面的改进功能，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 Framework 3.5中比较重要的新增功能，但这并不是全部。如果需要了解更多，可到网站http://www.microsoft.com上查找。

1.2 公共语言运行时

公共语言运行时CLR是用于.NET应用程序的执行引擎，它也是.NET最重要的特征之一。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还负责管理应用程序，为其分配内存和回收内存。

1.2.1 公共类型系统

著名的计算机科学家N.Wirth提出过这样一个公式：算法+数据结构=程序，他不仅指出了数据结构在计算机科学中的地位，也指出了它与算法的密切联系。算法与数据结构是相辅相成的，是构成程序缺一不可的两个方面。数据结构在.NET Framework平台中的形式就是公共类型系统CTS。

在开发应用程序时，使用一种语言编写的程序与使用另一种语言编写的程序之间很难进行数据交换，这是许多编程语言中存在的一个共同问题。例如，由于不同语言中参数的传递顺序存在着差别，在一个Pascal程序中调用C语言编写的方法时，需要格外小心，在C程序中调用Pascal语言编写的方法时也是如此。另外，C语言中的字符串是一个以ASCII码字符NULL（在程序中通常用“\0”表示）为结尾的字符数组。而在Pascal语言中，字符串实际上是一个字符数组，数组中的第一个字节实际上包含了字符串的长度。这就是为什么很难在不同语言之间进行数据通信的原因。

但是，各编程语言也都有类似的特点。如，支持各种数据类型（如整形、字符串形等）、代码模块化及面向对象。.NET平台利用了不同语言的这个共性，抽象出公共类型系统CTS。公共类型系统构成了.NET框架的公共语言运行时的基础，其中最重要的一个方面就是.NET平台的多语言支持，而运行于.NET平台的每一种语言又为了维护自己的语法特色，便使用别名来代替.NET的基础数据类型，如Visual Basic.NET中表示整数的Integer类型，以及C#中表示整数的Int类型实际上是基础数据类型System.Int32的化名。

公共类型系统不仅定义了所有的数据类型，而且提供了面向对象的模型，以及各种语言需要遵守的标准。CTS可以分为两个大类：值类型和引用类型。同时这两种类型，之间还可以进行强制转换，这种转换被称为Boxing（装箱）和UnBoxing（拆箱）。

从图1-2可以看出公共类型系统的基本结构，CTS的每一种类型都是对象，并继承一个基类——System.Object。

值类型是继承自ValueType类的，值类型的变量直接存储数据，实例是被分配存储在栈中的，并且永远不能为空。通过下面的示例可以看出，两个整形变量是分别存储数据的，如下面的代码，无论哪个变量发生变化都不会影响到另一个变量。

```
[Visual Basic]
Dim IntA As Int16 = 456
Dim IntB As Int16 = IntA
IntA = 789
Console.WriteLine("IntA = {0}", IntA)
```

```
Console.WriteLine("IntB = {0}", IntB)
[C#]
Int16 IntA = 456;
Int16 IntB = IntA;
IntA = 789;
Console.WriteLine("IntA = {0}", IntA);
Console.WriteLine("IntB = {0}", IntB);
```

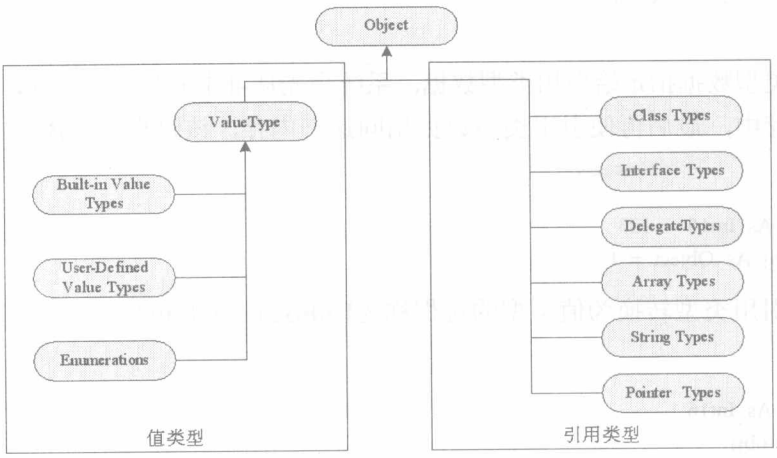

图1-2 公共类型系统CTS的体系

引用类型是继承自Object的，引用类型变量存储的是数据在内存中的地址，而实例是被分配在可以进行垃圾回收的堆中的。一份数据可以被多个变量引用，使用这种变量类型能够起到节省内存资源的作用，同时一个引用变量被修改会导致其他引用发生变更。引用类型的变量是可以为空的，这表示它不指向任何对象。如下面的代码，可以看出objA和objB实际上都指向内存中同一个对象的实例。

```
[Visual Basic]
Sub Main( )
    Dim objA As New test( )
    objA.iValue = 123
    Dim objB As test = objA
    objB.iValue = 321
    Console.WriteLine("objA = {0}", objA.iValue)
    Console.WriteLine("objB = {0}", objB.iValue)
    Console.Read( )
End Sub
Class test
    Public iValue As Int16
End Class
[C#]
class Class1
{
    static void Main(string[] args)
    {
        test objA = new test( );
        objA.iValue = 123;
        test objB = objA;
```

```
            objB.iValue = 321;
            Console.WriteLine("objA = {0}", objA.iValue);
            Console.WriteLine("objB = {0}", objB.iValue);
        }
    }
    class test
    {
        public Int16 iValue;
    }
```

如果将值类型数据指定给引用类型数据，系统会先从堆中分配一块内存，然后将值类型数据复制到该内存中，最后再使引用类型数据指向这一内存，该过程称为Boxing（装箱）。

例如：

```
Dim I As Int16 = 123
Dim obj As Object = I
```

反之，将引用类型转换为值类型的过程称为UnBoxing（拆箱）。

例如：

```
Dim J As Int16
J = Int(obj)
```

在装箱的过程中，内存的配置是很费时的操作，虽然这一切对程序员来说是透明的，但考虑到程序的执行效率，应该尽量避免不必要的装箱操作。

1.2.2 公共语言规范

公共语言规范（Common Language Specification，CLS）是由一组结构和限制规则构成的，用做库编写者和编译器编写者的指南。这样任何支持CLS的语言都可以完全使用库，并且使这些语言之间可以相互集成。公共语言规范是公共类型系统的子集。对于那些需要编写代码供其他开发人员使用的应用程序开发人员而言，公共语言规范也非常重要。如果开发人员遵循CLS规则来设计公共访问的API，那么就可以在支持公共语言运行时的任何其他编程语言中很容易地使用这些API。

1.2.3 中间语言

在传统的编译器中，开发人员编写的高级语言代码将一次性被转换为特定CPU体系结构上执行的本机指令。这是因为不同厂商生产的CPU体系结构都不同，并且各自都具有自己独立的一套指令集。除此之外，即使是同一个厂商，也常常制造出带有特殊指令附加功能的CPU。这就为程序在不同平台之间迁移造成了困难。

因此，为了提高程序的跨平台性，.NET语言的编译分为两个阶段。第一个阶段是将高级语言编译成一种称做IL的中间语言，与高级语言相比，IL更像是机器语言，然而IL却包含一些抽象概念（比如：类、异常），这也是这种语言被称为中间语言的原因。.NET框架中的中间语言称为MSIL，即“微软中间语言”，它是由一组特定的指令组成的，这些指令指明如何执行代码。

由于机器的CPU只能执行本地汇编语言，而不是IL，因此进一步将IL编译成汇编语言的工

作（也就是第二阶段）需要在运行时进行，这个过程由即时编译器（Just In Time，JIT）完成。为了充分利用各种CPU的独特功能，进而提高系统的性能，Microsoft公司为每一种目标CPU版本提供了不同版本的JIT。

中间语言的主要特征如下：

• 面向对象和使用接口

中间语言在设计时就是为了实现某些特殊的编程方法，这表示面向它的语言必须与编程方法兼容，Microsoft为IL选择的是传统的面向对象的编程，具有类的单一继承性。除了传统的面向对象编程外，中间语言还引入了接口的概念，它们是在带有COM的Windows操作系统下实现某种特殊编程方法的的第一种方式。

• 相异的值类型和引用类型

与其他编程语言一样，中间语言提供了许多预定义的基本数据类型。它的一个特性是值类型和引用类型有明显的区别。对于值类型，变量直接保存其数据，而对于引用类型，变量仅保存地址，对应的数据可以在该地址中找到。

• 强数据类型

中间语言的一个重要方面是它基于强数据类型，即所有的变量都必须显式声明为属于某个特定数据类型，中间语言一般不允许对模糊的数据类型执行任何操作。

• 使用异常来处理错误

.NET Framework可以根据异常机制处理错误，这与Java和C++是一样的。C++开发人员应注意到，由于IL具有非常强大的类型系统，所以在IL中以C++的方式使用异常不会带来相关的性能问题。

• 使用特性（Attribute）

特性最初是为了在程序中提供与某些项相关的额外信息，以供编译器使用。.NET提供了对特性的支持，因此现在C++、C#和Visual Basic也支持特性。但在.NET中为特性创建了一个新的机制，通过该机制用户可以在源代码中定义自己的特性。这些用户定义的特性将和对应数据类型或方法的元数据放在一起，这对于文档说明书十分有用。另外，与.NET的语言无关性的基本原理一样，特性也可以在一种语言的源代码中定义，而被用另一种语言编写的代码读取。

1.2.4 执行管理

.NET支持两种程序执行方式，即托管方式和非托管方式。托管是.NET中的一个专门的概念，也是它倡导的一种新的编程理念。托管就是将所有的与操作系统的交换都交由.NET Framework来完成，就像是把这些功能委托给.NET，所以称之为托管。非托管则与此相反。

由于托管代码是在公共语言运行时CLR上执行的代码，而不是直接在操作系统上执行，所以使用托管代码的应用程序可以获得公共语言运行时的各种服务，例如自动垃圾回收、运行库类型检查和安全支持等。这些服务用于提供独立于平台和语言的、统一的托管代码应用程序行为。

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++。托管代码下的所有语言共享统一的类库集合，并能被编码成为中间语言（IL）。编译器在托管执行环境下将中间语言（IL）编译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代码的安全。