第 1 章　Visual Basic.NET 概述

微软公司在 2000 年推出了.NET 战略，它是面向互联网时代构建的新一代平台，是微软公司在 21 世纪初的一个重大战略步骤。为了实现.NET 技术，微软公司开发了一整套基于.NET 框架重新设计的工具组件，它们被集成到 Visual Studio.NET 开发环境中，用于开发新的.NET 应用程序，而 Visual Basic.NET（简称 VB.NET）就是它的一个重要组成部分。目前，它的最新平台为 Visual Studio 2010。

1.1　Visual Basic.NET 和.NET 框架

虽然 Visual Basic.NET 是在 Visual Basic 6.0 的基础上产生的，语法与传统的 Visual Basic 有 90%基本相同或相似，但在其他很多方面，VB.NET 都和 VB 截然不同，以至于它是否该被看作 VB 的新版本抑或是另一个完全不同的语言争议不断。

相比于 VB 6.0，VB.NET 所做出的重大变革如下。

1．真正面向对象

学过 VB 6.0 的读者一定接触过面向对象的思想。不过 VB 6.0 只是**基于对象**的语言，学过之后是不能真理解面向对象的，属性、方法和事件只是面向对象编程的部分概念，真正的面向对象编程需要对象满足和支持下面三个特性：封装性、继承性和多态性。现在开始学习 VB.NET 了，它对 VB 的对象模型进行了最为彻底的改造，其处理的任何事物几乎都和对象有关系，完全支持面向对象的三个特性。

2．基于.NET 框架类库

在使用 Visual Basic 6.0 编写应用程序时，虽然可以从工具箱中选择组件，通过拖入的方式加入应用程序中，但在实现机理上，它们全部是 COM 组件，即可在 COM 应用程序中使用的 ActiveX 控件、ActiveX 类型的 DLL 和 ActiveX 类型的 EXE。而在 VB.NET 中，向窗体拖入的组件都是从.NET 框架提供的 SDK 相应的类继承，经实例化后，通过添加功能才把组件加入到设计窗体中。VB.NET 中使用的每一个组件都是从 Component 或 Control 基类派生出来的。

3．具备诸多现代编程特性

此外，VB.NET 还增加了很多新特性以满足现代软件开发的需要，其中包括：

（1）多线程。Visual Basic 6.0 只提供单线程模式，并不支持多线程。而 VB.NET 则全面支持多线程。

（2）异常处理。VB.NET 增加了结构化异常处理的新功能。

（3）Web 开发。使用 ASP.NET 技术来编写 Web 页，不再需要使用 VBScript 脚本语言。

（4）数据库访问。使用 ADO.NET，可以将任何控件的任何属性绑定到包含数据的任何结构中。

正是这些变革，才使 VB.NET 功能更强大，更具生命力，但同样也为传统 VB 程序员转型为 VB.NET 程序员带来了困难。因为 VB.NET 提出了很多新的功能、概念和观点，掌握 VB.NET 不仅要掌握语法，还需要理解并运用这些新的功能、概念和观点，当然首先要掌握.NET 框架。

运用.NET 框架（.NET Framework）可以建立.NET 应用程序。使用.NET 开发的程序需要在.NET Framework 下才能运行。.NET Framework 体系结构包括以下 5 部分：

（1）程序设计语言及公共语言规范（CLS）

（2）应用程序平台（ASP.NET 及 Windows 应用程序等）

（3）ADO.NET 及类库

（4）公共语言运行时（CLR）

（5）程序开发环境（Visual Studio）

其结构如图 1.1 所示。

构建在 Windows 操作系统之上的是公共语言运行时（CLR），其作用是负责执行程序，提供内存管理、线程管理、安全管理、异常处理、通用类型系统与生命周期监控等核心服务。在 CLR 之上是基类库，提供许多类与接口。

在.NET 框架基础上的应用程序主要包括 ASP.NET 应用程序和 Windows Forms 应用程序，其中 ASP.NET 应用程序又包含了 Web Forms 和 Web Service，它们组成了全新的互联网应用程序；而 Windows Forms 则是传统的窗口应用程序。

在.NET Framework 之上，无论采用哪种语言编写的程序，都先被编译成中间语言 IL，IL 经过再次编译才生成机器码，完成 IL 到机器码编译任务的是 JIT（Just In Time）编译器。上述处理过程如图 1.2 所示。

图 1.1　.NET 框架结构

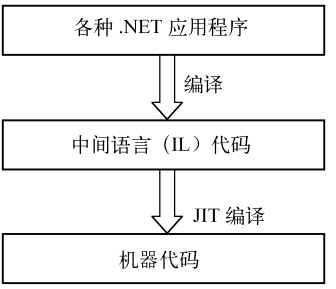


图 1.2　.NET 应用程序的编译过程

随着.NET 技术的发展，.NET Framework 也经历了几个发展阶段，从早期的.NET Framework 1.0、1.1 发展到 2.0、3.0、3.5，功能越来越强。2010 年，随着 Visual Studio 2010 的推出，.NET Framework 又由 3.5 升级为 4。

.NET Framework 4 在.NET Framework 3.5 的基础上进行了扩展，增加了很多新特性（如 WCF，WPF，WF，LINQ 和 AJAX 等），使.NET 技术更加强大和成熟。在利用 Visual Studio

2010 进行项目开发时，可根据所使用的特性选择.NET Framework 版本，实现与早期版本的兼容。

1.2　Visual Basic.NET 2010 集成开发环境

VB.NET 的开发环境与以前的 VB 开发环境有很大区别。以前的 VB、VC 等 IDE 都是各自独立的，而从 VB 6.0 之后，微软公司推出了 Visual Studio .NET 系列，将 VB.NET、VC.NET、VC#.NET 等多种开发语言统一到一个集成开发环境中，给用户一个统一的界面，从而为使用多种语言带来了方便，提高了开发效率。

1.2.1　Visual Studio 2010 简介

Visual Studio 2010（简称 VS 2010）是这个系列的最新版本，于 2010 年 4 月 12 日上市（图 1.3 为本产品 Logo），其IDE界面被重新设计和组织，变得更加简单明了。VS 2010 同时带来了.NET Framework 4.0、CTP（Community Technology Preview），并且支持开发面向Windows 7的应用程序，数据库方面除了支持Microsoft SQL Server外，还支持 IBM DB2和Oracle数据库。

图 1.3　VS 2010 Logo

新版的 Visual Studio 2010 实现了 9 大功能创新：

（1）C# 4.0 中的动态类型和动态编程；

（2）支持多显示器；

（3）支持 TDD；

（4）支持 Office；

（5）Quick Search 特性；

（6）C++ 0x 新特性；

（7）IDE 增强；

（8）使用 Visual C++ 2010 创建 Ribbon 界面；

（9）新增基于.NET 平台的函数型程序设计语言 Visual F#。

上述崭新的功能使它成为当前最流行的Windows应用程序开发工具。难怪有业内人士预言：VS 2010 将是史上又一个经典版本，不亚于当年的 VB 6.0。

1.2.2　Visual Basic.NET 2010 配置和使用

VS 2010 目前总共有 5 个版本：专业版（Professional）、高级版（Premium）、旗舰版（Ultimate）、学习版（Express）和测试版（Test）。其中，旗舰版的功能最强大，本书就用它来开发程序。

1．VS 2010 的安装

从网络下载获得 VS 2010 旗舰版的安装镜像文件 X16-60997VS2010UltimTrialCHS.iso，使用虚拟光驱工具 DAEMON Tools Lite（也可用其他的虚拟光驱软件）载入镜像，此时会自动运行 VS 2010 安装程序，如图 1.4 所示。

图 1.4　用虚拟光驱启动安装程序

单击【安装 Microsoft Visual Studio 2010】，进入安装向导，如图 1.5 所示，安装程序会自动加载所需的组件，然后一个接一个地安装……如图 1.6 所示，这个过程可能会持续一段时间，请用户耐心等待！

图 1.5　安装向导

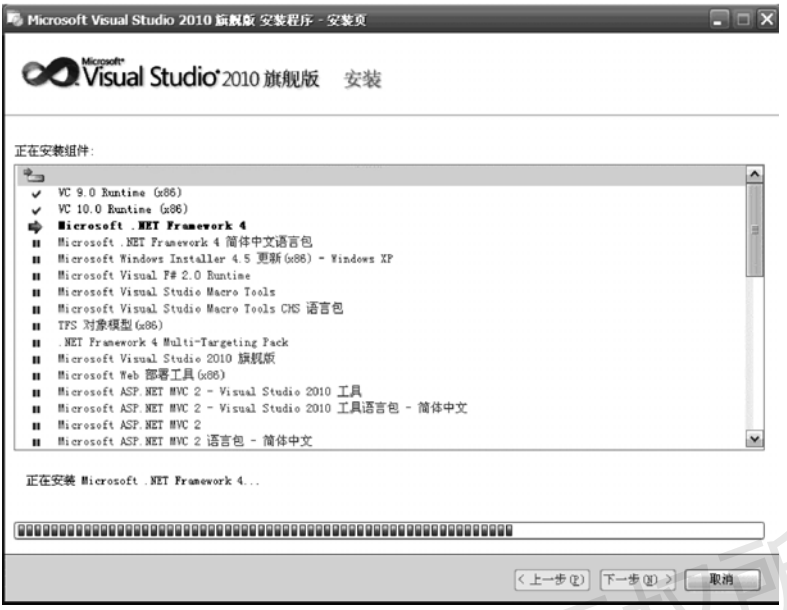


图 1.6　正在安装组件

安装过程结束后，会出现“完成页”，如图 1.7 所示，显示安装成功！

图 1.7　安装成功

单击【完成】按钮，结束安装。

2．VB.NET 环境设置

安装完成后，初次启动 VS 2010 会弹出【选择默认环境设置】对话框，供用户自定义开发环境，如图 1.8 所示，这里选择“Visual Basic 开发设置”编程环境。

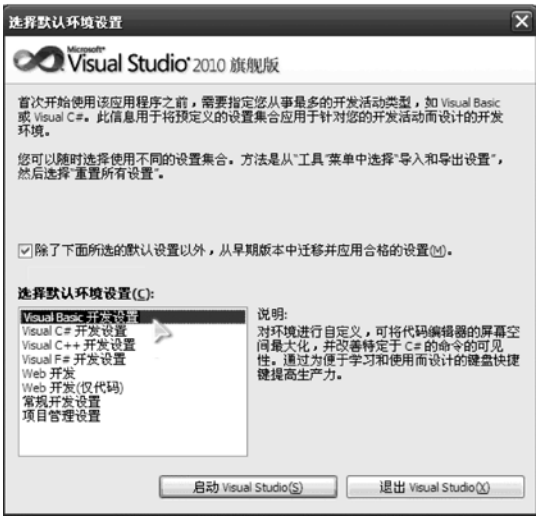


图 1.8　设置开发环境

单击【启动 Visual Studio（S）】按钮，进入 VB.NET 开发环境。

3．使用方法

（1）启动

开机进入 Windows 操作系统后，可用多种方法启动 VB.NET 环境。常用方法是从开始菜单启动。单击【开始】→【所有程序】→【Microsoft Visual Studio 2010】→【Microsoft Visual Studio 2010】，即可启 动 VB.NET 集成开发环境，启动后将显示“起始页”，如图 1.9 所示。

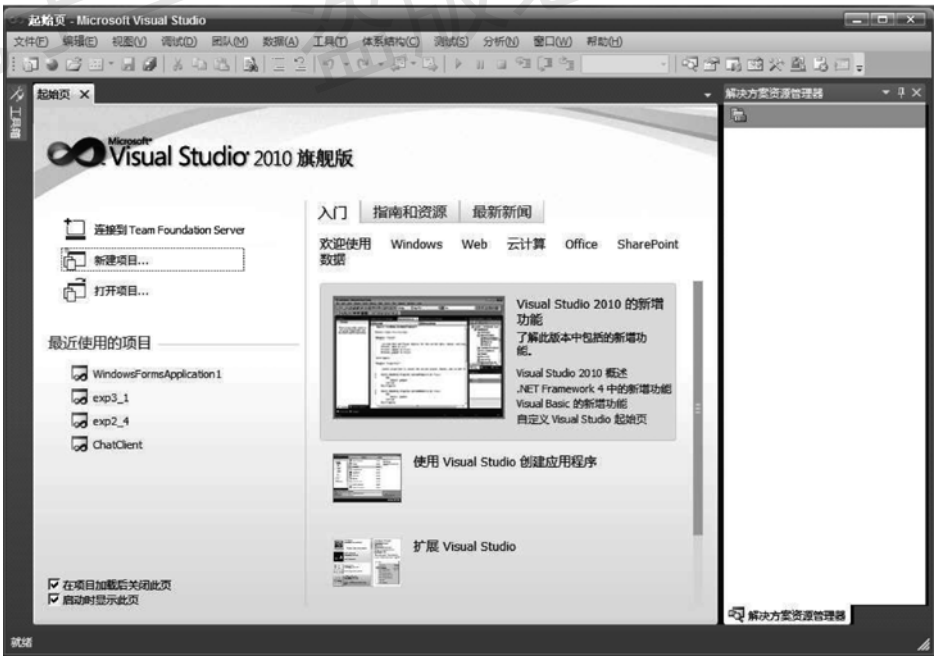


图 1.9　VS.NET 起始页

在“起始页”中允许用户新建或打开项目。

（2）建立项目

建立项目，可单击【新建项目...】项，将弹出【新建项目】对话框，如图 1.10 所示。

图 1.10　【新建项目】对话框

首先，在“已安装的模板”栏中选择“Visual Basic”项；然后在中间的列表中选择要开发的程序类型（此处选“Windows 窗体应用程序”）；然后，在下方“名称”栏中输入项目名称（这里取默认的“WindowsApplication1”）；最后单击【确定】按钮进入 VB.NET 2010 集成开发环境，如图 1.11 所示。

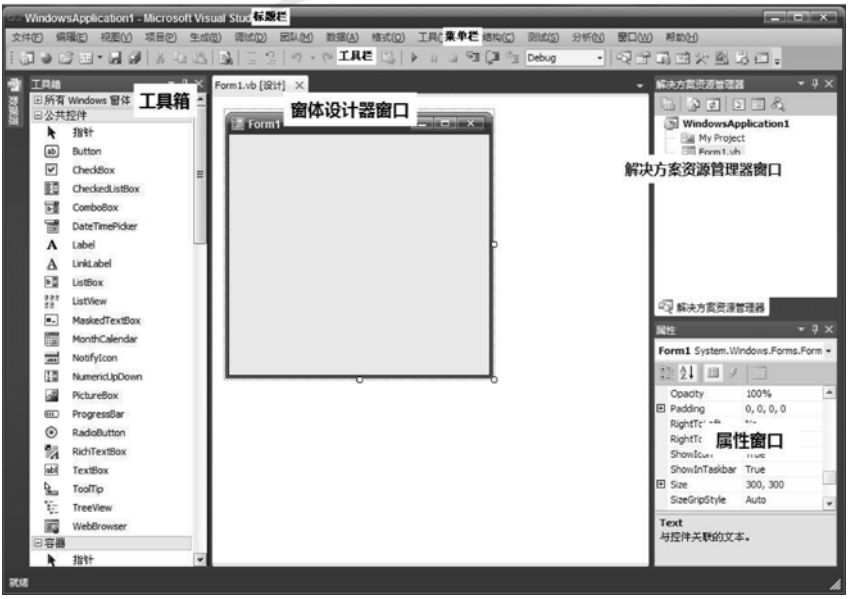


图 1.11　VB.NET 2010 集成开发环境

可以看出，屏幕被分成若干部分，包括标题栏、菜单栏、工具栏、窗体设计器窗口、工具箱、解决方案资源管理器窗口、属性窗口等。这些部分的用途将在本章稍后逐一介绍。

（3）退出

若退出开发环境，可以执行【文件】菜单中的【退出】命令，或者直接关闭窗口。如果当前程序尚未保存，系统会弹出【保存项目】对话框提示存盘，如图 1.12 所示，可在“名称”一栏修改项目名，在“位置”栏输入项目保存的路径；在“解决方案名称”栏输入解决方案的名称（此处这三者都采用默认设置）。

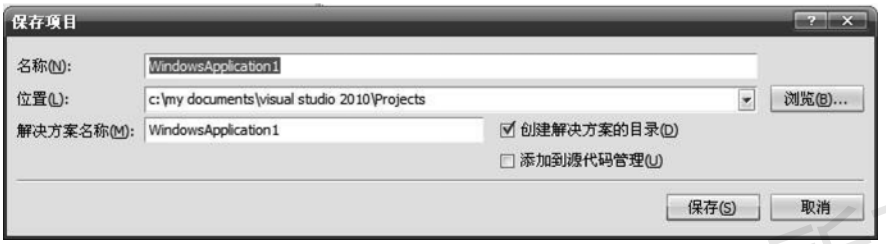


图 1.12　【保存项目】对话框

单击【保存】按钮则存盘后退出；单击【取消】按钮则不保存。

（4）打开项目

要打开已有项目，可在“起始页”中单击“最近使用的项目”列表中的某个项目名；也可以单击“打开项目...”项，在弹出的【打开项目】对话框中选择要打开的项目，如图 1.13 所示。

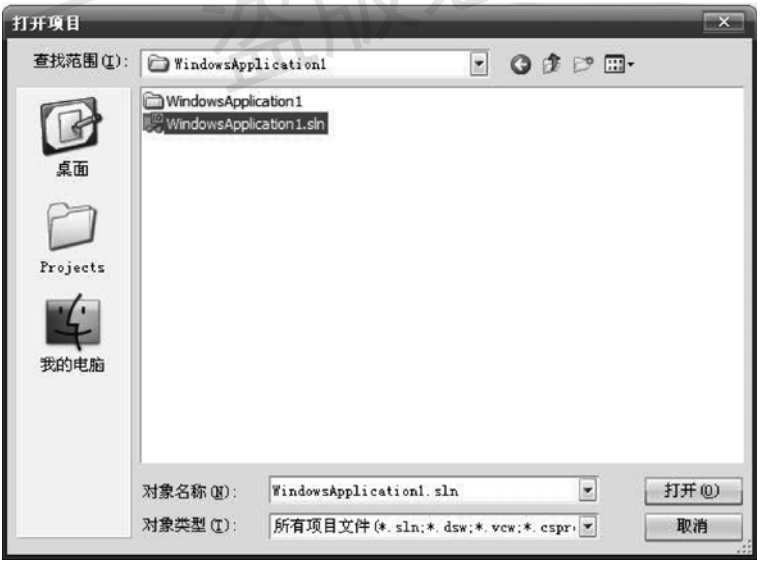


图 1.13　【打开项目】对话框

若对项目进行过更改，在退出环境时，系统会自动提示用户保存更改，如图 1.14 所示。

图 1.14　提示保存更改

1.2.3　开发工具的界面元素

1．标题栏

标题栏是 VS.NET 2010 窗口顶部的水平条，它显示应用程序的名字。默认情况下，用户建立一个新项目后，标题栏显示的是如下信息：

其中，“WindowsApplication1”为解决方案名称。随着工作状态的变化，标题信息也随之改变。当处于调试状态时，标题信息如下：

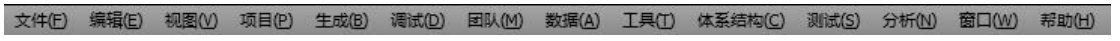
括号中的“正在调试”表明当前的工作状态处于“调试阶段”。当处于运行状态时，该括号中的信息又变为“正在运行”：

2．菜单栏

在标题栏的下面是集成环境的主菜单，开发者要完成的主要功能都是通过菜单实现的。在不同状态下，菜单栏的菜单项个数是不一样的。例如，启动 VB.NET 后，建立项目前（即在“起始页”状态下），菜单栏有 12 个菜单项：

而当建立或打开项目后，如果当前活动的窗口是窗体设计器，则菜单栏有 15 个菜单项：

如果当前活动的窗口是代码窗口，则菜单栏中有 14 个菜单项：



每个主菜单项又包含若干子菜单项，灰色的选项是禁用的；菜单项中显示在菜单名后面“( )”中的字母为键盘访问键，菜单项后面显示的为快捷键。例如，“新建项目”的操作是先按“Alt+F”组合键打开【文件】菜单，再按“N”键，或直接按“Ctrl+Shift+N”组合键，如图 1.15 所示。

图 1.15 【文件】菜单

（1）【文件】菜单（File）

【文件】菜单用于文件操作，如打开和新建项目，打印和保存文件等，对应的主要功能如表 1.1 所示。

表 1.1 【文件】菜单功能表

下拉菜单项	功 能
新建项目	建立新项目
新建网站	建立新网站
新建文件	建立新文件
打开项目	打开已有项目
打开网站	打开已有网站
打开文件	打开已有文件
添加	向当前解决方案添加新项目（网站）或现有项目（网站）等
关闭	关闭当前项
关闭项目	关闭打开的项目
保存 Form1.vb	保存对 Form1.vb 的修改，文件名不变
Form1.vb 另存为	将 Form1.vb 另存为其他文件名
全部保存	保存当前打开的所有项目
导出模板	将项目或项导出为可用作将来项目基础的模板
最近的文件	通过最近打开过的文件来打开相应的项目
最近使用的项目和解决方案	通过最近打开过的解决方案来打开相应的解决方案和项目
退出	退出 VS 2010 集成开发环境

（2）【视图】菜单（View）

【视图】菜单用于显示或隐藏各功能窗口或对话框。若不小心关闭了某个窗口，可以通过选择视图菜单项来显示该窗口。【视图】菜单还控制工具栏的显示，若要显示或关闭某个工具栏，只需单击【视图】→【工具栏】菜单项，如图 1.16 所示，找到相应的工具栏，在其前面打勾或去掉勾即可。

图 1.16　【视图】菜单

【视图】菜单对应的主要功能见表 1.2。

表 1.2　【视图】菜单功能表

下拉菜单项	功　能
解决方案资源管理器	打开解决方案资源管理器窗口
服务器资源管理器	打开服务器资源管理器窗口
对象浏览器	打开对象浏览器窗口
起始页	打开 VS 2010 起始页面
工具箱	打开工具箱
其他窗口	打开命令窗口、Web 浏览器、类视图、输出、任务列表等其他窗口
工具栏	打开或关闭各种快捷工具栏
属性窗口	打开用户控件的属性页

（3）【项目】菜单（Project）

【项目】菜单主要用于向程序中添加或移除各种元素，如窗体、控件、组件、模块、类等，如图 1.17 所示。

图 1.17　【项目】菜单

【项目】菜单使用简单，这里特别提醒如何添加 Web 引用功能。Web 引用指向一个 Web 服务，通过添加它，可以查找联机的 Web 服务，并在程序中使用它们。具体操作如下：选择【项目】→【添加服务引用】，弹出【添加服务引用】对话框，如图 1.18 所示。

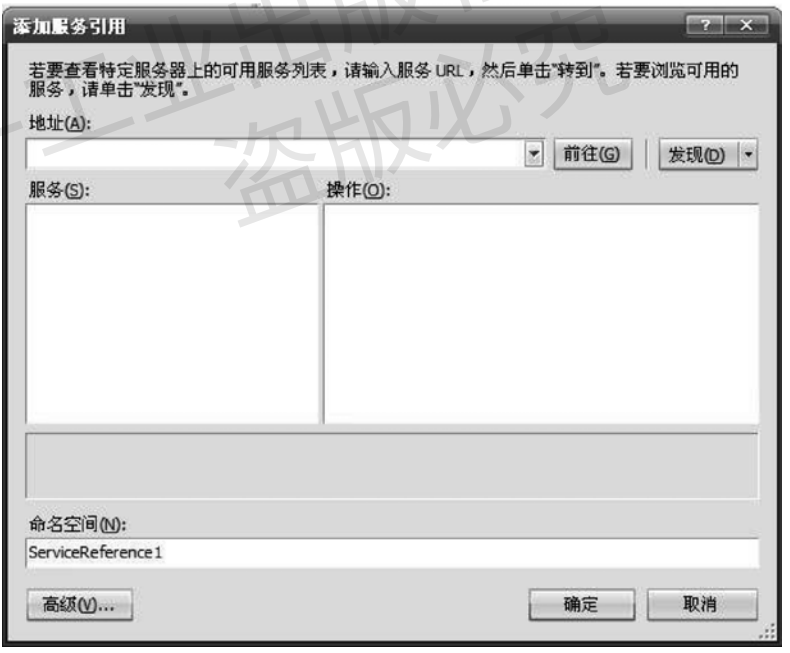


图 1.18　【添加服务引用】对话框

单击左下方【高级】按钮，弹出【服务引用设置】对话框，如图 1.19 所示。

单击左下方【添加 Web 引用】按钮，弹出【添加 Web 引用】对话框，如图 1.20 所示，即可添加 Web 引用。

图 1.19　【服务引用设置】对话框

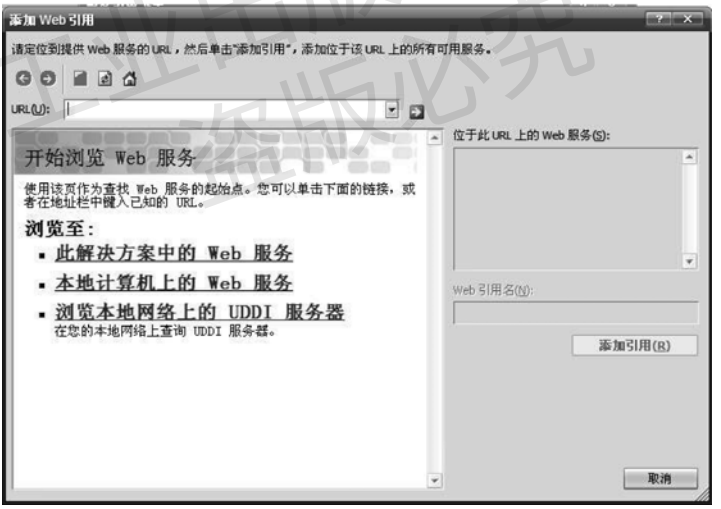


图 1.20　【添加 Web 引用】对话框

（4）【生成】菜单（Build）

【生成】菜单主要用于生成能运行的可执行程序文件，如图 1.21 所示，生成之后的程序可以脱离 VB.NET 环境独立运行，也可以用于发布程序。

（5）【调试】菜单（Debug）

【调试】菜单用于选择不同的调试程序方法，如监视窗口、逐语句、逐过程、设断点等，如图 1.22 所示。

图 1.21 【生成】菜单

图 1.22 【调试】菜单

对应的主要功能见表 1.3。

（6）【格式】菜单（Format）

【格式】菜单用于设计阶段窗体上各个控件的布局。利用它可以对所选定的对象调整格式，在设计多个对象时用来使界面整齐而进行统一操作。格式菜单如图 1.23 所示，主要功能见表 1.4。

表 1.3 【调试】菜单功能表

下拉菜单项	功　能
启动调试	以调试模式运行
逐语句	一句一句运行
逐过程	一个过程一个过程运行
新建断点	用于设置断点调试
删除所有断点	清除所有已设置的断点

图 1.23 【格式】菜单

表 1.4 【格式】菜单功能表

下拉菜单项	功　能
对齐	所有选中的对象对齐
使大小相同	所有选中的对象按宽或高统一尺寸
水平间距	对所有选中的对象水平间距统一调整
垂直间距	对所有选中的对象垂直间距统一调整
在窗体中居中	对象在窗体中居中对齐
顺序	对象按前、后顺序放置
锁定控件	使所选中的控件锁定，不能调整位置

（7）【工具】菜单（Tools）

【工具】菜单用于在设计工程时选择一些工具。例如，可用来连接到数据库、连接到服务器、添加/删除工具箱等。工具菜单如图 1.24 所示。

图 1.24　【工具】菜单

（8）其他菜单

菜单栏中还有【编辑】、【窗口】和【帮助】菜单，它们的功能与一般标准 Windows 程序的主菜单基本相同。另外，诸如【团队】、【数据】、【体系结构】、【测试】和【分析】等菜单的功能，对于初学者来说基本用不上，在此就不详细介绍了。

除了菜单栏中的菜单外，若在不同的窗口中单击鼠标右键，还可以得到相应的专用快捷菜单，也称上下文菜单或弹出菜单，其功能在后面学习中用到时再作介绍。

3．工具栏

工具栏在编程环境下提供对常用命令的快速访问。单击工具栏按钮，则执行该按钮所代表的操作。VB.NET 提供多种工具栏，用户可根据需要定义自己的工具栏。默认情况下，VB.NET 只显示标准工具栏，其他工具栏可以通过【视图】→【工具栏】命令打开或关闭。每种工具栏都有固定和浮动两种形式，把光标移到固定工具栏中没有图标的地方，按住左键向下拖动鼠标，即可把工具栏变为浮动的；如果双击浮动工具栏的标题，则又可变回固定工具栏。

默认工具栏如图 1.25 所示，这是启动 VB.NET 之后显示的标准工具栏，当鼠标停留在工具栏按钮上时可显示该按钮的功能提示。

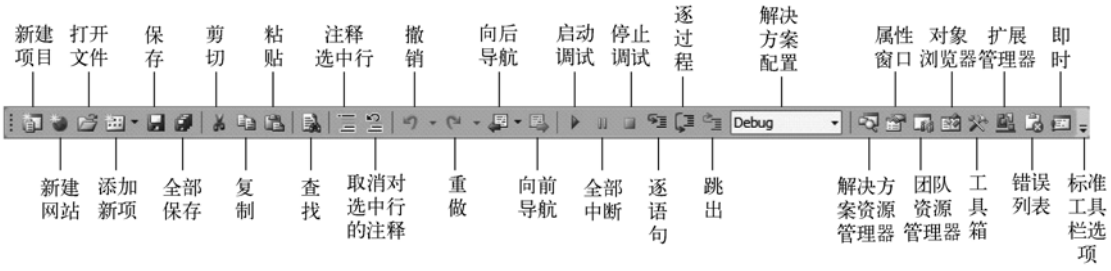


图 1.25　工具栏及其按钮

每一个工具栏按钮都对应菜单栏主菜单下的某个菜单项，比如工具栏前部第一个【新建

项目】按钮就相当于【文件】→【新建项目】菜单项的功能。

4．工具箱

工具箱（Toolbox）也称控件箱，它提供一组控件，用户在设计界面时可以从中选择所需的控件放入窗体中。工具箱位于屏幕的左侧，默认情况下是自动隐藏的，当鼠标接近工具箱敏感区域时，它会自动弹出，如图 1.26 所示，而鼠标一离开又会自动隐藏。

从图 1.26 可见，工具箱是由众多控件组成的，为便于管理，VB.NET 将常用的控件分别放在“所有 Windows 窗体”、“公共控件”、“容器”、“菜单和工具栏”、“数据”、“组件”、“打印”、“对话框”、“WPF 互操作性”、“报表”、“Visual Basic PowerPacks”、“常规”等 12 个选项卡中，如图 1.27 所示。

图 1.26　工具箱

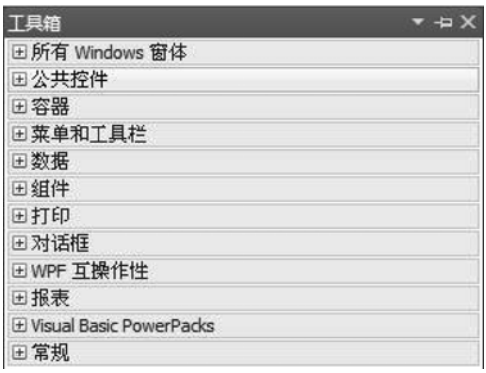


图 1.27　工具箱选项卡

这 12 个选项卡中存放的内容如表 1.5 所示。

表 1.5　工具箱选项卡内容

选项卡名称	内 容 说 明
所有 Windows 窗体	存放 Windows 程序界面设计所有的控件
公共控件	存放常用的控件
容器	存放容器类的控件
菜单和工具栏	存放菜单和工具栏的控件
数据	存放操作数据库的控件
组件	存放系统提供的组件
打印	存放打印相关的控件
对话框	存放各种对话框控件
WPF 互操作性	ElementHost 类可在 WinForm 界面上添加 WPF 控件
报表	ReportViewer 报表控件
Visual Basic PowerPacks	基本的可视化数据显示和图形控件
常规	保存了用户常用的控件，包括自定义控件

选项卡中的控件不是一成不变的，可根据需要增加或删除。在工具箱窗口中单击鼠标右键，从弹出菜单中选择【选择项】，会弹出一个包含所有可选控件的【选择工具箱项】对话框，如图 1.28 所示。

图 1.28　【选择工具箱项】对话框

通过勾选或取消勾选其中的各个控件，即可添加或删除选项卡中的控件。

1.2.4　常用工作窗口

在图 1.11 的 VB.NET 集成开发环境中显示了几个窗口，包括“解决方案资源管理器窗口”、“属性窗口”和“窗体设计器窗口”。工作中需要显示的窗口可由用户通过【视图】菜单来设置。

1．窗体设计器窗口

窗体设计器窗口简称窗体（Form），是用户自定义的窗口，用来设计应用程序的界面，它对应的是程序运行的最终结果。各种图形、图像、数据等都是通过窗体或其中的控件显示出来的。

窗体设计器窗口如图 1.29 所示。

在程序窗体的左上角是窗体的标题，如图 1.29 中的“Form1”。建立一个新项目后，系统将自动建立一个窗体，其默认名称和标题均为 Form1，窗体表面空白部分称为工作区或操作区，设计器窗口的标题是“Form1.vb [设计]”。

在设计应用程序时，用户根据需要，从工具箱中选择所需要的工具（控件），然后在工作区中画出相应的控件对象，这样就完成了程序的界面设计。

2．解决方案资源管理器窗口

解决方案资源管理器窗口位于窗体设计器的右边，它用来列出当前解决方案中所有项目，如图 1.30 所示。

图 1.29　窗体设计器窗口

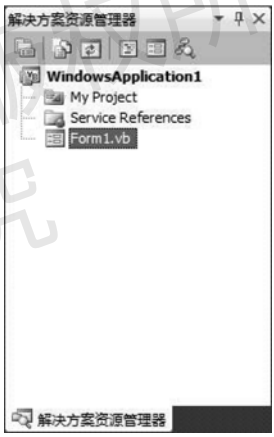


图 1.30　解决方案资源管理器窗口

“解决方案”相当于以前 VB 中的“工程组”，不同的是，“工程组”只能含 Visual Basic 项目，而“解决方案”可以包含不同语言的项目。

利用该窗口可以方便地组织待开发的项目、文件，配置应用程序或组件。在该窗口中，以树形结构显示解决方案及其项目的层次，可以方便地打开、修改、管理其中的对象，这些对象都是以文件形式保存在磁盘中的，其中 Form1.vb 是窗体模块的文件名。

新建项目都放在设定的解决方案中，一个解决方案可以含有一个或多个项目。默认情况下，解决方案的名字与项目名相同，而且存放项目和解决方案的文件夹名就是项目名。

3．属性窗口

属性窗口位于解决方案资源管理器的下方，用于列出当前选定窗体和控件的属性设置。属性即对象的特征。如图 1.31 所示是名称为“Form1”的窗体对象的属性。

属性显示方式可以有两种，图 1.31 是“按分类顺序”排列各个属性，而图 1.32 是“按字母顺序”排列各个属性，在属性窗口的上部有一个工具栏（见图 1.32），用户可以单击相应

的工具按钮来切换显示方式。

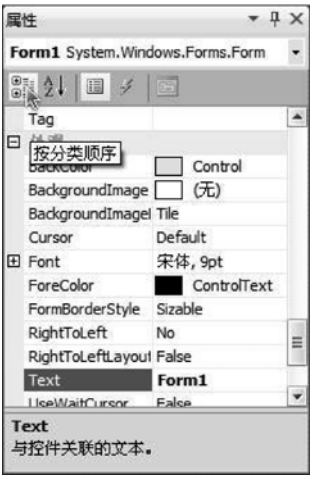


图 1.31　属性窗口（按分类顺序）

图 1.32　属性窗口（按字母顺序）

类和名称空间位于属性窗口的顶部，其下拉列表中的内容为应用程序中每个类的名字及其所在的名称空间。“属性列表”是一组属性名对应的设置值，“属性说明”用于说明该属性的用途。启动 VB.NET 后，类和名称空间里起初只有窗体信息。随着窗体中控件的增加，将把这些对象的有关信息加入到名称空间框的下拉列表中。

1.3　一个简单的 Visual Basic.NET 程序

VB.NET 继承了 VB 6.0 的易用性，初学者甚至可以一句代码都不写就创建并运行一个“Hello World!”程序。

【例 1.1】 在窗体上输出“Hello World!”。

Label 控件专门用来显示 GUI 程序界面上的说明性文字。打开之前创建的项目 WindowsApplication1，从工具箱拖一个 Label 控件到窗体设计器工作区中，如图 1.33 所示。

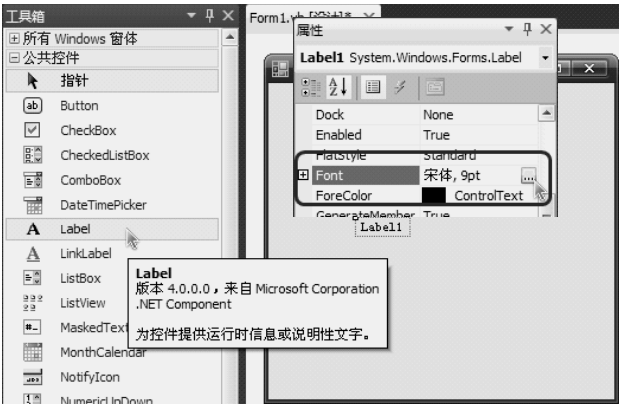


图 1.33　拖动 Label 控件到窗体

在 Label 的属性窗口中单击 Font 属性行右边的...，弹出【字体】对话框，如图 1.34 所示。

设置想要显示的字体为“Arial Black”，字形为“粗斜体”，大小为“小二号”字，勾选加“下划线”。将 Label 的 Text 属性设置为“Hello World!”。

设置完毕，单击工具栏▶按钮运行程序，结果如图 1.35 所示。

图 1.34　设置标签字体

图 1.35　显示“Hello World!”程序结果

可见，VB.NET 编程的门槛比任何其他语言都低得多，这尤其适合程序设计初学者。

习　题　1

1．VB.NET 相比 VB 6.0 有哪些革命性的变化？

2．.NET 应用程序的编译分哪几步？

3．如何启动和退出 VS 2010 环境？怎样建立一个项目？如何打开一个已有的项目？

4．如何向窗体中添加控件？怎样调整控件的位置和大小？

上机实验 1

1．动手实践，学会 Visual Studio 2010 集成开发环境的安装和配置。

2．熟悉 VB.NET 开发工具的各界面元素，了解其功能。

3．熟悉窗体属性窗口中的各个属性，试着改变其中的某几个，运行程序，看看有什么效果。

第 2 章　Visual Basic.NET 可视化程序设计入门

Visual Basic 与其他高级语言（C/C++/Java 等）的根本不同之处在于：它从诞生开始就是一种**完全可视化**的程序设计语言。可视化是 VB 的灵魂，也是一种革命性的编程思想，本章将为读者系统地阐述可视化程序设计的理念。

2.1　什么是可视化程序设计

2.1.1　概述

1. 简介

可视化（Visual）程序设计，亦即**可视化编程**，其含义是：以“**所见即所得**”的编程思想为原则，力图实现编程的可视化，即随时可以看到结果，程序与结果的调整同步。

可视化编程是相对于传统编程方式而言的，作为一种新的程序设计方法，它致力于让程序设计人员利用开发工具本身所提供的各种控件，像搭积木式地构造应用程序界面。

这样，“可视”通常就意味着无须编程，仅通过直观的操作方式即可完成界面设计的任务，因此它已成为目前最理想的 Windows 程序开发方式。

2. 概念

可视化编程涉及的新概念主要有：窗体、控件、属性、事件和事件过程等。

（1）窗体

窗体是指进行程序设计阶段的窗口，用户通过在窗体中放置各种部件来布置应用程序的运行界面。如图 2.1 所示，是一个学生信息注册程序的表单界面。

（2）控件

所谓控件，就是组成程序运行界面的各种部件，如按钮、文本框、复选框、单选按钮和滚动条等。这些控件都位于工具箱中，在图 2.1 中，用箭头和虚框标示出了它们与工具箱中图标的对应关系。

将控件安放到窗体上有两种方法，以下用安放按钮控件为例说明。

① 在工具箱中选择 Button 控件，按住鼠标左键不放，将其拖动到窗体中，然后释放鼠标左键，窗体中便出现了一个名为 Button1 的按钮，可用鼠标选中进一步调整其位置和大小，并设置其上显示的文字。

② 一种简便的方法是，直接双击工具栏 Button 控件，也可在窗体中安放一个按钮。

（3）属性

属性就是控件的性质。它说明控件在程序运行过程中是如何显示的，比如控件的大小是多少、显示在何处、是否可见、是否有效等。

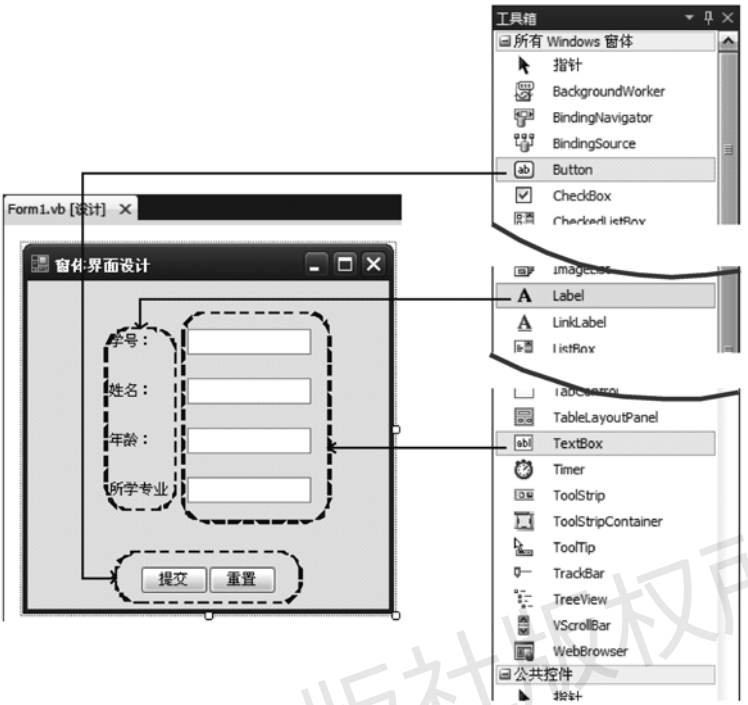


图 2.1　学生信息注册程序的表单界面

属性可分成三类：

① 设计属性。在程序设计时就可发挥作用的属性。

② 运行属性。在程序运行过程中才发挥作用的属性。

③ 只读属性。是一种只能查看而不能改变的属性。

在 VB.NET 中主要通过**属性窗口**（见第 1 章 1.2.4 节）来设置窗体和控件对象的属性。当然，某些情况下也可以在程序代码中给对象的属性赋值。

（4）事件

事件是对一个控件的操作。例如用鼠标单击一个命令按钮，在这里，单击鼠标就称为一个事件（Click 事件）。

（5）事件过程

事件过程就是某个事件发生后要执行的具体操作，类似其他语言中的方法或函数过程。实际上，它就是一段程序代码的集合。

有关事件过程及其驱动机制，本章稍后会举例说明。

3. 可视化编程语言

一种计算机高级语言是否是可视化语言，取决于以下两点：

① 面向对象或必须至少是**基于对象**的，引入了类的概念和事件驱动模式。

② 拥有一套成熟的图形化界面**支持可视化开发**的工具或集成开发环境（IDE）。

这样看来，凡是面向对象编程语言原则上都可发展成可视化编程语言，关键在于开发工具是否支持可视化编程方式。

能进行可视化程序设计的语言有很多，比较常用的有微软公司的 Visual Basic、Visual C++ 以及 Borland 公司的 Delphi 等，今天流行的某些 Java 的 IDE 以及.NET 上的 VB.NET、VC++.NET 和 C# 等也都是可视化编程语言。

但 VB 及其后继者 VB.NET 是**最彻底地**贯彻可视化设计思想的语言，学习可视化程序设计最理想的语言非 VB 和 VB.NET 莫属！

2.1.2　可视化编程环境

Visual Basic 6.0 就是一个成熟的可视化编程环境，VB.NET 开发环境 VS 系列继承了 VB 6.0 的易用性，除了汇集了众多控件的工具箱和位居中央的窗体设计器窗口外，还有一条功能强大的“布局工具栏”来支持程序界面的设计。

在 VS 2010 中选择主菜单【视图】→【工具栏】项，勾选其子菜单中的【布局】选项，如图 2.2 所示，就可以打开【布局工具栏】。

图 2.2　打开【布局工具栏】

【布局工具栏】中各按钮的功能如图 2.3 所示。

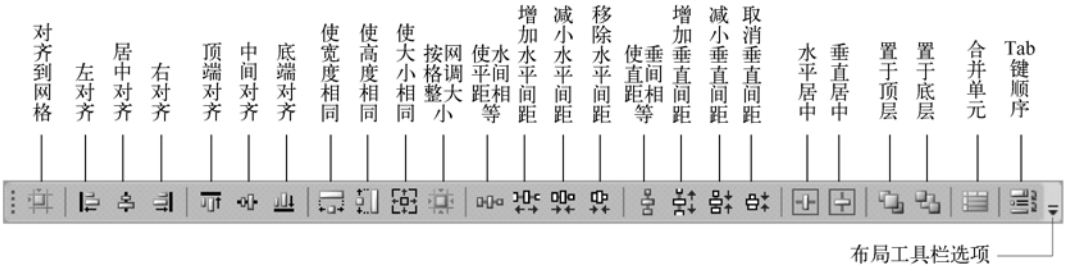


图 2.3　【布局工具栏】各按钮功能

下面举例说明，布局工具栏在设计界面时的应用。

【例 2.1】 设计如图 2.1 所示的学生信息注册表单，要求使用布局工具栏达到界面上各控件的对齐效果。

① 创建 VB.NET 工程，名为 FormDesign。

② 用拖放（或双击）的方法在窗体上安放 4 个标签（Label）、4 个文本框（TextBox）以及两个按钮（Button），如图 2.4（a）所示。

③ 将窗体 Text 的属性设为“窗体界面设计”；在属性窗口中为 4 个标签的 Text 属性分别设为“学号”、“姓名”、“年龄”、“所学专业”；两个按钮的 Text 属性分别设为“提交”和“重置”。用鼠标在窗体上拖动各控件，将它们按照预设计目标界面的大致位置安放好，并且依设计目标（粗略地）调整各控件的大小，这样得到图 2.4（b）所示的初始界面。

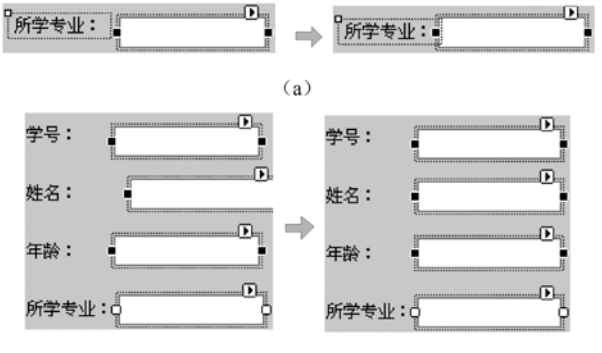
（a）　　（b）　　（c）

图 2.4　初始界面

④ 在图 2.4（b）中先选中“所学专业”标签，然后按住 Ctrl 键依次选中其他三个标签，单击布局工具栏中的“左对齐”（  ）和“使垂直间距相等”（  ）按钮，界面上各标签就如图 2.4（c）所示对齐了。

⑤ 选择“所学专业”标签后，按住 Ctrl 键的同时选中其右边的文本框，然后分别单击布局工具栏中的“中间对齐”（  ）和“移除水平间距”（  ）按钮效果如图 2.5（a）所示。

用同样的方法将其他三个文本框与它们的标签位置相协调，然后以“所学专业”右边的文本框为基准，其他三个文本框与之右对齐，效果如图 2.5（b）所示。



（a）

（b）

图 2.5　对齐文本框

⑥ 用鼠标在窗体上拖出一个矩形区，选中界面上所有的标签和文本框，单击布局工具栏的“水平居中”（）按钮，使所有的标签和文本框都在窗体的中央布局。

⑦ 设置【提交】和【重置】按钮“大小相同”（），移除水平间距并居中。

得到如图 2.1 所示的界面最终设计效果。

2.2　窗体

2.2.1　窗体的本质

窗体（Form）是用户交互的主要载体，是可视化程序设计的基础界面，通常为矩形，通过组合不同控件和编写代码，可从中得到信息并响应该信息。

窗体在本质上是**对象类**，因为它们从 Control 类继承，与.NET 框架中的所有对象一样，窗体是类的实例。如图 2.6 所示，如果查看窗体的对象层次，就会明白它实际上是从 Object 类逐层派生出来的子类。

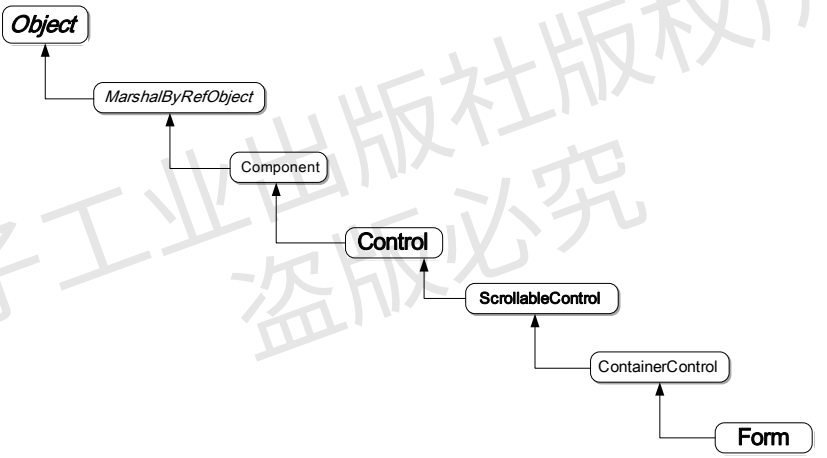


图 2.6　窗体类从 Object 类逐层派生而来

表 2.1 列出了窗体类的父类，以及它从每个父类继承的内容。

表 2.1　窗体类的对象层次表

类的层次（由高到低）	说　明
Object	最高层次的父类，所有的.NET 对象都继承自该类
MarshalByRefObject	由支持远程访问的应用程序使用。这个类可以访问跨应用程序边界的对象
Component	提供 IComponent 接口的基本实现，并且允许在应用程序之间共享对象
Control	是所有带有可视化界面的组件的基类
ScrollableControl	提供自动滚动功能
ContainerControl	允许一个组件包含其他的控件
Form	应用程序的窗口

当用户新建一个 VB.NET 项目时，将自动创建一个默认名为 Form1 的窗体，如图 2.1 所示，用户通过鼠标直接拖动窗体周围的小方块可调整窗体的大小。VB.NET 与以前使用的窗体引擎相比，具有如下优点：① 窗体可以自动改变其中控件的大小；② 可以把控件锁定在特定的位置，而无需借助第三方工具来完成这些工作……可见，用 VB.NET 定制窗体非常简单。

2.2.2 窗体的属性、方法和事件

作为 Form 类的对象，窗体对用户公开了定义其自身外观的属性、决定其行为的方法以及它与用户交互的事件。通过设置窗体属性及编写响应其事件的代码，可自定义各种不同特性的窗体对象，满足应用程序的要求。

1. 窗体的属性

窗体生成后的属性都取默认值，用户可通过【视图】→【属性窗口】菜单项，或按 F4 键，或用鼠标右键单击窗体，在弹出的快捷菜单中选择【属性】命令来激活属性窗口，并对其属性值进行设定。

窗体的常用属性如表 2.2 所示。

表 2.2 窗体的常用属性表

属　性	说　明
Name	用来设置窗体的名称，在程序设计中很重要，默认名称是 Form1，用户可以自己在 Name 属性中修改
Text	相当于 Visual Basic 6.0 中的 Caption 属性，用来设置窗体的标题
Size	它有两个子属性是 Height 和 Width，分别用来设置窗体的高度和宽度
Location	指定窗体在屏幕上的位置，是指窗体左上角的坐标
Font	它有若干子属性，用来描述窗体上的字体信息
IsMdiContainer	设置窗体是否是 MDI 性质的窗体。True 表示是，False 表示否
TopMost	设置窗体是否处于所有窗体的最上方。True 表示是，False 表示否。把它设置为 True，就可以使窗体即使在非激活状态，也能覆盖其他窗体。以前 Visual Basic 6.0 为了实现这样的效果，需要借助于 API 调用
Enabled	设定窗体能否对事件产生影响。它有 True 和 False 两种情况，默认值是 True，表示窗体能对事件产生影响，False 表示禁用
WindowState	设置窗体运行时的状态，Normal 是正常状态，此时的大小和位置由窗体的 Width、Height、Left、Top 决定，Minimized 表示窗体将最小化成图标，Maximized 表示窗体将最大化

2. 窗体的常用方法

VB.NET 有多个方法和语句来控制窗体的加载、显示、隐藏、卸载等，表 2.3 列出了窗体的常用方法。

表 2.3 窗体的常用方法

方　法	功　能
Show	加载并显示窗体
Close	关闭窗体
Hide	隐藏窗体
Update	重绘工作区的无效区域
Refresh	强制对象，使其工作区无效，并立即重绘

调用以上方法的语法格式如下：

```
窗体名.方法( )
```

使用 VS 2010 环境的窗体设计器创建的窗体本质上是类，当在运行中显示窗体的实例时，此类就用作创建新窗体的模板。

【例 2.2】 在对窗体的本质有一定认识的基础上，利用以上介绍的窗体属性和方法，设计一个能在运行时不断创建新窗体的程序。

① 创建 VB.NET 工程，名为 FormShow。

② 在“解决方案资源管理器”中右击工程名，从快捷菜单中选择【添加】→【Windows 窗体】，在【添加新项】对话框中给新窗体命名为“frmhello.vb”，如图 2.7 所示。

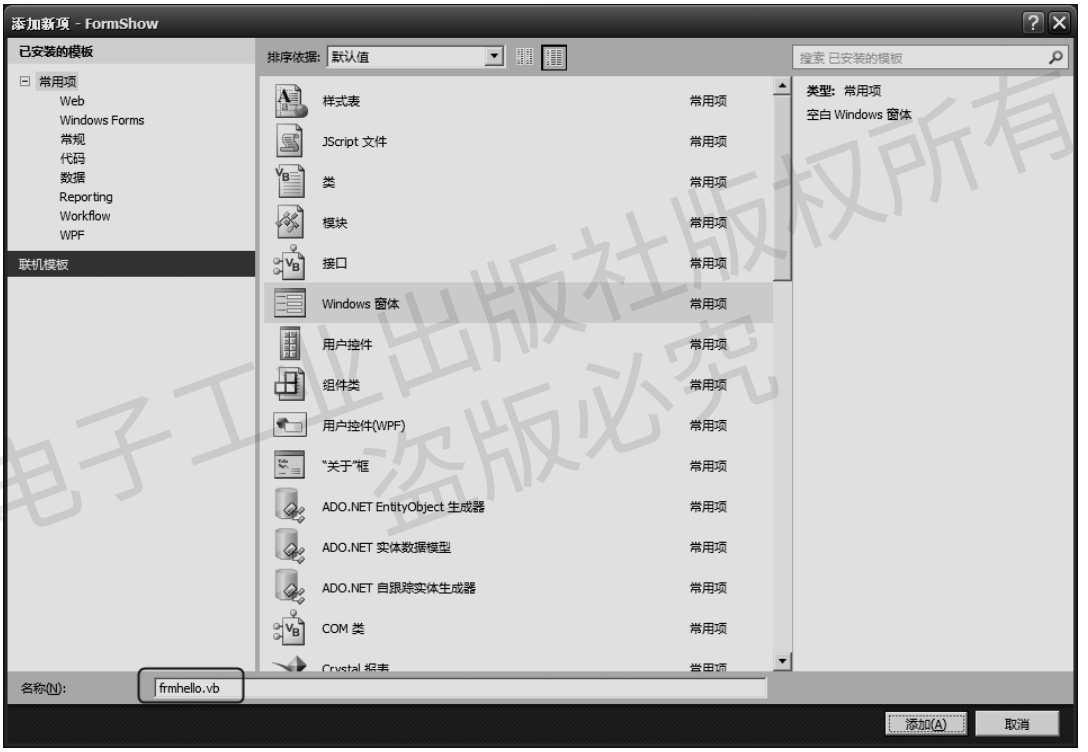


图 2.7　向工程中添加新窗体

③ 在窗体设计器中设计窗体 frmhello 的界面，效果如图 2.8 所示。

该界面设计得很简洁，就是在窗体上安放一个 Label 标签，标签文本 Text 属性设置为“Hello World!”，Font 属性设为“Times New Roman, 21.75pt, style=Bold”（粗二号罗马字体，当然读者也可根据个人喜好设为其他的字体）。

④ 在“解决方案资源管理器”中双击 Form1.vb 项，回到主窗体设计工作区，在属性窗口工具栏里单击“事件”（  ）按钮，在事件列表中双击 Click 行，进入代码编辑模式，输入以下程序代码：

图 2.8　窗体 frmhello 的界面

```
Public Class Form1
    Private Sub Form1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                                    Handles MyBase.Click
        Dim frmhello As New frmhello()
        frmhello.Show()                    '显示 frmhello 窗体
    End Sub
End Class
```

上段代码中加黑的部分，是需要读者自己编写添加的语句。Dim 关键字定义了窗体类 frmhello 的一个实例对象，而 Show 方法显示一个已经装入内存的窗体，如果调用时该窗体没有被加载，则 VB.NET 将自动加载该窗体。

⑤ 运行程序，上面代码将在单击窗体 Form1 后，自动加载 frmhello 窗体，效果如图 2.9 所示。

图 2.9　运行中动态加载窗体 frmhello 的实例

用户每单击一次窗体 Form1，屏幕上都会多出一个新窗体 frmhello，虽然它们长得一模一样，却是不同的对象实例。

注意：在 VB.NET 中，窗体作为类不能直接引用，使用前必须先声明。如上面的代码中先定义了 Dim frmhello As New frmhello()，然后才能调用该对象的 Show 方法。

3. 窗体的事件

每个窗体的实例对象都可以对外界的有关动作进行识别和响应，至于窗体能够响应哪些动作是由系统事先设计定义好的，称为**窗体的事件**。开发人员不能自己创建新的事件，只能给窗体对象所能识别的事件编写代码。

窗体的常用事件如表 2.4 所示。

表 2.4　窗体的常用事件

事　　件	功　　能
Load	加载窗体时触发该事件
Activated	窗体活动时触发该事件
Resize	窗体改变大小时触发该事件
Click	单击窗体时触发该事件

【例 2.3】 编程使窗体响应以下事件：① 单击窗体（Click 事件发生）时，窗体宽度增加 20 个像素点；② 当窗体大小改变（Resize 事件发生）时，程序弹出消息框显示“我变宽了!”。

① 新建 VB.NET 工程，名为 FormSize。

② 在设计器中将窗体调整到合适的大小。

③ 在属性窗口的工具栏里单击“事件”按钮，分别进入相应事件的代码编辑模式。

Click 事件代码：

```
Private Sub Form1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                        Handles MyBase.Click
    Me.Width = Me.Width + 20                '当单击窗体时让该窗体的宽度增加 20 个像素点
End Sub
```

Resize 事件代码：

```
Private Sub Form1_Resize(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                        Handles MyBase.Resize
    MsgBox("我变宽了！ ")                    '当窗体改变大小时显示该消息
End Sub
```

运行效果如图 2.10 所示。

图 2.10　窗体对事件的响应

2.3　控件及其属性

要想设计实用的程序界面，光有窗体还不够，还要在窗体中添加各种控件。控件提供丰富的事件过程，供设计者编写代码，从而完成程序各部分应执行的操作，还可以通过设置其属性值设计出精美的图形用户界面。可见，控件不仅在 VB.NET 可视化程序设计中是一个非常重要的角色，同时它们也是 VB.NET 编程的重要基础。

我们曾说过，控件本质上是继承于 Control 类的子类。事实上，Windows 操作系统完全

采用面向对象和组件化的设计理念，我们所见的一切 GUI 图形元素都是对象类，并且有着共同的祖先。关于这一点，请大家看图 2.11 所示的族谱（注意圆角矩形框出的分支）。

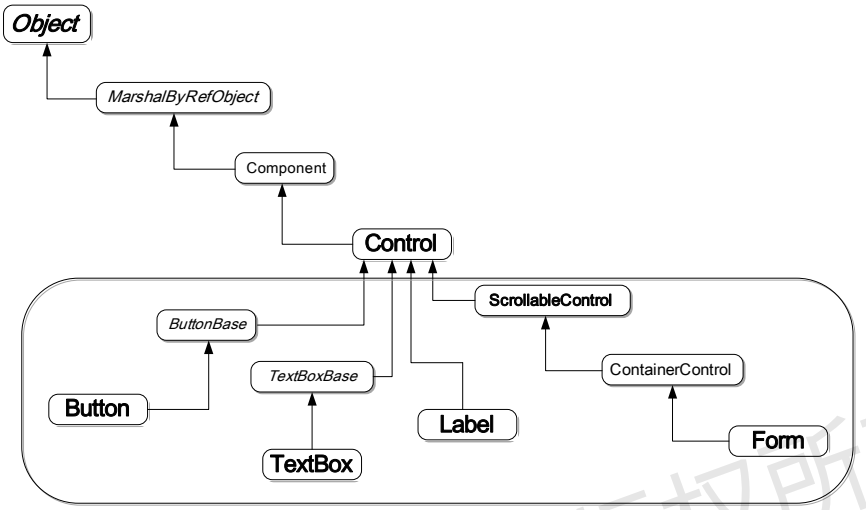


图 2.11　本是同根生的控件族谱

大家不妨将图 2.11 与图 2.6 仔细对比一下，就会明白，原来一切 Windows 图形元素皆是由 Control 类衍生而来的，**窗体实质上也是一种特殊的控件**，它们“本是同根生”!

本节仅介绍标签、文本框和按钮这三种最基本的控件及其属性。

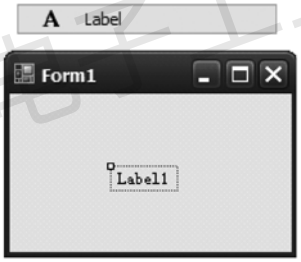


图 2.12　Label 控件的外观

2.3.1　最基本的控件

1. 标签

标签（Label）控件用于显示字符串，通常显示的是文字说明信息，用来标识输入或输出区域。

Label 控件在工具箱中的条目及其在窗体中的显示外观如图 2.12 所示。

表 2.5 给出了 Label 控件的常用属性。

表 2.5　Label 控件的常用属性

属　性	说　明
Autosize	设置标签能自动调整大小，以显示所有的内容
BorderStyle	设置标签是否具有边框以及边框的样式
Name	设置标签的名称，默认标签名为 Label1、Label2、…
Image	设置标签的背景图像
TabIndex	设置标签的索引
Text	设置标签上显示的文本
TextAlign	设置标签上显示字符的对齐方式
Visible	设置标签是否显示在窗体上

对于 Label 控件一般不写事件代码，尽管它也能响应很多事件，如 Click、Resize、TextChanged 等，但是在实际使用中，它主要用来标识信息。如果有特殊的需要，也可以编写事件代码程序，让它响应相应的事件。

2. 文本框

文本框（TextBox）控件用来显示输入和输出的文本信息，是开发应用程序时最常用的控件之一，其在工具箱中的条目及在窗体中的显示外观如图 2.13 所示。

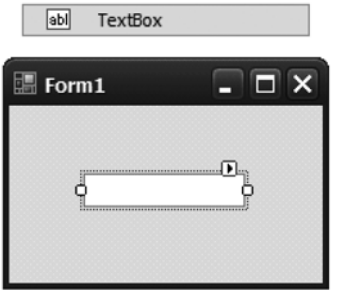


图 2.13　TextBox 控件的外观

TextBox 控件是相当灵活的数据显示工具，通常用于编辑文本，不过也可使其成为只读控件。它可以显示单行或多行文本；还可以对文本换行、控件大小及添加基本格式进行设置。

TextBox 控件显示的文本包含在其 Text 属性中。默认情况下，最多可在一个文本框中输入 2048 个字符。如果将 MultiLine 属性设置为 True，则最多可输入 32 KB 的文本。可以在运行时通过读取 Text 属性来检索文本框的当前内容。

表 2.6 列出了 TextBox 控件的常用属性。

表 2.6　TextBox 控件的常用属性

属　性	说　明
Text	用来存放数据。它的内容可预先输入，向控件赋予初始值。运行时，该属性用于取得用户输入的字符或向 Text 属性指定一个新值以替换现有文本。它是一个字符串，可以和 Visual Basic 的普通字符串变量一样使用
TextAlign	用来设定输入文字的对齐方式
MultiLine	用来设置文本框是否能接受多行文字。如果其值为 False，表示文本框只能接受单行文本；如果为 True，表示输入较长的字符串时可以自动换行
MaxLengh	用来设置文本框中所能放入的最大字符长度，默认值为 32767。如果在程序代码中将长度超过 MaxLength 属性设置值的文本赋给文本框，VB.NET 并不产生错误，但只赋给 Text 属性最大数量的字符，而额外的字符被截去。改变该属性不会对 TextBox 的当前内容产生影响，但将影响到以后内容的改变
ScrollBars	用来设置文本框中是否出现水平或垂直滚动条。单行文本框可以用一个水平滚动条，使用户可以浏览长文本行的任何部分。多行文本框可以增加一个水平或垂直滚动条或两个滚动条
PasswordChar	可在窗口中加入想要取代当前字符显示的符号。例如，在输入密码时，不想让输入的文字内容显示在屏幕上，就可以用*号作为显示符号。设置此属性后，无论输入什么字符，显示出来的都是 PasswordChar 指定的替代符号
ReadOnly	设置文本框是否为只读，如果用于接收用户数据，可设置为 False；如果仅仅是为了显示数据，可设置为 True
SelectionStart	设置文本框中选择的文本的起始位置
SelectionLength	设置文本框中选择的文本的长度
SelectedText	返回文本框中选择的文本内容

TextBox 控件最常用的事件是 TextChanged 事件，该事件在文本框的 Text 属性发生改变时触发。文本框还有自己的方法，它们为开发人员设置文本框提供了方便。表 2.7 列出了 TextBox 控件的常用方法。

表 2.7　TextBox 控件的常用方法

方　法	说　明
Copy	将选取的文本复制到剪贴板
Paste	将剪贴板中的文本粘贴到文本框中
Cut	将选取的文本剪切下并复制到剪贴板
Undo	文本框复原为最后一次改动时的内容
Clear	清除文本框的内容
SelectAll	选取全部文本

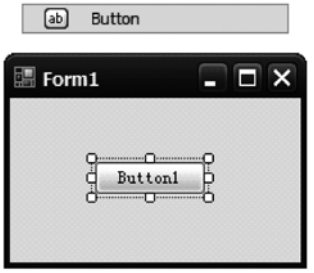


图 2.14　Button 控件的外观

3. 按钮

按钮（Button）控件用来执行某一命令功能，也称命令按钮。命令按钮在工具箱中的条目及其在窗体中的显示外观如图 2.14 所示。

命令按钮的最常用事件是 Click 事件。通常在 Click 事件中编写一段代码，当用户用鼠标单击该按钮时，就会执行某一特定的功能。

表 2.8 列出了 Button 控件的常用属性。

表 2.8　Button 控件的常用属性

属　性	说　明
Name	设置按钮的名称，可以在程序代码中使用这个名称对按钮进行操作
Visible	设置按钮是否可见，True 为可见，False 为不可见
Enable	设置按钮是否有效。为 True（默认值）时，表示命令按钮可以响应外部事件；False 表示按钮不能响应外部事件，这时命令按钮以淡色或模糊颜色显示
Text	设置按钮上显示的文本
TextAlign	设置按钮上文字的对齐方式

2.3.2　简单控件应用

【例 2.4】 在一个文本框中输入字符，在另一个文本框中同步显示相应的内容。

① 新建 VB.NET 工程，名为 TextChangeShow。

② 设计界面。

窗体的 Text 属性设为“文本同步显示”，在窗体上放置两个标签、两个文本框和一个按钮；设置两个标签的 Text 属性分别为“输入：”和“显示：”；设置两个文本框的 Text 属性均为空；设置按钮的 Text 属性为“清空”。用 2.1.2 节所讲的方法布局并对齐各控件，设计完成的界面如图 2.15 所示。

图 2.15　设计完成的界面

③ 本例要给 TextBox1 和 Button1 编写事件代码，选中目标控件，在属性窗口的工具栏里单击“事件”按钮，分别进入相应事件的代码编辑模式。

TextBox1 的 TextChanged 事件代码：

```
Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                        Handles TextBox1.TextChanged
    TextBox1.SelectAll()                                '选中 TextBox1 中的所有文本
    TextBox1.Copy()                                     '将选中的文本复制到剪贴板上
    TextBox2.Clear()                                    '将 TextBox2 中的所有文本删除
    TextBox2.Paste()                                    '将剪贴板中的文本粘贴到 TextBox2 中
    TextBox1.SelectionStart = Len(TextBox1.Text)        '取消 TextBox1 文本的选中状态
End Sub
```

Button1 的 Click 事件代码：

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                        Handles Button1.Click
    TextBox1.Clear()                                    '清空 TextBox1 的内容
    TextBox2.Clear()                                    '清空 TextBox2 的内容
End Sub
```

④ 运行程序，在标签“输入：”后的文本框输入任意字符，下方的“显示：”标签后的文本框会同步显示这些字符，如图 2.16 所示。

图 2.16　运行时界面

无论何时，单击【清空】按钮都会将两个文本框全部清空。

2.4　事件驱动

在上文的几个例子中大家会发现，编写程序代码都是在特定的事件过程中进行的。**事件驱动**是 VB.NET 的基本编程范式，为此，这里专辟一节系统地阐述事件驱动。

2.4.1　基本概念

所谓**事件驱动**，通俗地说，就是你点什么按钮（即产生什么事件），计算机就执行什么操作（即执行那个事件的过程代码）。当然，事件的种类繁多，远不止“单击按钮”这一种，甚至也不仅限于用户的操作。

事件驱动的核心自然是事件。从事件角度说，事件驱动程序由一个事件收集器、一个事件发送器和一个事件处理器组成。

（1）事件收集器。专门负责收集所有事件，包括来自用户的（如鼠标、键盘事件等）、来自硬件的（如时钟事件等）和来自软件的（如操作系统、应用程序本身的消息等）事件。

（2）事件发送器。负责将收集器收集到的事件分发到目标对象中。

（3）事件处理器。做具体的事件响应工作，它往往要到实现阶段才完全确定。

由于 Windows 本身是基于事件驱动模型的，因而在 Windows 操作系统下实现事件驱动模式的编程相当便利。值得庆幸的是，事件收集器已经由 Windows 完成了；事件发送器也由 Windows 完成了一部分，而.NET 及 VS 2010 环境又进一步将事件发送器和处理器彻底完善并封装了起来，对 VB.NET 程序员完全透明……感谢 Windows 和.NET 平台所做的这一切，若靠用户自己手工编程来实现事件驱动机制，所要做的工作的复杂程度可能远远超出我们的想象！对于 VB.NET 程序员来说，唯一能够接触到的就是事件处理器，编写事件处理代码才是他们所关心的内容。

2.4.2 事件过程

广义地说，事件驱动中的“事件”，是指能被对象（窗体和控件）识别的一切动作。例如，单击（Click）、双击（DoubleClick）、内容改变（TextChanged）和定时（Timer）等。为一个事件所编写的程序代码称为**事件过程**。当 VB.NET 对象的某个事件发生时，便自动调用相应的事件过程。

事件过程是一种特殊的函数过程，它附加在窗体或控件上，故分为**窗体事件过程**和**控件事件过程**。事件过程前面的声明都使用 Private 来指定它的有效范围。

1．窗体事件过程

窗体事件过程由窗体名（Name 属性）、下划线和事件名组成，其语法格式如下：

```
Private Sub  窗体名_事件名 ([参数列表])
    [局部变量和常数声明]
    语句块
End Sub
```

例如，在例 2.2 中，单击窗体 Form1 的事件过程 Form1_Click 执行的操作是创建并显示窗体 frmhello，其代码如下：

```
Private Sub Form1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                          Handles MyBase.Click
    Dim frmhello As New frmhello()
    frmhello.Show()                                    '显示  frmhello  窗体
End Sub
```

2．控件事件过程

控件事件过程由控件名（Name 属性）、下划线和事件名组成，其语法格式如下：

```
Private Sub  控件名_事件名 ([参数列表])
    [局部变量和常数声明]
    语句块
End Sub
```

例如，在例 2.4 中单击【清空】（Button1）按钮的事件过程，执行清空两个文本框内容的操作，其代码如下：

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                              Handles Button1.Click
    TextBox1.Clear()                                    '清空 TextBox1 的内容
    TextBox2.Clear()                                    '清空 TextBox2 的内容
End Sub
```

3．如何建立事件过程

前面的程序实例都是在“代码编辑器”窗口中编写事件过程的，“代码编辑器”窗口会自动显示 VB.NET 的保留字，以便用户很容易地看出哪些是自己的编码。

打开“代码编辑器”窗口有以下三种方法：

（1）最简便的方法是在设计的窗体上双击窗体或控件，系统会自动打开“代码编辑器”窗口，并出现该窗体或控件的默认过程代码。例如，在例 2.2 中，双击窗体 Form1 进入代码编辑模式，出现的默认过程代码为：

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                              Handles MyBase.Load
End Sub
```

（2）选择【视图】→【代码】菜单项，也可打开“代码编辑器”窗口。不过这种方式需要用户自己编写事件过程。比如，在代码编辑窗口中直接输入上段“Form1_Load”代码。

注意：写代码的时候，要严格按照以上定义的格式来写，不要随意改变事件或对象的名称，若想改变，应该通过属性窗口操作。

（3）通过属性窗口进入特定对象的事件过程，即选中目标控件后，在属性窗口的工具栏里单击“事件”（  ）按钮，进入相应事件过程的代码编辑模式。这也是我们在本章前面一些例子中使用的方法，它充分利用了开发环境本身提供的可视化功能，能自动生成一些固定格式的代码框架，引导用户把主要精力放在实现程序应用功能的语句编写上，自动化程度高，不易出错，推荐大家采用。

2.4.3　常用事件

事件的种类很多，要掌握所有的事件可不是一件容易的事。本小节将介绍 VB.NET 中最常用的两大类事件：鼠标事件和键盘事件。

1. 鼠标事件

鼠标事件是 VB.NET 中最常用的事件，它是由鼠标触发的。鼠标事件包括 Click、DoubleClick、MouseMove、MouseDown、MouseUp 等，本章的许多例子已经用到了鼠标的相关事件。

表 2.9 列出了常见的鼠标事件及其触发条件。

表 2.9　常见的鼠标事件及其触发条件

事 件 名 称	触 发 条 件
Click	单击鼠标左键时触发
DoubleClick	双击鼠标左键时触发
MouseMove	鼠标移动时触发
MouseDown	鼠标被按下时触发
MouseUp	鼠标被按下又被释放时触发
MouseLeave	鼠标离开对象时触发
MouseWheel	鼠标中间的滚动轮滚动时触发

 注意：在一个单击鼠标过程中，其实触发了三个事件，依次为 MouseDown、Click 和 MouseUp。双击鼠标则依次触发了 6 个事件，分别为 MouseDown、Click、MouseUp、MouseDown、DoubleClick、MouseUp。

除非要确定事件的触发顺序，否则不必对鼠标的所有事件编程，只需对其中的一个或几个事件编写代码即可。

【例 2.5】 测试鼠标事件。

① 新建 VB.NET 工程，名为 MouseEvent。

② 界面设计。

窗体 Text 属性设为“鼠标事件测试”。在窗体上放置 1 个文本框和 1 个标签，设置文本框的 ScrollBars 属性为 Vertical，Multiline 属性为 True；设置标签的 BorderStyle 属性为 Fixed3D，Font 属性为“楷体_GB2312，18pt”（小二号字），Text 属性为“鼠标动作区”。设计后的界面效果如图 2.17 所示。

③ 给 Label1 的不同鼠标事件编写响应代码，每一种事件触发后都将在文本框中显示该事件的名称，从中可以看出各个事件的执行顺序。

Label1 对象的几种鼠标事件的响应代码如下。

MouseDown（鼠标按下）事件代码：

```
Private Sub Label1_MouseDown(ByVal sender As System.Object, ByVal e As
                             System.Windows.Forms.MouseEventArgs) Handles Label1.MouseDown
    TextBox1.Text = TextBox1.Text + "MouseDown  事件" + vbCrLf
End Sub
```

MouseUp（鼠标释放）事件代码：

```
Private Sub Label1_MouseUp(ByVal sender As System.Object, ByVal e As
                           System.Windows.Forms.MouseEventArgs) Handles Label1.MouseUp
    TextBox1.Text = TextBox1.Text + "MouseUp  事件" + vbCrLf
End Sub
```

Click（鼠标单击）事件代码：

```
Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
                                                    Handles Label1.Click
        TextBox1.Text = TextBox1.Text + "Click 事件" + vbCrLf
    End Sub
```

DoubleClick（鼠标双击）事件代码：

```
    Private Sub Label1_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                    Handles Label1.DoubleClick
        TextBox1.Text = TextBox1.Text + "DoubleClick 事件" + vbCrLf
    End Sub
```

④ 测试结果。

图 2.18 显示了在“鼠标动作区”双击鼠标左键后的事件顺序。

图 2.17　测试鼠标事件的程序界面

图 2.18　双击鼠标左键后的事件顺序

2. 键盘事件

键盘事件也是 VB.NET 中最常见的事件，它由键盘触发。键盘事件包括 KeyDown、KeyUp 和 KeyPress 等，分别代表键被按下、键弹起和一个完整的按键事件。可以通过对象参数 e 的 KeyChar 属性来判定用户的按键。

【例 2.6】 测试键盘事件。

① 新建 VB.NET 工程，名为 KeyEvent。

② 界面设计。

窗体 Text 属性设为“键盘事件测试”。在窗体上放置 1 个文本框和 1 个标签，设置文本框的 ReadOnly 属性为 True，Size 属性为“21, 21”；设置标签的 Text 属性为“您按下的键是：”。设计后的界面效果如图 2.19 所示。

③ 编写文本框的 KeyPress 事件代码：

```
    Private Sub TextBox1_KeyPress(ByVal sender As System.Object, ByVal e As
                                  System.Windows.Forms.KeyPressEventArgs) Handles TextBox1.KeyPress
        TextBox1.Text = e.KeyChar
    End Sub
```

④ 测试结果。

运行程序，用户在键盘上任意按键，文本框内即时显示用户所按键的键值，如图 2.20 所示。

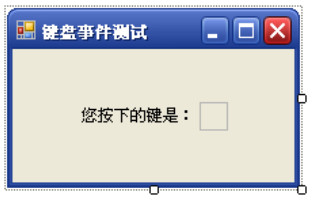


图 2.19　测试键盘事件的程序界面

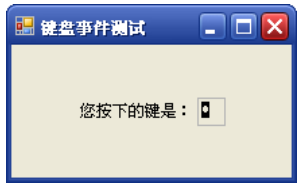


图 2.20　即时显示用户按键的键值

图 2.20 中显示的是用户按下 Backspace（退格键）时的情况，退格键的可显示字符是“□”。

2.5　可视化程序设计举例

本节以一个计算圆面积的程序为例，结合前面几节的知识点，系统地介绍可视化程序设计的一般流程和方法，引领大家进入 VB.NET 程序设计的大门。

2.5.1　可视化程序设计的一般步骤

使用 VB.NET 编程，一般先设计应用程序界面，然后再分别编写各对象事件的程序代码或其他处理程序，一般步骤归纳如下。

1．创建应用程序界面

界面是用户和程序交互的桥梁，用 VB.NET 创建的 Windows 应用程序界面一般由窗体、按钮、菜单、文本框和图像框等构成。根据程序的功能要求和用户与程序之间的信息交流的需要来确定需要哪些对象，规划界面的布局。

2．设置窗体和控件的属性

根据规划的界面要求设置各个窗体和控件对象的属性，如对象的外观、名称、颜色、大小等。大多数属性的取值既可以在设计时通过属性窗口来设置，也可以在程序运行时通过编程来动态地设置或修改。

3．编写程序代码

采用事件驱动的编程机制，针对窗体中各控件所支持的事件或方法编写代码。当界面设计完成后，就可以通过“代码编辑器”窗口来编写事件过程代码了，以完成对相应事件作出响应和信息处理等任务。

4．保存应用程序

一个 VB.NET 程序就是一个项目，在建立一个新的应用程序（项目）时，系统要求用户输入项目名称和存放路径，然后根据用户提供的项目名，在指定文件夹中建立一个用项目名命名的子文件夹，并在该子文件夹中保存与应用程序有关的所有文件，包括解决方案文件（.sln）、项目文件（.vbproj）、窗体文件（.vb）等。当打开一个项目（文件）时，与该项目有关的所有文件同时被装载。

5．运行和调试程序

程序设计并保存后，需要调试运行程序以便发现错误。可以通过【调试】→【启动调试】菜单项来运行程序，也可通过工具栏中的“启动调试”（  ）按钮或 F5 键来运行程序。当运行中出现错误时，VB.NET 将在“输出”窗口的“调试”窗口中显示错误信息。若程序没有错误，运行后将在项目所存放的文件夹的“bin”子文件夹中，自动生成应用程序的可执行文件（扩展名为.exe），该可执行文件可以脱离 VB.NET 环境单独运行。

2.5.2　一个计算圆面积的程序

【例 2.7】 窗体界面由 2 个标签（Label）、2 个文本框（TextBox）和 2 个命令按钮（Button）组成。在设计时，文本框中为空白。在运行时，输入半径后单击【计算】命令按钮，圆面积文本框中会显示面积值，单击【退出】命令按钮，将结束程序运行，效果如图 2.21 所示。

图 2.21　计算圆面积的程序界面

1．创建应用程序界面

（1）创建解决方案和项目。启动 VB.NET，选择【文件】→【新建项目】，“项目类型”选择“Visual Basic/Windows”，“模板”选择“Windows 窗体应用程序”，给项目命名为“CircleArea”。确定后系统新建一个解决方案（默认名是“CircleArea”），该解决方案中包含了项目“CircleArea”，项目中包含一个窗体文件（默认名为 Form1.vb），此时屏幕上会出现一个空白窗体。

（2）向窗体中添加控件。使用工具箱在窗体中添加标签（Label）、按钮（Button）和文本框（TextBox）三种控件。

（3）调整、移动和布局控件。按照 2.1.2 节例 2.1 所演示的方法，参照图 2.21 调整、移动和对齐控件，布局本程序的 GUI 界面。

2．设置窗体和控件的属性

（1）窗体的标题为“圆面积计算”，因此需将窗体对象 Form1 的 Text 属性值设为“圆面积计算”。

（2）当程序启动后，文本框（TextBox1 和 TextBox2）中应没有文本内容，而文本框在启动后显示的默认内容是由其 Text 属性值决定的，因此需设置其 Text 属性值为空（即删除该属性原有的值）。

（3）为了表示文本框的含义，分别在 2 个文本框左侧添加了一个标签控件，标签显示的文字是由其 Text 属性决定的，故分别设置标签 Label1 和 Label2 的 Text 属性值为“半径：”

和“圆面积：”。

（4）在程序界面中还有两个命令按钮（Button1 和 Button2），按钮上显示的文字是由其 Text 属性值决定的，故分别设置 Button1 和 Button2 的 Text 属性值为“计算”和“退出”。

各控件的属性设置见表 2.10。

表 2.10　各控件的属性设置

对　象	控 件 名	属 性 名	属 性 值
Form	Form1	Text	圆面积计算
Label	Label1	Text	半径：
Label	Label2	Text	圆面积：
TextBox	TextBox1	Text	
TextBox	TextBox2	Text	
Button	Button1	Text	计算
Button	Button2	Text	退出

3．编写程序代码

当窗体和控件的布局及其属性设置完成后，下一步就要编写代码来实现功能了。本例中有 2 个事件要处理，分别是【计算】和【退出】按钮的鼠标单击事件。VB.NET 集成开发环境能自动生成事件代码的模板，用户只需进入“代码编辑器”窗口，在生成的模板中添加自己的代码即可，如图 2.22 所示。

图 2.22　代码编辑器窗口

选择 Click 事件，在 Sub 和 End Sub 语句之间输入下列代码，使单击【计算】按钮时，计算圆面积并在 TextBox2 文本框中显示结果：

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                                                        Handles Button1.Click
    Dim r As Single                     '声明一个实数类型的变量 r
    r = TextBox1.Text                   '从 TextBox1 中读取半径值存入变量 r
    TextBox2.Text = Math.PI * r * r     '计算圆面积并显示在 TextBox2 中
End Sub
```

【退出】按钮的鼠标单击事件代码如下：

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
                                                    Handles Button2.Click
        End
    End Sub
```

4．保存应用程序

使用【文件】→【保存】菜单命令或单击工具栏上的“保存”按钮，可将正在编辑的代码和设计的窗体存盘；若使用【文件】→【全部保存】命令或单击工具栏上的“全部保存”按钮，则可保存当前项目中的所有文件。

5．运行和调试程序

运行程序有如下三种方法：

（1）使用【调试】→【启动调试】菜单命令。

（2）单击工具栏中的运行按钮。

（3）按 F5 键。

运行程序后，即显示用户界面，输入半径值后单击【计算】命令按钮，圆面积文本框中会显示面积值，如图 2.21 所示，再单击【退出】命令按钮，程序即结束，窗口被关闭。

习　题　2

1．什么是可视化程序设计？简述其基本概念。

2．窗体的本质是什么？有哪些常用的属性和方法？

3．什么是事件驱动机制？建立事件过程有哪三种方式？

上机实验 2

1．按照【例 2.1】操作，练习可视化编程环境的使用，熟悉布局工具栏各按钮的功能。

2．仿照【例 2.4】制作一些小程序，熟练基本控件的使用。

3．按照【例 2.7】的讲解，学着制作计算圆面积的程序，从中体会可视化程序开发的通行流程。

第 3 章　Visual Basic.NET 语言基础

通过前面两章的学习，大家已经初步了解了 VB.NET 可视化编程的知识，但若想真正掌握 VB.NET 编程，还是要从语言基础学起。本章将系统介绍 Visual Basic.NET 语言，包括基本数据类型、数据的运算、语句控制结构、函数、数组以及 VB.NET 程序面向过程的构成和程序设计方法等。

3.1　基本数据类型

在应用程序中总要处理各种数据，编程语言通过规定一些数据类型来存储不同类型的数据。Visual Basic.NET 内置了 12 种基本数据类型，每种类型拥有一个固定的名字，表 3.1 列出了所有的基本数据类型。

表 3.1　12 种 Visual Basic.NET 基本数据类型

类　型	关键字	存储空间（字节）	取 值 范 围
字符型	Char	2	0～65535（无符号）
变长字符串型	String	取决于实现平台	0 到约 20 亿个 Unicode 字符
字节型	Byte	1	0～255（无符号）
短整型	Short	2	–32768～32767
整型	Integer	4	–2147483648～2147483647
长整型	Long	8	–9223372036854775808～9223372036854775807
小数型	Decimal	16	无小数点时为+/– 79228162514264337593543950335 有小数点时为+/– 7.9228162514264337593543950335 之间带 28 位小数的数 最小的非 0 数为 +/– 0.0000000000000000000000000001（28 位小数）
单精度浮点型	Single	4	负数范围 –3.402823E38～–1.401298E–45 正数范围 1.401298E–45～3.402823E38
双精度浮点型	Double	8	负数范围 –1.79769313486232E308～–4.94065645841247E–324 正数范围 4.94065645841247E–324～1.79769313486232E308
布尔型	Boolean	取决于实现平台	True 或 False
日期型	DateTime	8	0001 年 1 月 1 日 0:00:00 到 9999 年 12 月 31 日 23:59:59
对象型	Object	4（32 位平台）或 8（64 位平台）	任何类型数据都可以存储在 Object 类型的变量中

这 12 种类型可分为三大类：数值、字符和其他数据类型。下面分别介绍它们。

3.1.1　数值数据类型

数值数据类型包括字节型（Byte）、短整型（Short）、整型（Integer）、长整型（Long）、小数型（Decimal）、单精度浮点型（Single）和双精度浮点型（Double）7 种。

1．字节型（Byte）

字节型是无符号整型数据类型，用 1 字节（8 位）来存储数值，可存储 0～255（$=2^8-1$）范围内的整数。例如，下面的语句将 128 赋值给 bytMyByte 变量：

```
Dim bytMyByte As Byte
bytMyByte = 128
```

Byte 是非常有用的数据类型，因字节是计算机的基本存储单元，以字节为单位的处理或算术运算速度极快。

2．短整型（Short）

短整型是整型数据类型，用 2 字节（16 位）来存储带符号整数，其可表示的整数范围是−32768～+32767，即-2^{15}～$+2^{15}-1$。例如，下面的语句将−20128 赋值给 shoMyShort 变量：

```
Dim shoMyShort As Short
shoMyShort = −20128
```

3．整型（Integer）

整型也用于存储带符号整数，它用 4 字节（32 位）来存储数值，其可表示的整数范围是−2147483648～+2147483647，即-2^{31}～$+2^{31}-1$。例如，下面的语句将 200012 赋值给 intMyInteger 变量：

```
Dim intMyInteger As Integer
intMyInteger = 200012
```

4．长整型（Long）

长整型以 8 字节（64 位）来存储带符号整数，其可表示的整数范围是−9223372036854775808～+9223372036854775807，即-2^{63}～$+2^{63}-1$。例如，下面的语句将 200012012812 赋值给 lngMyLong 变量：

```
Dim lngMyLong As Long
lngMyLong = 200012012812
```

5．小数型（Decimal）

小数型是非整型数据类型，用来存储小数，它以 16 字节（128 位）来存储数值。当小数位为 0 时，它所支持的最大可能值为+/−79228162514264337593543950335。对于 28 位小数而言，最大支持数非常小，为+/−7.9228162514264337593543950335 ，而最小的非 0 值为 +/− 0.0000000000000000000000000001（28 位小数）。例如，下面的语句将 0.00012 赋值给 decMyDecimal 变量：

```
Dim decMyDecimal As Decimal
decMyDecimal = 0.00012
```

Decimal 类型比较适合财务类数据的计算，即需要记录的数的位数很大，但又不允许出现四舍五入的计算误差。

6．单精度浮点型（Single）

单精度浮点型用来存储单精度浮点数，它用以 4 字节（32 位）来存储数值，其中符号占 1 位，指数占 8 位，其余 23 位表示尾数。单精度浮点型比小数型支持的有效位少，且可能导致四舍五入的误差，但它可以表示的数的范围比小数型要大。单精度浮点数可以精确到 7 位十进制数，其负数的取值范围为−3.402823E+38～−1.401298E−45，正数的取值范围为 1.401298E−45～3.402823E+38。例如，下面的语句将−1.4E06 赋值给 sngMySingle 变量：

```
Dim sngMySingle As Single
sngMySingle = −1.4E06
```

 注意：浮点数可以用 mmmEeee 格式来表示，其中 mmm 表示尾数部分（有效数字），eee 表示指数部分（10 的方幂）。

7．双精度浮点型（Double）

双精度浮点型用来存储双精度浮点数，它用 8 字节（64 位）来存储数值，其中符号占 1 位，指数占 11 位，其余 52 位表示尾数。双精度浮点数比单精度浮点数支持更大的数据范围，它可以精确到 15 或 16 位十进制数，其负数的取值范围为−1.79769313486232E+308～−4.94065645841247E−324，正数的取值范围为 4.94065645841247E−324～1.79769313486232E+308。例如，下面的语句将 1.4000123E106 赋值给 dblMyDouble 变量：

```
Dim dblMyDouble As Double
dblMyDouble = 1.4000123E106
```

3.1.2 字符数据类型

字符数据类型包括字符型（Char）和字符串型（String）两种。

1．字符型（Char）

Char 类型是单个 Unicode 字符，以 16 位无符号数值形式存储，即一个 Unicode 字符用 2 字节存储。

一般来说，Char 数据类型用于存储单个字符，如下面的语句将字符“A”赋值给 chrMyChar 变量：

```
Dim chrMyChar As Char
chrMyChar = "A"
```

采用这种方式，可以将单个符号存入定义为 Char 类型的变量中，且它以 0～65535 之间的数字形式存储，但如果显示 Char 类型变量内容，用户看到的仍然是一个文本符号。

 注意：

① Unicode 是以数字形式表示符号的国际标准码，它克服了不同编码系统存在的问题。Unicode 与语言、平台及程序无关。读者通过 www.unicode.org 可以了解更多信息。

② 虽然 Char 数据类型是以无符号的数值形式存储，但是不能直接在 Char 类型和数值类型之间进行转换。

2．字符串型（String）

字符串是一个字符序列，以不带符号的 16 位（2 字节）数字序列形式存储，每个数字的取值范围是 0～65535，表示一个 Unicode 字符。一个字符串可以存储 0～2^{31}（约 21 亿）个 Unicode 字符。

String 类型的字符串放在一对西文双引号内，其中长度为 0 的字符串（不含任何字符）称为空字符串。例如，下面的一组语句将“张三”赋值给 strMyName 变量，为 strMyStr 变量赋值为空字符串，将“2.0”赋值给 strMyNumStr 变量：

```
Dim  strMyName  As  String
Dim  strMyStr  As  String
Dim  strMyNumStr  As  String
strMyName  =  "张三"
strMyStr  =  ""
strMyNumStr  =  "2.0"
```

3.1.3　其他数据类型

其他数据类型包括布尔型（Boolean）、日期型（Date）和对象型（Object）三种。

1．布尔型（Boolean）

布尔型变量用 2 字节（16 位）的数值形式存储逻辑值，其值只能是 True 或 False。在 Visual Basic.NET 中，如果将这两个逻辑值转换成数值类型，对应的值分别为−1 和 0。当将数值类型转换为布尔值时，0 转换为 False，其他值则转换为 True。

布尔变量的值可用于记录布尔型变量的状态，一个布尔型变量的状态不是 True 就是 False。当用户要完成某一任务（如打开一个窗口）时，可将变量设置成布尔型，并通过变量的值确定执行哪个操作。例如，下面的代码根据布尔型变量 blnMyBoolean 的值确定执行 A 操作或 B 操作：

```
Dim  blnMyBoolean  As  Boolean
blnMyBoolean  =  False
If  blnMyBoolean  =  False Then
     ' 执行 A 操作
Else
     ' 执行 B 操作
End If
```

注意：永远不要编写依赖 True 和 False 的等价数值的代码，即应当限定将 Boolean 变量作为逻辑值，而不要用与其等价的数值 1 或 0 来代替。

2．日期型（Date）

日期型变量用 8 字节（64 位）的整数值形式存储。VB.NET 把对 Date 类型的处理与数值类型区分开来。Date 类型必须以 mm/dd/yyyy（月/日/年）的格式定义，如 12/22/2012。文字串必须以一对“#”括起来，如#12/22/2012#，该类型可表示的日期范围从公元 1 年 1 月 1

日到公元 9999 年 12 月 31 日。例如，可以按以下方式定义存储日期变量：

```
Dim datMyDate1 As Date
datMyDate1 = #12/22/2012#
```

Date 数据类型也用于存储时间信息，所存储的时间范围可以是 00:00:00 到 23:59:59 之间的任意值，时间数据必须以 hh:mm:ss（小时:分钟:秒）格式定义，如 16:20:09。例如，可以按以下方式定义时间变量：

```
Dim datMyDate2 As Date
datMyDate2 = #16:20:09#
```

Date 数据类型也可同时存储日期和时间信息，数据必须以“mm/dd/yyyy hh:mm:ss”（月/日/年 小时:分钟:秒）格式定义。例如，可以按以下方式定义日期时间变量：

```
Dim datMyDate3 As Date
datMyDate3 = #12/6/2004 10:30:05#
```

3．对象型（Object）

对象型变量用 4 字节（32 位）地址形式存储，该地址为对象引用。可以为声明是 Object 类型的变量分配任何引用类型（字符串、数组、类或接口），Object 变量也可引用其他任何数据类型的数据（数值型、布尔型、字符型、日期型、结构型或枚举型）。

以上介绍了 Visual Basic.NET 的基本数据类型，此外，还可以使用数组、枚举、结构、集合、类和接口等其他一些较为复杂的数据类型，这些将在本书后续章节介绍。

3.2 数据的运算

3.2.1 常量和变量

在程序中需要处理各种类型的数据，数据的形式可以是常量或变量。

1．常量

常量也叫常数，是指在程序执行期间其值不变的量，它可以是任何数据类型。在 VB.NET 中常量分为一般常量和符号常量。

（1）一般常量

一般常量包括：数值常量、字符常量、逻辑常量和日期常量。

先来看几个一般常量的实例：

数值常量（由正/负号、数字和小数点组成）：123，−265，−75.32。

字符常量（用双引号括起来）："abC"，"李明"，"你好！"。

逻辑常量（只有两个）：True（真），False（假）。

日期常量（用“#”括起来）：#23/8/2003#，#January 1, 1999#。

数值整数大多数都是十进制数（基数为 10），但有时也用十六进制数（基数为 16）或八进制数（基数为 8）表示。

各种数值整数表示的实例如下：

十进制数：如 123，−456，0。

八进制数：用前缀&O 表示，如&O123。

十六进制数：用前缀&H 表示，如&H123。

表 3.2 为十进制、八进制和十六进制整数的 5 个实例的对应关系表。

表 3.2　十进制、八进制和十六进制数的对应关系

十　进　制	八　进　制	十 六 进 制
9	&O11	&H9
15	&O17	&HF
16	&O20	&H10
20	&O24	&H14
255	&O377	&HFF

浮点数：用尾数、指数符号、指数来表示，用指数符号 E 表示单精度浮点数，用 D 表示双精度浮点数。

例如，123.45E−3 表示单精度浮点数，它的值为 123.45×10^{-3}；236D5 表示双精度浮点数，它的值为 236×10^{5}。

VB.NET 根据输入常量的形式来决定使用什么数据类型来保存它。在默认情况下，把整数常量作为 Integer 类型处理，把小数常量作为 Double 类型处理。

例如，根据输入值的形式来决定数据类型：

```
Var1 = 30          ' 此值为Integer类型
Var2 = 18.4        ' 此值为 Double 类型
Var3 = True        ' 此值为 Boolean 类型
```

为了显式地指定常量的类型，可以用 VB.NET 提供的值类型字符，方法是在值的后面加上类型字符。表 3.3 列出了值类型字符及用法示例。

表 3.3　值类型字符及用法示例

值类型字符	数 据 类 型	示　　例
C	Char	Var1 = "hello"C
S	Short	Var1 = 314 S
I	Integer	Var1 = 314 I
L	Long	Var1 = 314 L
D	Decimal	Var1 = 314 D
F	Single	Var1 = 314 F
R	Double	Var1 = 314 R

注意：表中未列出的其他数据类型，没有相应的值类型字符。

（2）符号常量

在 VB.NET 中，可以用定义的符号常量来代替数值或字符串。

定义符号常量的语法格式如下：

```
[Public|Private] Const 常量名[As 类型] = 表达式[ , 常量名[As 类型] = 表达式] …
```

其中：

① Public|Private，可选项。Public 表示所定义常量在整个项目中都可见；Private 表示所定义常量只在所声明的程序模块中可见。

② 常量名，必选项。代表所定义常量的符号名称，须符合 VB.NET 变量命名规则。

③ As 类型，可选项。类型可以是 VB.NET 支持的所有数据类型，每个常量都必须用单独的 As 子句，若省略，则默认为 Object 类型。

④ 表达式，必选项。“表达式”由文字常量、算术运算符（指数运算符 ^ 除外）、逻辑运算符组成，也可以使用如"Hello world"之类的字符串，但不能使用字符串连接符、变量、内部函数或用户定义的函数。

例如，下面两条语句分别定义了一个字符串常量和一个整型常量：

```
Public   Const   MyVar1   As   String = "hello"          ' 定义字符串常量
Private   Const   MyVar2   As   Integer = 18             ' 定义整型常量
```

也可以在一条语句中定义两个以上的符号常量，例如：

```
Public   Const   MyVar3   As   Integer = 18, Const   MyVar4   As   Double = MyVar3 + 13.2
```

除了使用“As 类型”子句来指定常量类型外，还可以用值类型字符（见表 3.3）或类型说明符（见表 3.6）来指定常量类型，例如：

```
Public   Const   PAI@ = 3.1415926                       ' 定义小数常量
Const   NIAN = 365L                                     ' 定义整数常量
```

上例中定义了一个符号常量 PAI，通过在其常量名后加上类型说明符@，表明定义了一个 Decimal 类型的常量，值为 3.1415926；另外还定义了一个符号常量 NIAN，通过在其值表达式后加上值类型字符 L，表明定义了一个 Long 类型的常量，值为 365。

2．变量

变量（Variable）是指在程序运行过程中其值可变的量，变量实际代表内存中特定的存储单元。变量具有名称和数据类型，名称表示内存的位置，通过名称访问变量中的数据；数据类型则决定了该变量的存储方式。

（1）变量命名

变量名表示变量所存储的数据的一个名称，通过名称来引用变量数据的值。给变量命名必须遵循下列规则：

① 变量名必须以英文字母开头，不能以数字或其他字符开头。例如，6ABC、$ABC 都是不合法的。但最后一个字符可以是类型声明字符（%、$、@、#、&、!）。

② 变量名只能由字母、数字或下划线（_）组成，不能包含句号（.）或空格。例如，ab.、a%b、How are you、A$B 等都是不合法的。

③ 变量名最长不能超过 255 个字符。

④ 变量名不能和 VB.NET 的关键字同名。例如，Or、If、Loop、Len、Abs、Mod 等都是关键字，不能作为变量名。另外，变量名也不能是末尾带有类型声明字符的关键字，如 Object 和 Object$都是非法的变量名。但可以把关键字嵌入变量名中，如变量 Object_Object 是合法的。

实际上，除变量名外，过程名、结构类型名、数组名、元素名和符号常量名等都必须遵循上述规则。

在 VB.NET 中，变量名的大小写是不敏感的，即编译器不区分变量名中字母的大小写。例如，counter 和 COUNTER 被解释为同一变量。为便于阅读，在给变量命名时，每个单词的第一个字母一般用大写，如 PrintText。

（2）命名约定

如果在一个程序中有许多变量，那么需要一种方式以知道变量包含什么类型的数据，比如是整数还是字符串。为了让程序员一眼就能从变量名看出变量的类型，对变量的命名应采用良好的约定。

命名约定不是强制性的，通常取决于程序员的命名习惯。目前最常用的命名约定是匈牙利标记法（Hungarian Notation），它用变量名的前三个小写字母表示数据类型，第 4 个字母大写，表示从这开始是变量的具有实际意义的名字。表 3.4 列出了本书建议的变量命名约定。

表 3.4　变量命名约定

数据类型	前　缀	示　例
Char	chr	chrChar
String	str	strName
Byte	byt	bytByte
Short	sho	shoShort
Integer	int	intAge
Long	lng	lngLong
Decimal	dec	decScore
Single	sng	sngSingle
Double	dbl	dblDouble
Boolean	bln	blnSex
Date	dat	datBirthday
Object	obj	objObject

（3）变量声明

任何变量都具有一定的数据类型，用户可以通过声明来定义不同类型的变量。变量的声明分为“显式声明”和“隐式声明”。

① 显式声明

显式声明是在变量使用之前，用 Dim、Public、Private、Static、Protected、Friend Protected、Friend、Shared 等关键字声明变量。其中 Dim 是最常用的声明变量语句，它可以声明一个或多个变量。Dim 语句的语法格式如下：

```
Dim 变量名 As 数据类型
Dim 变量名 1 As 数据类型 1 , 变量名 2 As 数据类型 2…
```

例如，下面两行语句，分别声明了一个 String 变量 strSomeString 和一个 Date 变量 datSomeDate：

```
Dim strSomeString As String
Dim datSomeDate As Date
```

在一个语句中也可以声明多个变量，而且类型可以不同。若变量是相同类型的，只需使用一个 As 子句。例如，下面两行语句中，第一行一次声明多个不同类型的变量，第二行则一次声明了多个同类（Char 类型）的变量：

```
Dim strX As String , datY As Date , intZ As Integer
Dim chrA , chrB , chrC As Char
```

变量允许具有初始值，可以在声明变量的同时对它初始化。例如，在下面代码中，声明了一个 Integer 变量、一个 Boolean 变量和一个 Object 变量，并分别将它们初始化为 80、True 以及新创建的 Label 类实例：

```
Dim intScore As Integer = 80
Private blnSex As Boolean = True
Protected objLabel As New Label
```

也可以在声明多个不同类型变量的同时对它们初始化。例如，在下面这条语句中，声明了一个 Integer 变量、一个 Boolean 变量，并分别将它们初始化为 80、True：

```
Dim intScore As Integer = 80 , blnSex As Boolean = True
```

但是，在声明多个同类型变量的同时不能对它们初始化。例如，下面两行语句是错误的：

```
Dim intScore = 80 , intAge = 20 As Integer
Dim intScore , intAge As Integer = 50
```

如果在声明时没有指定变量的初始值，VB.NET 将把它们初始化为相应数据类型的默认值。各种数据类型的默认初始值见表 3.5。

表 3.5 各种数据类型的默认初始值

数 据 类 型	默认初始值
Char 类型	二进制数 0
所有数值类型（包括 Byte）	0
所有引用类型（包括 Object、String 和数组）	Nothing
Boolean 类型	False
Date 类型	公元 1 年 1 月 1 日 12:00 AM

显式声明还可以使用 Static、Public、Private、Protected、Friend、Friend Protected、Shared 等关键字，它们声明变量的方法与 Dim 类似。