第1章 .NET平台与集成开发环境

内容概要

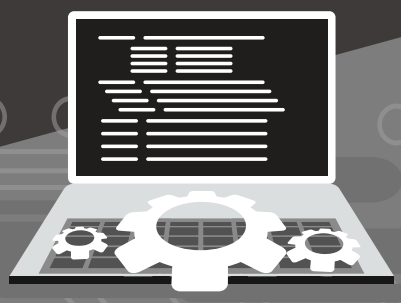
.NET 平台提供了一个多语言组件开发和执行的环境，它不但使得应用程序的开发和发布更加简单，并且实现了众多语言间的无缝集成。本章将介绍 .NET 平台的发展历程和构成，.NET 的三个实现平台 .NET Framework、Mono 和 .NET Core，并解释 .NET 程序的编译和执行过程，然后进一步介绍基于 .NET 平台的 Visual Studio 2019 集成开发环境。通过本章的学习，读者将会熟悉使用 C# 语言编程的环境基础，为下一步深入学习 Visual Studio 2019 集成开发环境下的 C# 语言编程奠定基础。

学习目标

- 了解 .NET 平台的构成
- 熟悉集成开发环境 Visual Studio 2019
- 熟悉 C# 与 .NET 的关系
- 掌握 .NET 程序的编译和执行

课时安排

- 理论学习　2 课时
- 上机操作　1 课时



1.1 .NET 平台概述

.NET 平台又称 .NET 框架（.NET Framework），是 .NET 的核心组成部分，提供了一个多语言组件开发和执行的环境，是一个完全可操控的、安全的和特性丰富的应用开发执行环境，这不但使得应用程序的开发和发布更加简单，并且实现了众多语言间的无缝集成。C# 语言是微软公司针对 .NET 平台推出的主流语言，它不但继承了 C++、Java 等面向对象语言的强大功能特性，同时还继承了 Visual Basic、Delphi 等编程语言的可视化快速开发功能，是当前第一个完全面向组件的语言。作为 .NET 平台的第一语言，C# 语言几乎集中了所有关于软件开发和软件工程研究的最新成果。本节主要介绍与 C# 语言密切相关的 .NET 平台和 Visual Studio 2019 集成开发环境。

1.1.1 .NET 平台简介

作为微软的集成开发平台，.NET 技术提供了迅速修改、部署、处理并且使用连接的能力，提高了 Web 服务的高效性；同时，.NET 技术也使创建稳定、可靠而又安全的 Windows 桌面应用程序更为容易。目前，.NET 已经成为一个庞大而复杂的软件开发与运行平台，包含“一堆”的子技术领域。

1. 桌面应用程序开发技术

在很长的一段时间，Windows Form 成为 .NET 桌面领域的主流技术，而且有一大批各式各样的第三方控件，其功能可谓应有尽有，使用方便。然而 .NET 3.0 中出现的 WPF，在界面设计和用户体验上比 Windows Form 要强得多，比如其强大的数据绑定、动画、依赖属性和路由事件机制等。WPF 的性能在 .NET 4.0 上有了进一步的改进。WPF 相对于 Windows 客户端的开发来说，向前跨出了巨大的一步，提供了超丰富的 .NET UI 框架，集成了矢量图形、丰富的流动文字支持 flow text support、3D 视觉效果和强大无比的控件模型框架。

2. 数据存取技术

.NET 平台融合了 ADO.NET、LINQ 和 WCF Data Service 等数据存取技术。ADO.NET 不仅提供了对 XML 的强大支持，还引入了一些新的对象，如驻于内存的数据缓冲区 DataSet、用来高效率读取数据并且只读的记录集 DataReader 等。LINQ (Language Integrated Query，语言集成查询) 是 Visual Studio 2008 和 .NET Framework 3.5 版本中引入的一项创新功能，它在对象领域和数据领域之间架起了一座桥梁。LINQ 是编程语言的一个组成部分，在编写程序时可以得到很好的编译时语法检查、丰富的元数据、智能感知、静态类型等强类型语言的好处。

3. Web 开发技术

.NET 平台底层使用 ADO.NET 实体框架或 LINQ to SQL 构造数据模型，通过提取数据模型中的元数据，动态选择合适的模板生成网页，避免了真实项目中不得不为每个数据存

取任务设计不同网页的负担，而且提供了很多方式允许用户定制网站。.NET 平台的另一种 Web 应用架构代表技术 Silverlight 充分利用客户端的计算资源，大大地降低了对服务端的依赖，并且易于构造良好的用户体验。

4. 插件技术

.NET 4.0 引入了 Managed Extensibility Framework(MEF) 技术。MEF 通过简单地给代码附加 [Import] 和 [Export] 标记就可以清晰地表明组件之间的“服务消费”与“服务提供”关系，并在底层使用反射动态地完成组件识别、装配工作，从而使得开发基于插件架构的应用系统变得简单。

5. 函数式编程语言 F#

F# 是微软 .NET 平台上一门新兴的函数式编程语言，通过 F#，开发人员可以轻松应对多核多并发时代的并行计算和分布问题。

.NET 发展至今，几乎可以在所有可用的平台上使用：从服务器、台式机到移动设备、游戏机、虚拟现实、增强现实环境、手表，甚至是像 Raspberri-Pi 等的小型嵌入式系统。整个框架以一种最开放的方式开源了代码，这也增加了其成为未来 UNIX 系统的核心组件到业界制造的新型秘密设备的可能。

1.1.2 .NET 平台的构成

众所周知，微软的灵魂产品 Windows 操作系统是硬件设备和软件运行环境的平台，它消除了不同硬件设备之间的差别，使外部设备都变成了可以自由使用、无缝集成的一个整体。与 Windows 操作系统类似，微软推出的 .NET 平台能够消除互联环境中不同硬件、软件、服务的差别，使不同的设备、不同的操作系统都可以相互通信，使不同的程序和服务之间都可以相互调用。.NET 平台几乎包含微软正在研发或已经得到广泛应用的各种软件开发技术。对 .NET 程序员来说，应主要关心 .NET 平台的以下几个组成部分。

- ◎ .NET 平台：微软推出的一种运行于各个操作系统之上的新的软件运行平台，提供了 .NET 程序运行时支持和功能强大的类库，是其他所有 .NET 技术产品的坚实基础。只要安装了 .NET Framework，则从 Windows 98 到 Windows XP 都可以运行 .NET 程序。
- ◎ .NET 编程语言：.NET 平台支持 20 多种编程语言，传统的各种编程语言有许多都已经或正在被移植到 .NET 平台。目前由微软公司提供的 .NET 编程语言主要有 Visual Basic.NET（改进过的 Visual Basic）、C++、C#、F#。
- ◎ Visual Studio.NET 集成开发环境：用来开发、测试和部署应用程序。Visual Studio.NET 历经微软公司持续多年的完善，已经成为世界一流的“软件集成开发环境（Integrated Development Environment，IDE）”。
- ◎ .NET 软件产品：微软公司几乎所有的软件产品都基于 .NET Framework 或包容 .NET 技术，包括 Windows 操作系统、SQL Server 数据库服务器、Office 商业应用开发与运行平台、Azure 云计算平台等。

1.1.3 .NET Framework、Mono 和 .NET Core

随着微软 .NET 开源的推进，现在 .NET 的实现上有了四个平台：.NET Framework、Mono、.NET Core 和 .NET Native，如图 1-1 所示。

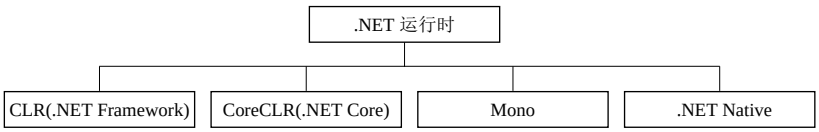


图 1-1 .NET 四个实现平台

1. .NET Framework

.NET Framework 是 .NET 平台的关键组件，提供了 .NET 程序运行时支持和功能强大的类库。从开发各种应用软件的程序员角度来看，.NET Framework 用易于理解与使用的面向对象方式调用 Windows 操作系统提供的各种功能。.NET Framework 在应用程序和操作系统之间起到承上启下的作用，向内包容着操作系统内核，向外给运行于其上的 .NET 应用程序提供访问操作系统核心功能的服务。在 .NET Framework 下编程，程序员不再需要与各种复杂的 Windows API 函数打交道，只需使用现成的 .NET Framework 类库即可。

.NET Framework 的体系结构如图 1-2 所示，主要由公共语言运行库（CLR，Common Language Runtime）和 .NET Framework 类库构成。

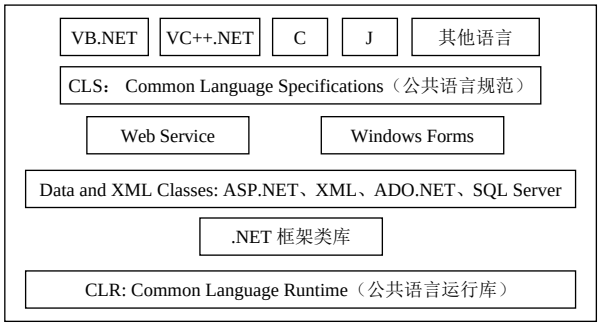


图 1-2 .NET Framework 的体系结构

CLR 是 .NET Framework 的核心执行环境，也称为 .NET 运行库。CLR 是一个技术规范，无论程序使用什么语言编写，只要能编译成 CIL 公共中间语言（最早称微软中间语言 MSIL），就可以在它的支持下运行。这意味着在不久的将来，可以在 Windows 环境下运行传统的非 Windows 语言。而 .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，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，该库可以完成以前要通过 Windows API 来完成的绝大多数任务。

2. Mono

Windows 平台下开发的 .NET 应用程序都依赖于 .NET Framework，而 .NET Framework 又依赖于 Windows 平台下的 CLR，因此，无法实现跨平台。在这种背景下，Mono 出现了，跨平台实现了 .NET Framework 的编译器、CLR 和基础类库。

Mono 是一个由 Xamarin 公司（先前是 Novell，最早为 Ximian）所主持的自由开放源代码项目。该项目的目标是创建一系列符合 ECMA 标准（Ecma-334 和 Ecma-335）的 .NET 工具，包括 C# 编译器和通用语言架构。与微软的 .NET Framework 不同，Mono 项目不仅可以运行于 Windows 系统上，还可以运行于 Linux、FreeBSD、UNIX、OS X 和 Solaris 系统上，甚至可以运行在一些游戏平台上，如 Playstation 3、Wii 和 XBox 360。近几年由于移动设备的兴起，Mono 的衍生项目（MonoTouch 和 Mono for Android）还可以让基于 .NET 框架的程序轻易地移植到 Android 和 iOS 设备上，这让原本 Windows 上的 C# 程序员上手移动开发的周期大大缩短。

Mono 通过内置的 C# 语言编译器、CLR 运行时和各种类库，可以使 .NET 应用程序运行在 Windows、Linux、FreeBSD 等不同的平台上。2019 年底，Mono 6.4 版本发布，引入了显式接口 bug。Mono 的体系结构如图 1-3 所示。Mono 通常与实时编译器一起使用，支持所有当前已发布的 .NET Standard 版本。Mono 实时编译引擎可以将代码实时编译或者预先编译到原生代码。对于那些没有列出来的系统，使用的是代码解释器。

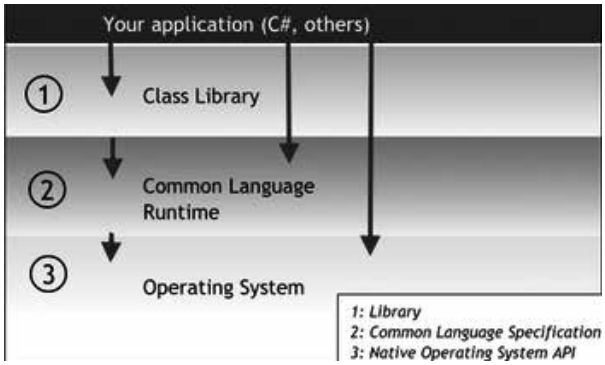


图 1-3　Mono 的体系结构

3. .NET Core

早在 2004 年，Novell 公司 (Xamarin 公司的前身) 就开始研发如何实现 .NET 的跨平台化应用。当时，Xamarin 公司的 Mono Project 开源项目陆续实现了在大部分 Linux 版本、Android 甚至一些游戏开发平台上运行 .NET 应用。2014 年，微软改变了 .NET 技术的发展策略，在 connect(); 大会上发布开源 .NET Core，称 .NET Core 的愿景是成为所有平台的单一代码库。2016 年微软收购了 Xamarin 公司，其所有的 Mono Project 采用 MIT 协议开源。

2016 年 6 月 27 日，微软在 Red Hat DevNation 大会上正式发布了 .NET Core 以及 ASP.NET Core 1.0，并提供了下载。.NET Core 是一个开源的、跨平台的 .NET 实现，是新一代 .NET 的基石，支持 Windows、Linux、macOS 和 Docker。

4. .NET Framework、Mono 和 .NET Core 三者的关系

.NET Core、.NET Framework 和 Mono 是新一代 .NET 平台的三大框架，.NET Core 是一个开源的、跨平台的 .NET 实现，而 .NET Framework 是基于 Windows 的 .NET 实现，Mono 是 .NET Framework 的一个开源、跨平台的实现。.NET Core 定位于跨平台服务端应用开发，.NET Framework 定位于 Windows 桌面应用开发，Mono (Xamarin-flavored Mono)

定位于移动应用开发。另一方面，.NET Framework、Mono、.NET Core 都是微软基于 .NET 标准库上的实现，具有共同的语言工具、运行时组件和构建工具，三者的区别和联系如图 1-4 所示。到目前为止，微软在 Windows 平台上的 .NET Framework 的实现最为完整，.NET Core 只完成了 .NET Framework 的 25% 的功能。

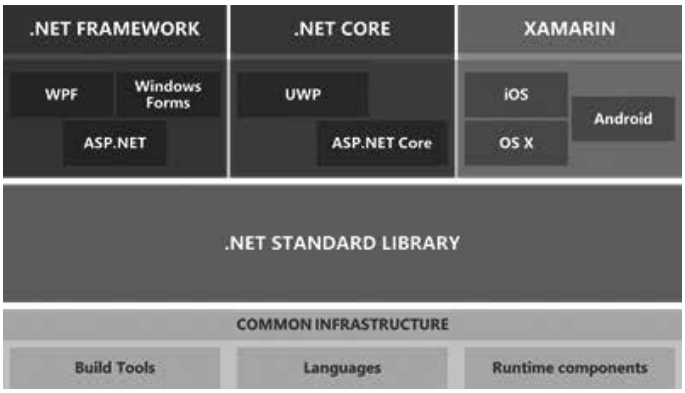


图 1-4　.NET Core、.NET Framework 和 XAMARIN

.NET Core 是第一个由微软官方所开发，具有跨出 Windows 平台能力的开发平台，作为下一代 .NET 开发的基石，同时它会和现有的 .NET Framework 并行，在 Windows 上可同时存有这两种执行环境。另外，.NET Core 的程序可存取 .NET Framework 的类别库以保有相容性 (仅限 Windows 系统)，.NET Core 本身的类别库也重新设计，改为以套件散布方式提供，也就是说，以 .NET Core 开发的应用程序不再需要传统大包装的 Framework Runtime，只需要执行前下载 Core CLR，并且在执行程序前先行还原套件就可以执行，套件的版本与各应用程序可各自独立。同时套件化的管理也保有版本相容性，应用程序可视需要抓取所需的版本，而不用固定在一个版本。这一重大进步使得以往写个小程序还要带打包 Framework 的景象将不再出现。

随着 .NET Core Framework 的开发完成，.NET Framework 与 Mono 将基于 .NET Core 重新构建。.NET Framework 将成为 .NET Core 在 Windows 上的一个发行版，Mono 将成为 .NET Core 的一个跨平台发行版。

1.1.4　.NET 程序的编译和执行

与传统的 Windows 应用程序相比，.NET 应用程序有很多不同的地方，尤其是在编译与执行期间。传统的 Windows 应用程序会被编译器直接编译成与特定机器相关的本地应用程序，这类程序只能在特定操作系统及硬件系统上运行，而 .NET 应用程序在编译时只会被编译成 CIL 中间代码，在运行期间被即时编译成本地指令，从而可达到跨平台的效果，使平台与上层软件完全隔离。由公共语言运行库 CLR（而不是直接由操作系统）执行的托管代码的编译和执行过程如图 1-5 所示，具体步骤如下。

1. 选择编译器

为获得公共语言运行库 CLR 提供的优秀运行能力，必须使用一个或多个针对 CLR 的

语言编译器，如 Visual Basic、C#、Visual C++、JScript 或许多第三方编译器（如 Eiffel、Perl 或 COBOL 编译器）中的某一个。

不同的编程语言往往使用不同的特定术语表达相同的程序构造，要想让不同的语言之间有最佳的相容性，以便互相调用或继承，这些面向 .NET 的语言编译器就需共同遵守 CLS。CLS 清晰地描述了支持 .NET 的编译器必须支持的最小和完全特征集，以便生成可由 CLR 承载的代码，被基于 .NET 平台的其他语言用统一的方式进行访问，以使由不同编程语言编写的程序能无缝地融入 .NET 世界。

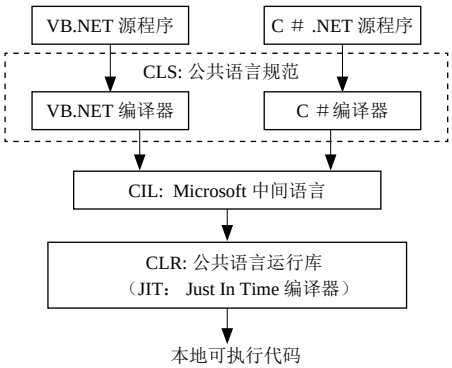


图 1-5 .NET 程序执行流程

2. 将代码编译为 CIL

将源代码翻译为 Microsoft 中间语言 CIL 并生成所需的元数据。

CIL 是一组可以有效地转换为本机代码且独立于 CPU 的指令，它不仅包括用于加载、存储和初始化对象以及调用对象方法的指令，还包括用于算术和逻辑运算、控制流、直接内存访问、异常处理和其他操作的指令。当编译器产生 CIL 时，它也产生元数据。元数据描述代码中的类型，包括每种类型的定义、每种类型成员的签名、代码引用的成员和执行时使用的其他数据。CIL 和元数据包含在一个可移植可执行（Portable Executable，PE）文件中，此文件基于并扩展过去用于可执行内容的已发布 Microsoft PE 和通用对象文件格式 (COFF)，使得操作系统能够识别公共语言运行库 CLR 映像。

3. 将 CIL 翻译为本机代码

在执行时，实时编译器 (JIT) 将 CIL 翻译为本机代码。

要使代码可运行，必须先将 CIL 转换为特定于 CPU 的代码，这通常是通过实时编译器 (JIT) 来完成的。由于公共语言运行库 CLR 为它支持的每种计算机结构都提供了一种或多种 JIT 编译器，因此同一组 CIL 可以在所支持的任何结构上 JIT 编译和运行。JIT 编译考虑了在执行过程中某些代码可能永远不会被调用的可能性，它不是耗费时间和内存将 PE 文件中的所有 CIL 都转换为本机代码，而是在执行期间根据需要转换 CIL 并将生成的本机代码存储在内存中，以供该进程上下文中的后续调用访问。

在编译为本机代码的过程中，CIL 代码必须通过验证过程，检查 CIL 和元数据以确定代码是否是类型安全的。类型安全将帮助对象彼此隔离，因而可以保护它们免遭无意或恶意的破坏。

4. 运行代码

公共语言运行库 CLR 提供使执行能够发生以及可在执行期间使用的各种服务结构。在执行过程中，托管代码接收若干服务，这些服务涉及垃圾回收、安全性、与非托管代码的互操作性、跨语言调试支持、增强的部署以及版本控制支持等。

1.1.5 C# 与 .NET

C# 是 Microsoft 专门为 .NET 平台创建的、用于开发运行在公共语言运行库 CLR 上的应用程序的语言之一。虽然 C# 本身并不是 .NET 的一部分，但是由于 C# 语言是和 .NET 平台一起使用的，如果要使用 C# 高效地开发应用程序，理解 .NET 非常重要。.NET 为 C# 提供了一个强大的、易用的、逻辑结构一致的程序设计环境。在 .NET 运行库的支持下，.NET 框架的各种优点在 C# 中表现得淋漓尽致。

C# 具有如下特点。

◎ 语法简洁。

◎ 面向对象设计。

◎ 与 Web 紧密结合。

◎ 完整的安全性和错误处理。

◎ 兼容性。

◎ 灵活性。

C# 是 .NET 公共语言运行环境的内置语言，符合 .NET CLR 中的公共语言运行规范。由 C# 编写的所有代码总是在 .NET 平台上运行，因此，C# 代码可以从公共语言运行库的服务中获益。C# 与 .NET 的密切关系反映在以下两个方面。

◎ C# 的结构和方法论反映了 .NET 基础方法论。

◎ 在许多情况下，C# 的特定语言功能取决于 .NET 的功能，或依赖于 .NET 基类。

1.2 集成开发环境 Visual Studio

目前，C# 编程语言最成熟的开发环境仍然是 Microsoft Visual Studio 集成开发环境。Visual Studio 是由微软自行研发的一个功能强大的可自定义编程系统，可以利用它所包含的各种工具快速有效地开发功能强大的 Windows 和 Web 程序。C# 最新的开发环境 Visual Studio 2019(简称 VS 2019) 可在官方网站免费下载。

Visual Studio 2019 是一套完整的开发工具集，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 服务以及其他类型 (如移动设备) 的应用程序。此外，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。

1.2.1 启动 Visual Studio 开发环境

选择【开始】|【所有程序】| Microsoft Visual Studio 2019 命令，启动 Visual Studio 2019 开发环境，即出现如图 1-6 所示的 Visual Studio 2019 集成开发环境起始页。

图 1-6 VS 2019 起始页

开发人员要使用 Visual Studio 2019 IDE 创建应用程序，可以在如图 1-6 所示的界面中选择【文件】|【新建项目】菜单命令，或者直接选择【新建项目】选项，Visual Studio 2019 将弹出如图 1-7 所示的【创建新项目】对话框。

在【所有语言】下拉列表框中选择 C# 选项，在【所有平台】下拉列表框中选择 Windows 选项，【所有项目类型】下拉列表框中选择【桌面】选项，并从出现的项目类型中选择【Windows 窗体应用 (.NET Framework)】（当然也可以选择其他类型），如图 1-8 所示，单击【下一步】按钮，弹出如图 1-9 所示的【配置新项目】对话框，最后单击【创建】按钮，弹出如图 1-10 所示的 Visual Studio 2019 默认开发主界面，就完成了项目的创建。

图 1-7 【创建新项目】对话框

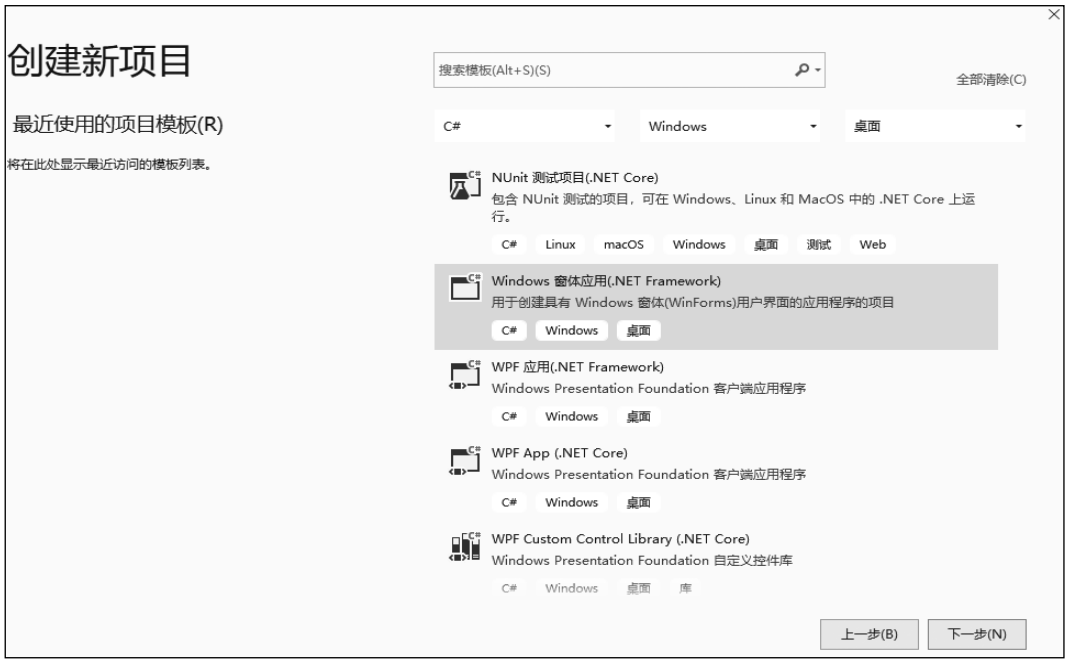


图 1-8　选择【Windows 窗体应用】（.NET Framework）类型

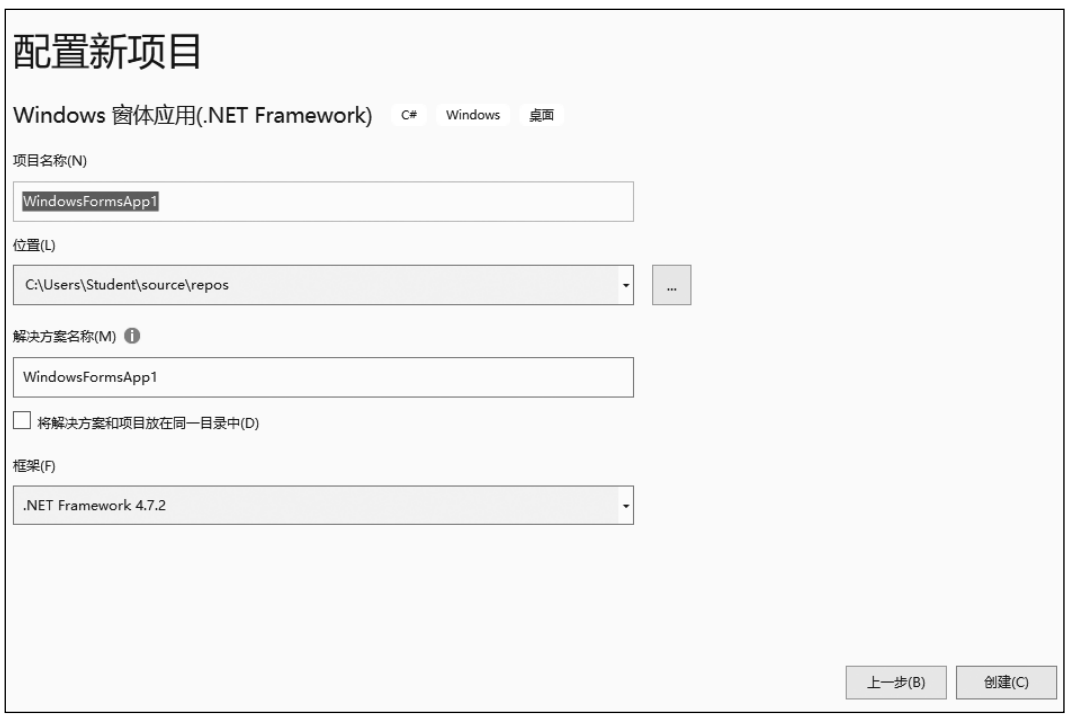


图 1-9　【配置新项目】窗口

1.2.2 Visual Studio 主窗口

Visual Studio 2019 集成开发环境将代码编辑器、编译器、调试器、图形界面设计器等工具和服务集成在一个环境中，能够有效提高开发软件的效率。在 Visual Studio 2019 集成开发环境中开发的每一个程序集对应一个项目（Project），而多个相关的项目又可以组成一个解决方案（Solution）。创建项目后打开的如图 1-10 所示的默认主界面主要包括以下几个部分。

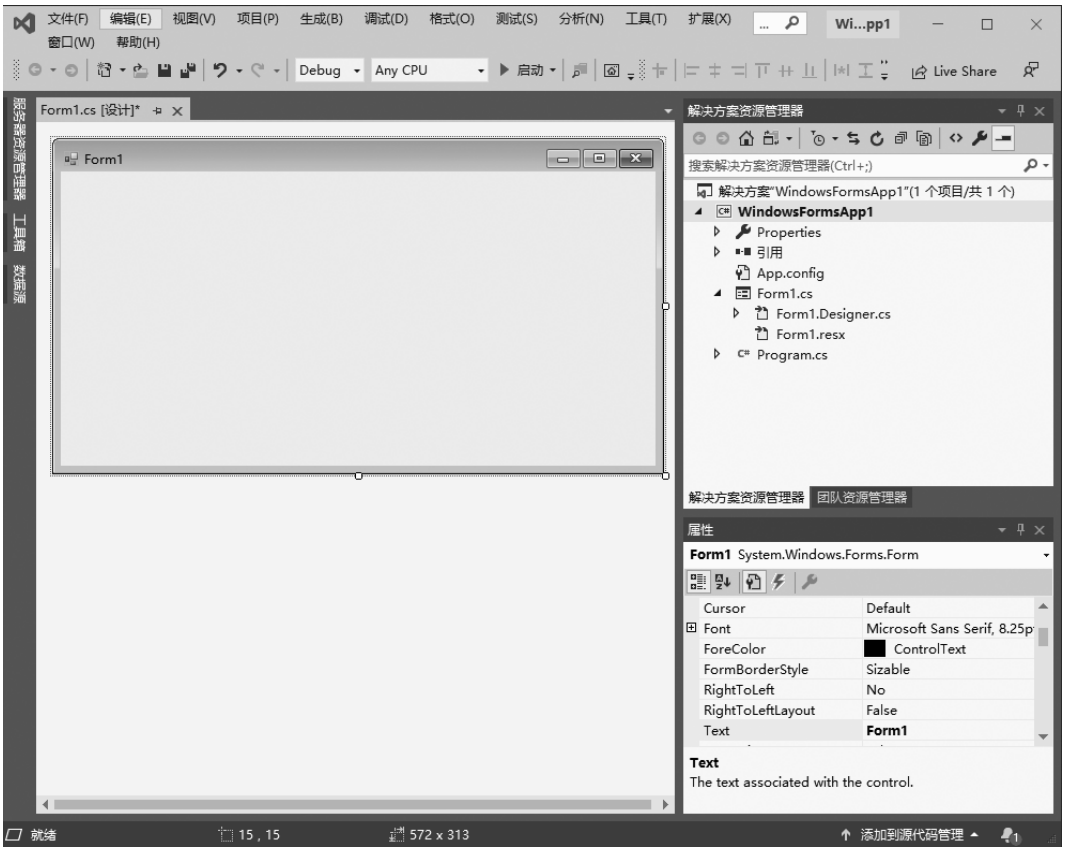


图 1-10　VS 2019 主界面

1. 菜单栏

位于标题栏的下方，其中包含用于开发、维护、编译、运行和调试程序以及配置开发环境的各项命令。Visual Studio 2019 菜单栏中所有可用的命令既可以通过鼠标单击执行，也可以通过 Alt+ 相应字母快捷键执行。其中，【文件】、【编辑】和【视图】是三个比较常用的主菜单。

2. 工具栏

位于菜单栏的下方，提供了常用命令的快捷方式。为了操作更方便、快捷，Visual

Studio 2019 常用的菜单命令按功能分组，被分别放入相应的工具栏中。根据当前窗体的不同类型，工具栏会动态改变。工具栏包括【布局】、【标准】、【数据设计】、【格式设置】、【生成】、【调试】、【文本编辑器】等选项，可通过【视图】|【工具栏】中的菜单命令打开或关闭工具栏选项，默认打开的工具栏选项有【标准】工具栏和【布局】工具栏。

3. 工具箱

工具箱以选项卡的形式来分组显示常用控件，包括公共控件、容器、数据等工具的集合，如图 1-11 所示。当需要某个控件时，可以通过双击所需控件直接将其添加到窗体上；也可以先单击选择需要的控件，再将其拖曳到设计窗体上。工具箱中的控件可以通过工具箱右键快捷菜单来进行排序、删除、设置显示方式等。

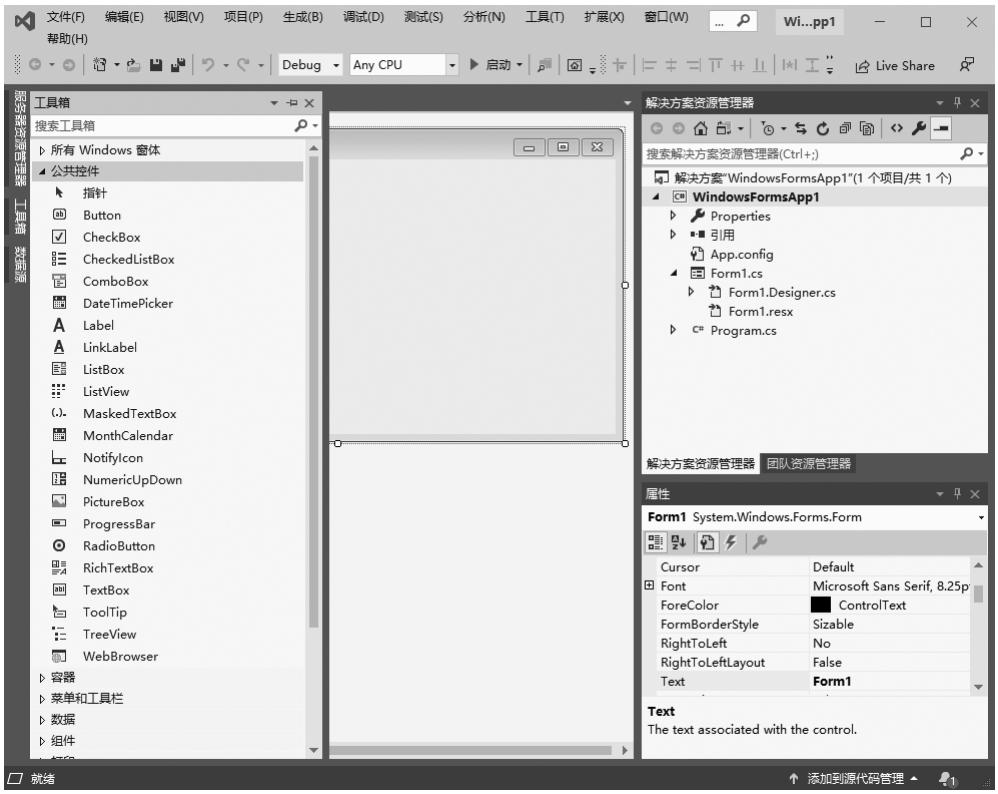


图 1-11　工具箱

4. 工作区

位于开发环境中央，用于具体项目的开发，如设计界面各控件的整体布局，事件代码的编写等。新建项目时，Visual Studio 2019 会自动添加一个窗体设计界面，如图 1-10 所示，可以根据需要把工具箱中的控件加入窗体中设置用户界面，此时，Visual Studio 2019 会自动在源文件中添加必要的 C# 代码，并在项目中实例化这些控件 (在 .NET 中，所有的控件

实际上都是特定基类的实例)。在窗体中的任意位置右击，在弹出的快捷菜单中选择【查看代码】命令，即可切换到如图 1-12 所示的窗体代码编辑窗口。

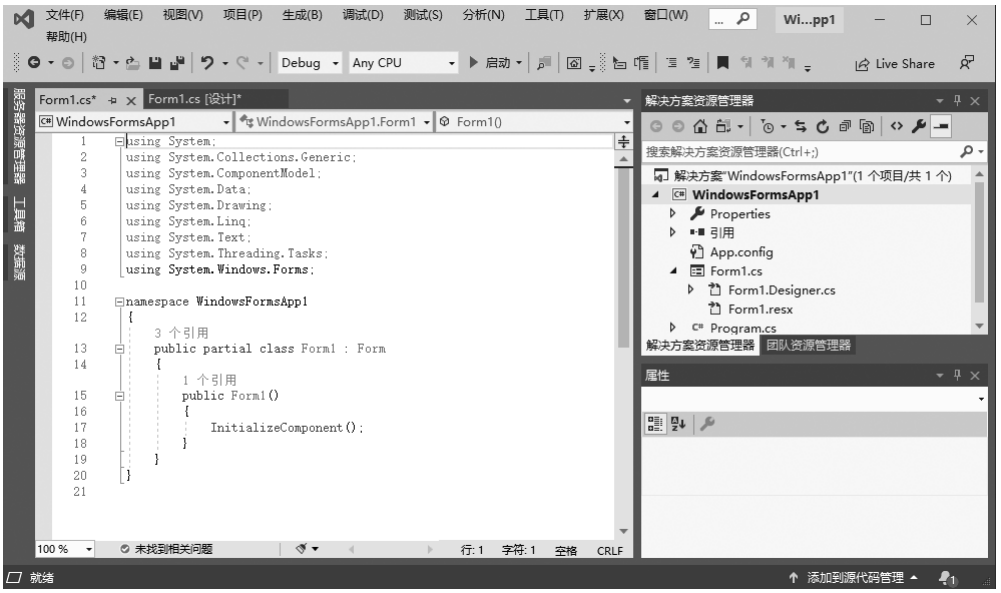


图 1-12　工作区中的代码编辑窗口

在代码编辑窗口中，程序员可以编写 C# 代码。代码编辑窗口的功能相当复杂，例如，在输入语句时，可以自动布局代码，方法是缩进代码行、匹配代码块的左右花括号等。同时，在输入语句时，还能执行一些语法检查，给可能产生编译错误的代码加上下划线，这也称为设计期间的调试。另外，代码编辑窗口还提供了 IntelliSense（智能感知）功能。在开始输入时，IntelliSense 会自动显示类名、字段名或方法名。在开始输入方法的参数时，IntelliSense 也会显示可用的重载方法的参数列表。当 IntelliSense 列表框因某种原因不可见时，可以按快捷键 Ctrl+Backspace 打开。

5. 解决方案资源管理器

位于开发环境的右侧，如图 1-12 所示，通过树形视图对当前解决方案进行管理。解决方案是树的根节点，解决方案中的每一个项目都是根节点的子节点，项目节点下则列出了该项目中使用的各种文件、引用和资源。

6. 状态栏

位于开发环境的底部，用于对光标位置、编辑方式等当前状态给出提示。

7. 错误列表

位于工作区的下方，用于输出当前操作的错误信息，如图 1-13 所示。如果主窗口中没有显示错误列表，可通过【视图】|【错误列表】命令打开错误列表。

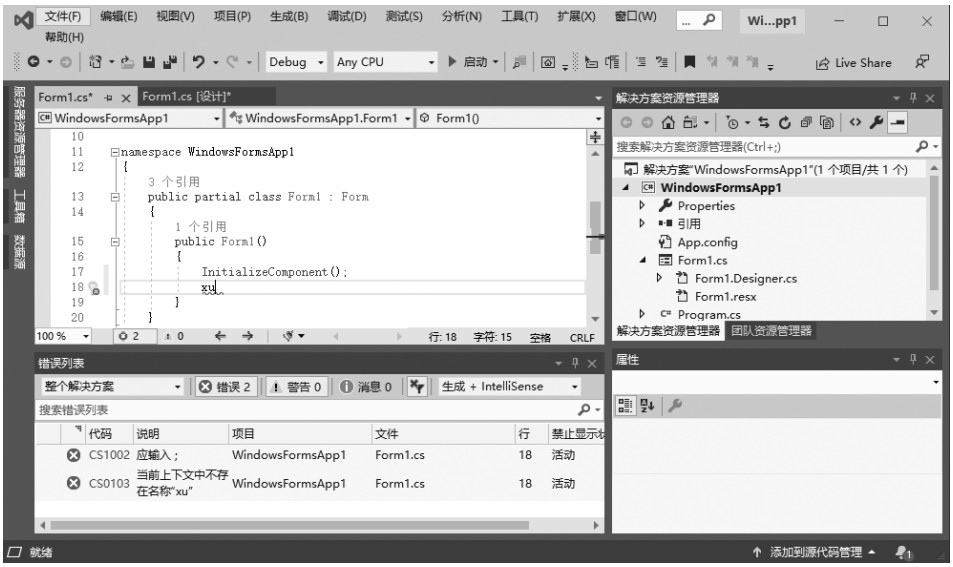


图 1-13　错误列表窗口

8. 服务器资源管理器

位于开发环境的左侧，用于快速访问本地或网络上的各项服务器资源。如果主窗口中没有显示服务器资源管理器，可通过【视图】|【服务器资源管理器】命令打开服务器资源管理器。

9. 【属性】窗口

位于解决方案资源管理器的下方，如图 1-11 所示，用于查看或编辑当前所选元素的具体信息。窗体应用程序开发中的各个控件属性都可以通过【属性】窗口来设置；此外，【属性】窗口还提供了针对控件的事件的管理功能，方便编程时对事件的处理。如果主窗口中没有显示【属性】窗口，可通过【视图】|【属性窗口】命令打开【属性】窗口。

其他常用的窗口还有管理程序中的类及其关系的类视图、显示当前操作输出结果的输出窗口等。用户可以根据需要来移动、调整、打开、关闭它们，或是通过【视图】菜单来控制它们的显示。大部分窗口还可以通过选项卡的方式切换，如代码编辑区可一次打开多个源文件，以便能最大限度地利用有限的屏幕空间。

1.2.3　Visual Studio 帮助系统

Visual Studio 中提供了一个广泛的帮助工具，包含对 C# 语言各方面知识的讲解，用户可以在其中查看任何 C# 语句、类、属性、方法、编程概念及一些编程的例子。在 Visual Studio 2019 菜单栏中选择【帮助】|【技术支持】命令，弹出如图 1-14 所示的联机帮助主界面。

联机帮助实际上就是 .NET 语言的超大型词典，用户可以在该词典中查找 .NET 语言的结构、声明以及使用方法。联机帮助还是一个智能的查询软件，它为使用者提供了一种强大的搜索功能。在如图 1-14 所示的联机帮助主界面中单击工具栏中的 Search 按钮，并在文

本框中输入搜索的内容提要，按 Enter 键后，搜索的结果将以概要的方式呈现在主界面中，开发人员可以根据自己的需要选择不同的文档进行阅读，如图 1-15 所示。

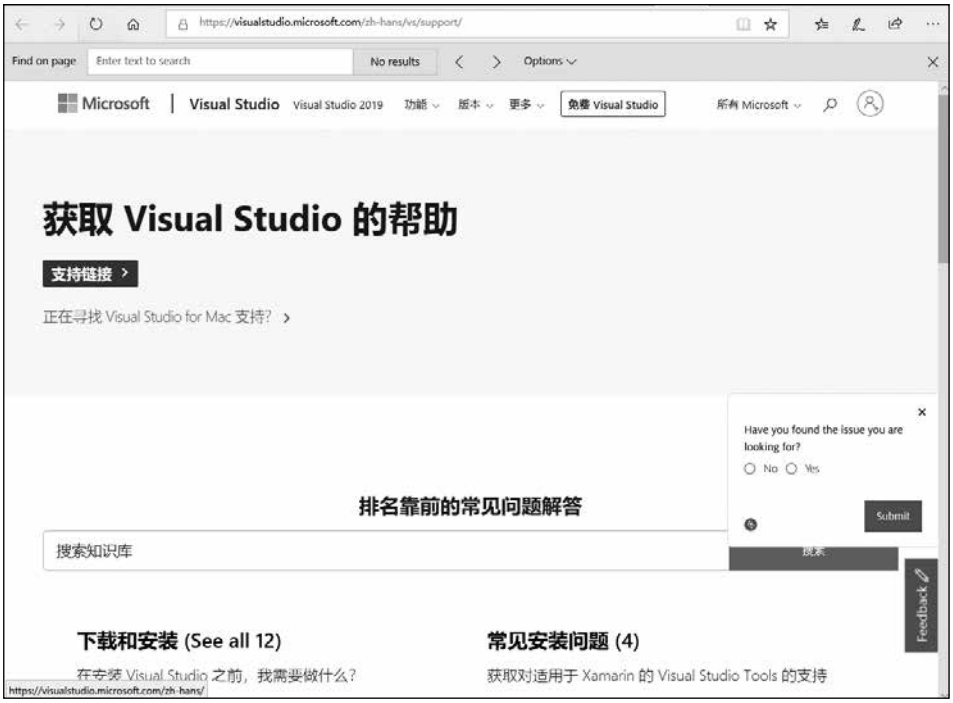


图 1-14　联机帮助主界面

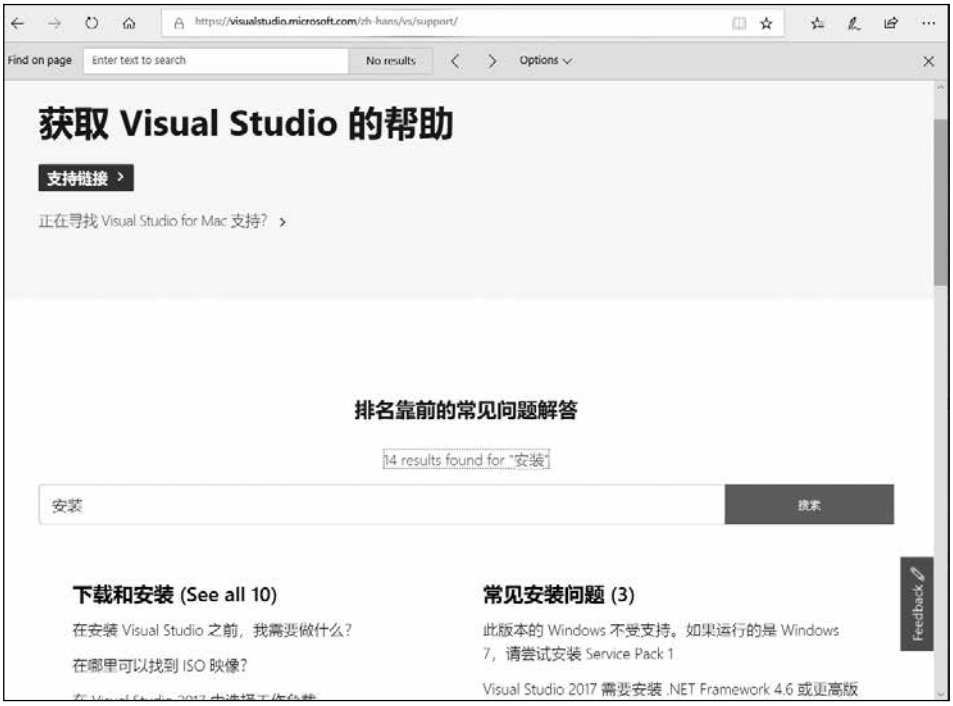


图 1-15　联机帮助的【搜索】功能

强化练习

本章主要介绍了 .NET 平台的发展历程、.NET 平台的构成、.NET 程序的编译和执行、集成开发环境 Visual Studio 2019 的启动和主要组成。在此基础上，重点介绍了 .NET 程序的执行流程、.NET Framework 的体系结构以及如何通过集成开发环境 Visual Studio 2019 创建项目。熟练掌握 .NET 程序的执行流程和 .NET Framework 的体系结构后，读者可以自行练习以下操作，亲身体验通过集成开发环境 Visual Studio 2019 编程的乐趣。

练习：创建一个项目，熟悉 Visual Studio 2019 主窗口的构成。