第 3 章

C# 项目初步构建

3.1 Microsoft Visual Studio 环境介绍

Microsoft Visual Studio 是一套完整的开发工具集，用户可以利用它生成 ASP. NET Web 应用程序、XML WebServices、桌面应用程序和移动应用程序。它提供了在设计、开发、调试和部署 Web 应用程序、XML WebServices 和传统的客户端应用程序所需的工具。

3.1.1 创建项目

视频 3 项目初步构建

创建项目的过程非常简单，首先要启动 Microsoft Visual Studio 开发环境。选择“开始”→“所有程序”→Visual Studio 2019 命令，即可进入 Microsoft Visual Studio 开发环境。界面如图 3-1 所示。

图 3-1 启动界面

第一次启动时显示如图 3-2 所示对话框，在其中选择 Visual C＃开发设置。

图 3-2　开发环境设置

提示：如果选错了，按以下方法更改：单击“工具”→“导入和导出设置”→“重置所有设置”命令。

Microsoft Visual Studio 的起始页界面如图 3-3 所示。

图 3-3　起始页

启动 Microsoft Visual Studio 开发环境之后，可以通过两种方法创建项目：一种是选择“文件”→“新建”→“项目”命令，如图 3-4 所示；另一种是在“起始页”中选择“创建新项目”命令，如图 3-5 所示。

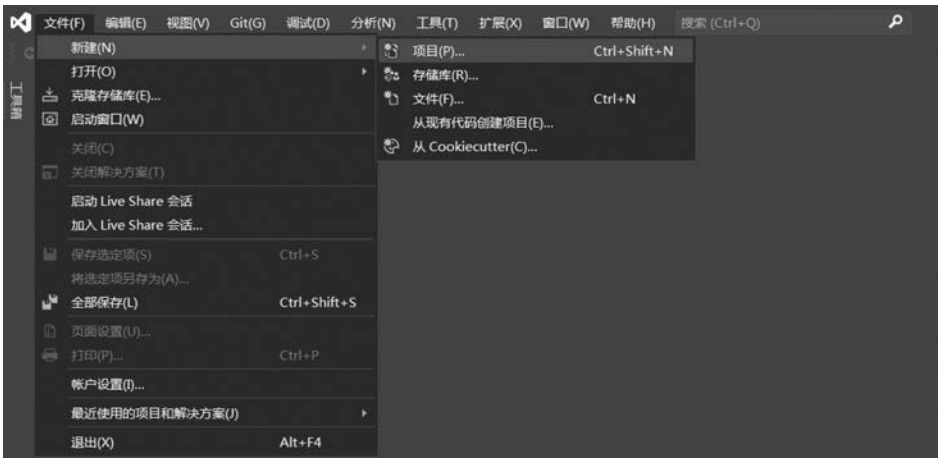

图 3-4　新建项目 1

图 3-5　新建项目 2

单击如图 3-5 中的“创建新项目”按钮，弹出如图 3-6 所示的“创建新项目”对话框。

选择创建“Windows 窗体应用程序”后，如图 3-7 所示用户可以对新建的项目进行命名、选择保存的位置、是否创建解决方案目录的设定。在命名时可以使用用户自定义的名称，也可使用默认名，用户可以单击“浏览”按钮设置项目保存的位置。需要注意的是，解决方案名称与项目名称一定要统一，然后单击“创建”按钮，完成项目的创建。

图 3-6　新建项目 3

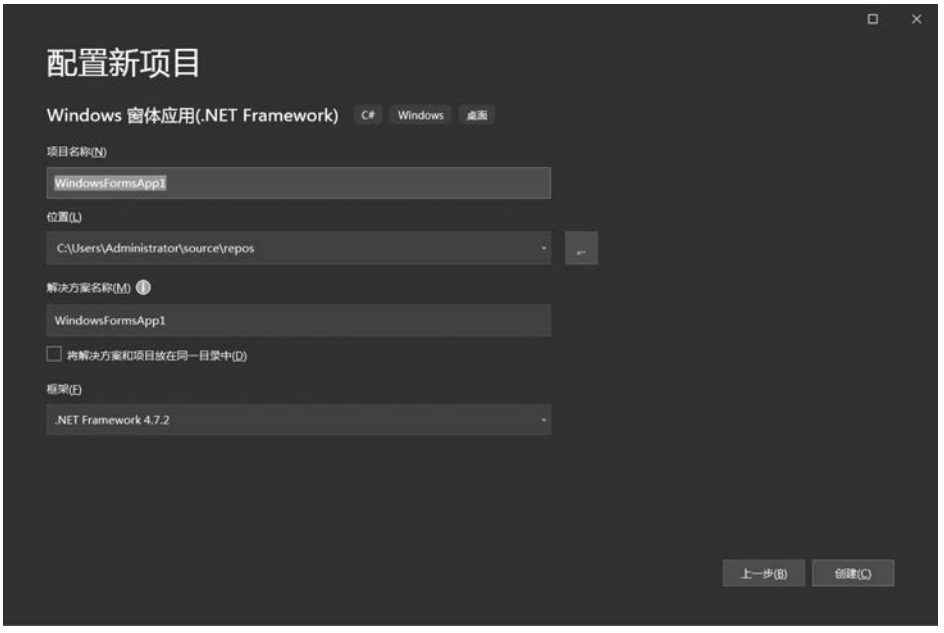

图 3-7　项目名称和选择保存位置

3.1.2　菜单栏

菜单栏显示了所有可用的命令。通过单击鼠标可以执行菜单命令，也可以通过 Alt 键加上菜单项对应的字母执行菜单命令。部分菜单命令及其作用如表 3-1 所示。

表 3-1　菜单栏功能表

菜　单　项	菜 单 命 令	作　　用
文件	新建	建立一个新的项目、网站、文件等
	打开	打开一个已经存在的项目、文件等
	关闭	关闭当前页面
	关闭解决方案	关闭当前解决方案
	全部保存	将项目中所有文件保存
	导出模板	将当前项目作为模板保存起来，生成.zip 文件
	页面设置	设置打印机及打印属性
	打印	打印选择的指定内容
	最近的文件	打开最近操作的文件（如类文件）
	最近使用的项目和解决方案	打开最近操作的文件（如解决方案）
	退出	退出集成开发环境
编辑	撤销	撤销上一步操作
	重做	重做上一步所做的修改
	撤销上次全局操作	撤销上一步全局操作
	重做上次全局操作	重做上一步所做的全局修改
	剪切	将选定内容放入剪贴板，同时删除文档中所选的内容
	复制	将选定内容放入剪贴板，但不删除文档中所选的内容
	粘贴	将剪贴板中的内容粘贴到当前光标处
	删除	删除所选定内容
	全选	选择当前文档中全部内容
	查找和替换	在当前窗口文件中查找指定内容，可将查找到的内容替换为指定信息
	转到	选择定位到“结果”窗格的那一行
	书签	显示书签功能菜单
视图	代码	显示代码编辑窗口
	设计器	打开设计器窗口
	服务器资源管理器	显示服务器资源管理器窗口
	解决方案资源管理器	显示解决方案资源管理器窗口
	类视图	显示类视图窗口
	代码定义窗口	显示代码定义窗口
	对象浏览器	显示对象浏览器窗口
	错误列表	显示错误列表窗口
	输出	显示输出窗口
	属性窗口	显示属性窗口
	任务列表	显示任务列表窗口
	工具箱	显示工具箱窗口
	查找结果	显示查找结果
	其他窗口	显示其他窗口（如命令窗口、起始页等）
	工具栏	打开工具栏菜单（如标准工具栏、调试工具栏）
	全屏显示	将当前窗体全屏显示
	向后导航	将控制权移交给下一任务
	向前导航	将控制权移交给上一任务
	属性页	为用户控件显示属性页

3.1.3 工具栏

为了使操作更方便、快捷，菜单项中常用的命令按功能分组分别放入相应的工具栏。通过工具栏即可迅速访问它们。常用的工具栏有标准工具栏和调试工具栏，界面如图 3-8 所示。下面分别进行介绍。

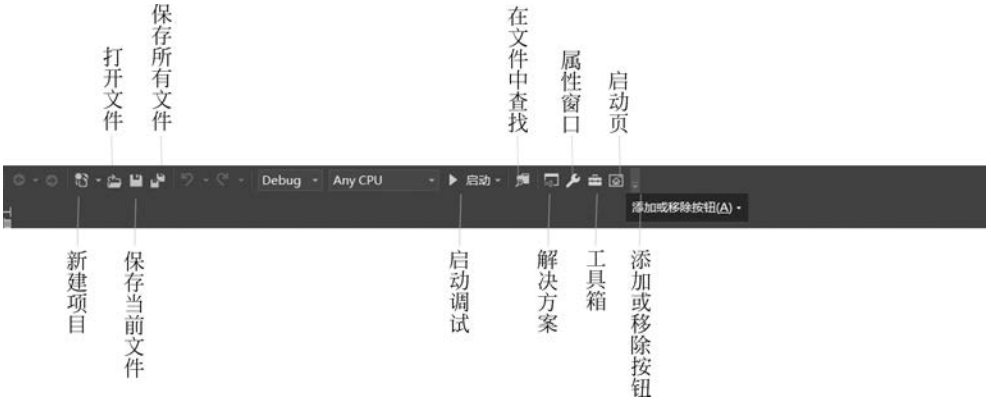

图 3-8 工具栏

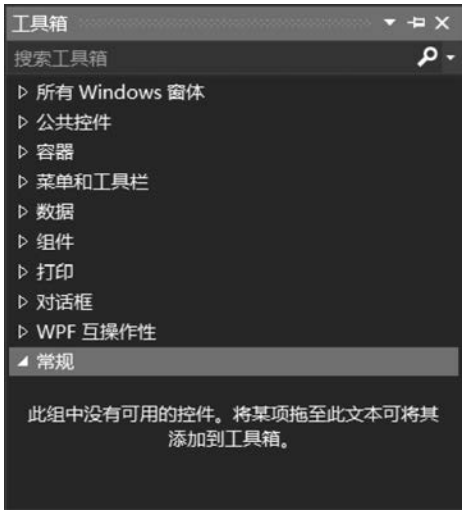

图 3-9 工具箱

3.1.4 工具箱

工具箱是 Microsoft Visual Studio 的重要工具，每个开发人员都必须对这个工具非常熟悉。工具箱提供了进行 Windows 窗体应用程序开发所必需的控件。通过工具，开发人员可以方便地进行可视化的窗体设计，简化了程序设计的工作量，提高了工作效率。

工具箱中包含了建立应用程序的各种控件以及非图形化的组件。工具箱由不同的选项卡组成，各类控件、组件分别放在“所有 Windows 窗体”“公共控件”“数据”“组件”“常规”等选项卡下面，如图 3-9 所示。

3.1.5 属性

在“属性”面板中可查看控件、类、项目的属性。窗口的左边显示属性的名称，右边显示相对应的属性，底部显示所选属性的说明信息，如图 3-10 所示。

“属性”面板采用了两种方式管理属性和方法，分别为按类型方式和按字母排序方式。可以根据习惯采用不同的方式。

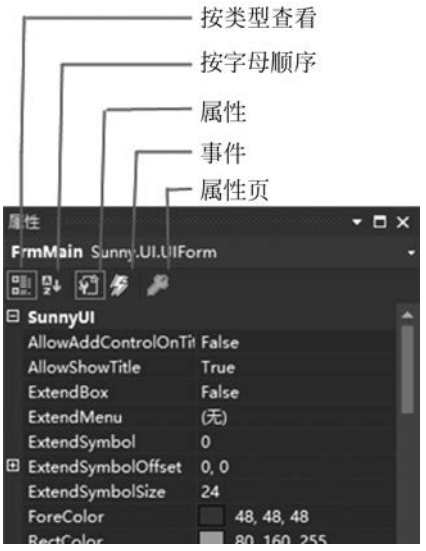

图 3-10　“属性”面板

3.2　认识 C# 项目

3.2.1　案例描述

在 C# 项目中包含了很多的项目文件，我们可以在项目的源文件中写出自己想要实现的程序，如图 3-11 所示。单击“运行”按钮即可运行 C# 项目，如图 3-12 所示。下面通过一个案例来认识一下 C# 项目。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Exam
{
    0 个引用
    class Program
    {
        0 个引用
        static void Main(string[] args)
        {
            Console.WriteLine("开始学习C#课程了！");
            Console.ReadKey();
        }
    }
}
```

图 3-11　案例图

图 3-12　运行结果

3.2.2　知识引入

1. C# 程序结构

C# 程序的结构大体可以分为注释、命名空间、类、Main 方法、标识符、关键字和语句。下面对 C# 程序的结构进行详细介绍。

1）注释

编译器编译程序时不执行注释的代码或文字，其主要功能是对某行或某段代码进行说明，以方便对代码的理解与维护。这一过程就好像是超市中各商品的下面都附有价格标签，对商品的价格进行说明。注释可以分为行注释和块注释两种，行注释都以“//”开头。

如果注释的行数较少，一般使用行注释。对于连续多行的大段注释，则使用块注释，块注释通常以“/*”开始，以“*/”结束，注释的内容放在它们之间。例如：

```
/*程序主入口*/
//主程序入口
```

注意：注释可以出现在代码的任意位置，但是不能分隔关键字和标识符。

2）命名空间

C#程序是利用命名空间组织起来的。命名空间既用作程序的“内部”组织系统，也用作向“外部”公开的组织系统（即一种向其他程序公开自己拥有的程序元素的方法）。如果要调用某个命名空间中的类或者方法，首先需要使用 using 指令引入命名空间，using 指令将命名空间名所标识的命名空间内的类型成员导入当前编译单元中，从而可以直接使用每个被导入的类型的标识符，而不必加上它们的完全限定名。

C#中的各命名空间就好像是一个存储了不同类型的仓库，而 using 指令就好比是一把钥匙，命名空间的名称就好比仓库的名称，可以通过钥匙打开指定名称的仓库，从而在仓库中获取所需的物品。

using 指令的基本形式为

```
using 命名空间名;
```

如：namespace HelloWorld。

提示：

（1）同一个命名空间是指逻辑上属于一个范围，物理上的存储不一定要相同。

（2）用户也可以在项目的命名空间中定义命名空间，只是这样定义，不能用 using 来引用自定义的命名空间。

3）类

类是一种数据结构，它可以封装数据成员、函数成员和其他的类。类是创建对象的模板。C#中所有的语句都必须位于类内。因此，类是C#语言的核心和基本构成模块。C#支持自定义类，使用C#编程就是编写自己的类来描述实际需要解决的问题。

类就好比一个企业有各个不同的部门，如行政部、后勤部、研发部、秘书室和人事部等，在各个部门中都有自己的工作方法，相当于在类中定义的变量、方法等。如果要完成一个重要的工作，仅靠一个部门是不行的，可能要研发部、市场部等多个部门一起配合才可以顺利完成，这时可以让这几个部门临时组成一个工作小组，对项目进行研发，这个小组就相当于类的继承，也就是该小组可以动用这几个部门中的所有资源和设备。

使用任何新的类之前都必须声明它，一个类一旦被声明，就可以当作一种新的类型来使用。在C#中通过使用class关键字来声明类，声明形式如下：

```
[类修饰符] class [类名] [基类或接口]
[类体]
```

提示：在C#中，类名首字母需大写。如：class Program、class Student。

4）Main方法

Main方法是程序的入口点，C#程序中必须包含一个Main方法，在该方法中可以创建对象和调用其他方法，一个C#程序中只能有一个Main方法，并且在C#中所有的Main方法都必须是静态的。C#是一种面向对象的编程语言，Main方法既是程序的启动入口点，也是一个类的成员。由于程序启动时还没有创建类的对象，因此，必须将入口点Main方法定义为静态方法，使它可以不依赖于类的实例对象而执行。

Main方法就相当于汽车的电瓶，在生产汽车时，将各个零件进行组装，相当于程序的编写。当汽车组装完成后，就要检测汽车是否可用，如果想启动汽车，就必须通过电瓶来启动汽车的各个部件，如发动机、车灯等，电瓶就相当于启动汽车的入口点。

可以用3个修饰符修饰Main方法，分别是public、static和void。

- public：说明Main方法是共有的，在类的外面也可以调用整个方法。
- static：说明方法是一个静态方法，即这个方法属于类的本身，而不是这个类的特定对象。调用静态方法不能使用类的实例化对象，必须直接使用类名来调用。
- void：此修饰符说明方法无返回值。

5）标识符及关键字

标识符是指在程序中用来表示事物的单词，例如，System命名空间中的类Console，以及Console类的方法WriteLine都是标识符。标识符的命名有3个基本规则，分别介绍如下。

- 标识符只能由数字、字母和下画线组成。
- 标识符必须以字母或者下画线开头。
- 标识符不能是关键字。

注意：在对类、变量、方法等进行命名时，不要与标识符和关键字重名。

所谓关键字，是指在C#语言中具有特殊意义的单词，它们被C#设定为保留字，不能随意使用。例如，在“Hello World!”程序中的static和void都是关键字。

6）C#语句

语句是构造所有C#程序的基本单位。语句可以声明局部变量或常数、调用方法、创建对象或将值赋给变量、属性或字段，语句通常以分号终止。

2. 程序编写规范

下面的内容中将详细介绍代码的书写规则以及命名规范，使用代码书写规则和命名规范可以使程序代码更加规范化，对代码的理解与维护起到至关重要的作用。

1）代码书写规则

代码书写规则通常对应用程序的功能没有影响，但它们对于改善源代码的理解是有帮

助的。养成良好的习惯对于软件的开发和维护都是很有益的。下面介绍一些代码书写规则。

(1) 尽量使用接口，然后使用类实现接口，以提高程序的灵活性。

(2) 一行不要超过 80 个字符。

(3) 尽量不要手工更改计算机生成的代码，若必须更改，一定要改成和计算机生成的代码一样的风格。

(4) 关键的语句(包括声明关键的变量)必须要写注释。

(5) 建议局部变量在最接近使用它的地方声明。

(6) 不要使用 goto 系列语句，除非是用在跳出深层循环时。

(7) 避免写超过 5 个参数的方法。如果要传递多个参数，则使用结构。

(8) 避免书写代码量过大的 try…catch 模块。

(9) 避免在同一个文件中放置多个类。

(10) 生成和构建一个长的字符串时，一定要使用 StringBuilder 类型，而不用 string 类型。

(11) switch 语句一定要有 default 语句来处理意外情况。

(12) 对于 if 语句，应该使用一对“{}”把语句块包括起来。

(13) 尽量不使用 this 关键字引用。

2) 命名规范

命名规范在编写代码中起到很重要的作用，虽然不遵循命名规范，程序也可以运行，但是使用命名规范可以很直观地了解代码所代表的含义。下面列出一些命名规范，供读者参考。

(1) 用 Pascal 规则来命名方法和类型，Pascal 的命名规则是第一个字母必须大写，并且后面的连接词的第一个字母均为大写。

(2) 用 Camel 规则来命名局部变量和方法的参数，该规则是指名称中第一个单词的第一个字母小写，后面连接词的首字母也大写。

(3) 所有的成员变量前加前缀“_”。

(4) 接口的名称加前缀“I”。

(5) 方法的命名，一般将其命名为动宾短语。

(6) 所有的成员变量声明在类的顶端，用一个换行把它和方法分开。

(7) 用有意义的名字定义命名空间，如公司名、产品名。

(8) 使用某个控件的值时，尽量命名局部变量。

注意：在类中定义私有变量和私有方法，变量和方法只能在该类中使用，不能对类进行实例化，也不能对其进行调用。

3.2.3 案例实现

1. 案例分析

如图 3-11 中所示的案例是一个非常简单的 C＃控制台项目，向控制台输出一条信息。

2. 代码实现

从菜单中选择“文件”→“新建”→“项目”命令，出现新建项目窗口，如图 3-13 所示。

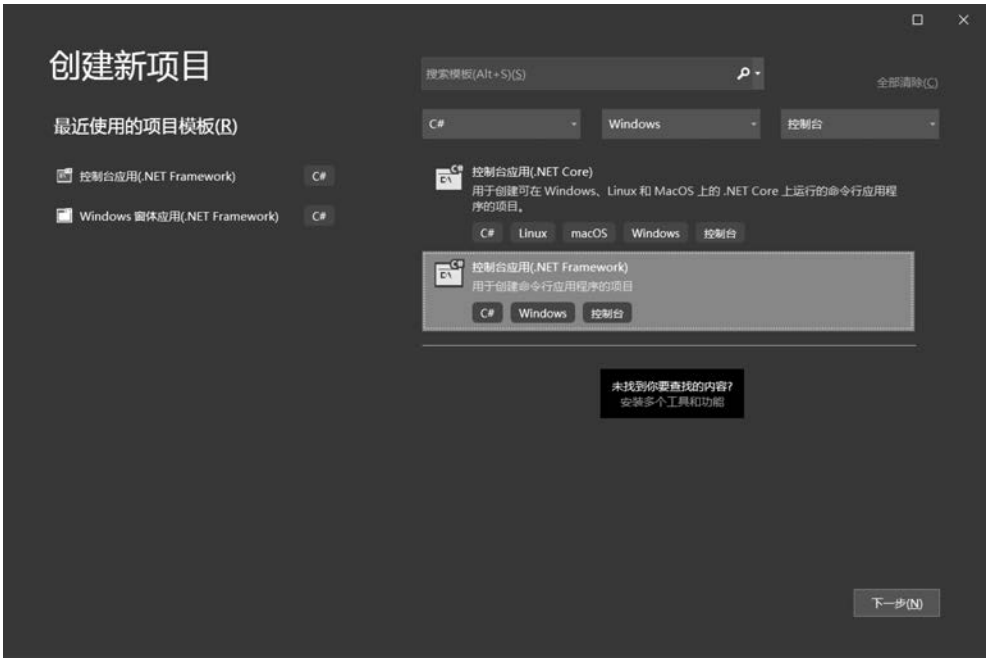

图 3-13　创建控制台项目

单击“下一步”按钮，就创建了一个 C# 的控制台应用程序。在 Main 方法中输入如图 3-14 所示的代码。

在 Microsoft Visual Studio 的菜单栏中选择“生成”→“生成解决方案”命令。如果程序没有错误，那么在窗口下方不会出现错误和警告，状态栏中会显示“生成成功”，说明程序编译成功，可以运行了。

3. 运行效果

案例的运行结果如图 3-15 所示。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Exam
{
    0 个引用
    class Program
    {
        0 个引用
        static void Main(string[] args)
        {
            Console.WriteLine("开始学习C#课程了！");
            Console.ReadKey();
        }
    }
}
```

图 3-14　代码示例

图 3-15　运行效果

本章小结

本章首先介绍了 Visual Studio 2019 的环境，然后介绍了 C＃程序的结构、代码书写规则和命名规范。在 C＃程序的结构中，命名空间、类以及 C＃语句是需要重点掌握的内容。命名空间在 C＃中占有非常重要的地位，我们可以通过引用命名空间，将命名空间中的成员引入到当前的编译单元中。而类是 C＃语言的核心和基本单元模块，程序员通过编写类来表述实际开发过程中需要解决的问题。本章列出了程序编写中需要注意的规则和命名规范。用户在程序代码编写的过程中，应当养成良好的编程习惯。