第 1 章

.NET Core

CHAPTER 1

.NET Core 是一个由微软（Microsoft）提供的免费、开源，具备跨平台能力的应用程序开发框架，可在 Windows、Linux 和 macOS 等操作系统上运行。并且广泛应用于硬件设备、云服务、嵌入式系统和物联网解决方案等领域。.NET Core 的源代码托管在 GitHub 上，由微软和 .NET 社区共同合作维护。该框架旨在提供专业、严谨的开发环境，并具备良好的可读性。

1.1　.NET Core 简介


视频讲解

1.1.1　.NET Core 发展简史

.NET Core 是由微软开发的一条全新跨平台产品线，具有完全开源的特点。该产品线于 2016 年推出了 .NET Core 1.0，致力于提供适用于构建现代跨平台应用程序的 API，并通过 ASP.NET Core 为 Linux 提供服务。然而，由于 1.0 版本的限制和局限性，该系列已经停止维护和支持。与此同时，NET Framework 升级到了 4.6 版本。2017 年，.NET Core 2.0 发布的同时，.NET Framework 也升级到了 4.7 版本。.NET Core 2.x 系列增加了引用 .NET Framework 库的能力，并带来了更大的性能改进。2019 年，.NET Core 3.0 发布，引入了 Windows 桌面应用程序开发、WPF 和 Windows Forms 的现代化和改进，以及 VisualC# 8.0 的语言特性等。同时，.NET Framework 也在该时期升级到了 4.8 最终版本。为了提高可读性，上述补充版本信息是基于已知的历史发布情况进行添加的。

2020 年，微软决定关闭 .NET Framework，并将 .NET Core 更名为 .NET。这一举措旨在整合 .NET Framework 和 .NET Core 的功能，以提供更高性能、更多可选组件和更广泛支持。2021 年，微软推出了统一的长期支持版本 .NET 6.0。该版本支持多种操作系统和平台，包括 Windows、macOS 和 Linux，并提供了对 Web、移动和云等应用程序类型的支持。在 .NET 6 中，开发人员可以利用新的语言特性、增强的工具和框架组件来构建高性能和现代化的应用程序。

2022 年，微软发布了 .NET 7.0 版本。该版本引入了 .NET MAUI（Multi-platform App UI），通过新的控件和 API，提供了更好的性能和可靠性。2023 年 2 月，微软发布了 .NET 8.0 预览版。该版本改进了使用容器镜像的方式，使 .NET 应用程序的表现得到优化。作为 .NET 的一部分，.NET 与 C# 语言紧密结合，同时还引入了原生编译、值类型、结构化并发和快速数组等新功能。此外，.NET 还支持本机 AOT（ahead-of-time）编译，以提高性能和启动速度。图 1-1 展示了 .NET Core 的发展时间线。

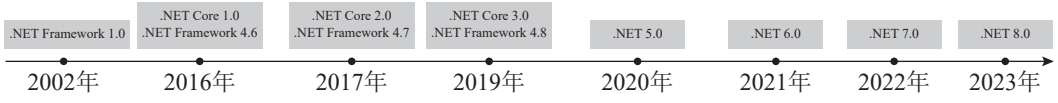

图 1-1　.NET Core 发展时间轴


视频讲解

1.1.2　.NET Framework

.NET Framework 是一个早在 2002 年就开始存在的原始 .NET 实现。从 4.5 版本开始，它支持了 .NET Standard，因此用于 .NET Standard 的代码可以在这些版本的 .NET Framework 上运行。此外，它还包含了一些特定于 Windows 平台的 API，例如，通过 Windows 窗体和 WPF 进行 Windows 桌面开发的 API。基于这些特点，.NET Framework 非常适用于开发 Windows 桌面应用程序。虽然 .NET Framework 与 .NET Core 有密切的联系，但两者之间也存在一些差异，主要差异如下。

（1）应用模板的差异：在 .NET Framework 中，大部分模板是基于 Windows 操作系统的，例如，利用 DirectX 生成的 WPF。不是所有 .NET Framework 的应用模板都能被 .NET Core 所

支持。然而，控制台和 ASP.NET Core 应用模板则是两者都支持的。

（2）API 的差异：早期的 .NET Core 是 .NET Framework 的一个子集，它实现了 .NET Framework 中一些子系统的子级，并包含了许多与 .NET Framework 相同的 API。随着时间的推移，这个子集会不断扩大。


视频讲解

（3）平台的差异：.NET Framework 仅支持 Windows 操作系统，而 .NET Core 除了支持 Windows 操作系统外，还支持 mac OS 和 Linux 操作系统。

（4）开源属性的差异：.NET Core 是一个开源项目，而 .NET Framework 的代码只有部分是开源的。

1.1.3　Mono 运行环境


视频讲解

Mono 是由 .NET 实现的，在需要小型运行时的场景中被广泛使用。它是一种在 Android、macOS、iOS、tvOS 和 watchOS 上驱动 Xamarin 应用程序运行的运行时环境，专门设计用于占用较少内存。此外，Mono 还支持使用 Unity 引擎生成的游戏，并且支持所有目前已发布的 .NET Standard 版本。通常情况下，Mono 会与实时编译器一起使用，但也提供了适用于 iOS 等平台的完整静态编译器。Mono 与 .NET Core 之间的主要差异如下。

（1）应用模板的差异：Mono 通过 Xamarin 产品支持 .NET Framework 的 Windows Forms 等应用模板，而 .NET Core 则不支持这些内容。

（2）API 的差异：Mono 是 .NET Framework 的大型子集，其 API 使用与 .NET Framework 相同的程序集名称和组成要素。而 .NET Core 只实现了 .NET Framework 中子系统的子级。

（3）焦点的差异：Mono 的主要焦点是移动平台，而 .NET Core 的焦点是云平台。

1.1.4　.NET Standard

.NET Standard 是为多个 .NET 实现设计的一套正式的 .NET API 规范。其主要目的是提高 .NET 生态系统的一致性，使开发人员能够通过统一的 API 创建可在各种 .NET 实现中使用的可移植库。引入 .NET Standard 后，各个不同的 .NET 框架将共享统一的基类库，如图 1-2 所示。该规范的引入有助于简化开发过程，并提供更好的代码可重用性和跨平台兼容性。

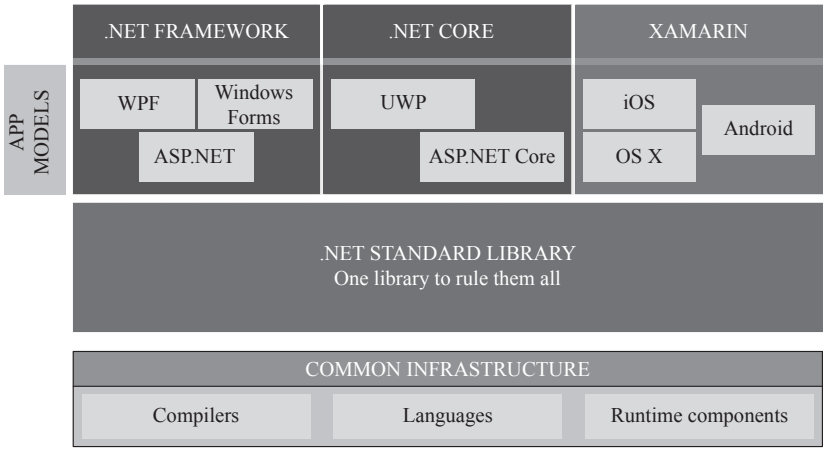

图 1-2　.NET Standard 框架图

各种 .NET 实现都遵循特定版本的 .NET Standard 进行版本控制。每个 .NET 实现版本都会公布其所支持的最高 .NET Standard 版本。随着版本的更新，会添加更多的 API。当库是针对某个特定版本的 .NET Standard 生成时，它可以在任何实现该版本或更高版本的 .NET Standard 的 .NET 实现上正常运行。.NET Standard 的最低支持平台版本如表 1-1 所示。通过遵循 .NET Standard 规范，开发人员可以在不同的 .NET 实现之间实现代码的可移植性，使得开发过程更加便捷和灵活。

表 1-1　.NET Standard 最低支持平台版本

.NET实现	版本支持								
.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0	2.1
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0
.NET framework	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1	不支持
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4	6.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14	12.16
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8	5.16
Xamarin. Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0	10.0
通用 Windows 平台	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299	不支持
Unity	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2021.2

1.1.5　.NET Core 特点

.NET Core 具有以下特点：开源、跨平台、现代、灵活、轻量级、快速、友好、可共享，并为未来的软件开发而构建。

（1）免费开源：.NET Core 是一个开源的框架，由微软和 .NET 社区共同合作维护，源代码托管在 GitHub 上（https://github.com/dotNET/core）。任何开发人员都可以参与 .NET Core 的开发。数千名活跃开发人员正在改进特性、添加新特性和修复 bug 等问题。

（2）跨平台：.NET Core 可以运行在 Windows、macOS 和 Linux 操作系统上，对于每个操作系统有不同的 .NET Core 运行时，执行代码，生成相同的输出。同时，.NET Core 跨体系结构（包括 x86、x64 和 ARM）是一致的，可以导入相同的程序集和库，并在多个平台上使用。

（3）可共享：.NET Core 使用一种由 .NET Standard 编写的统一 API 模型，这种模型对所有 .NET 应用程序都是通用的。相同的 API 或库可以与多种语言的多个平台一起使用。

（4）应用领域广泛：与一些较旧的框架不同，.NET Core 旨在解决现代需求，包括移动友好、构建一次在任何地方运行、可伸缩和高性能。各种不同类型的应用程序都能够被开发并且运行在 .NET Core 平台上，例如，移动端应用程序、桌面端应用程序、Web 应用程序、Cloud 云应用、物联网（IoT）应用、机器学习、微服务、游戏等。

（5）支持多种语言：可以使用 C#、F# 和 Visual Basic 编程语言来开发 .NET Core 应用程序。可以使用自己喜欢的开发工具（IDE），包括 Visual Studio 2017/2019、Visual Studio Code、Sublime Text、Vim 等。

（6）模块化的结构：.NET Core 通过使用 NuGet 包管理，支持模块化开发。.NET Core 有各种不同的 NuGet 包，可以根据需要添加到项目中，甚至 .NET Core 类库也是以包管理的形式提供的。.NET Core 应用程序默认的包是 Microsoft.NETCore.App，模块化的结构减少了内存占用，提升了性能，并且更易于维护。

（7）部署更灵活：.NET Core 应用程序可以部署在用户范围内、系统范围内或者 Docker 容器中。

1.2　ASP.NET Core 简介

ASP.NET Core 是一个由微软和社区开发的跨平台、高性能开源框架，它是下一代 ASP.NET 框架，用于构建现代化的云端应用程序，并提供与互联网连接的功能。该框架具有模块化的特性，可在 Windows 操作系统上完整运行 .NET Framework，也可在跨平台的 .NET Core 上运行。利用 ASP.NET Core，开发人员可以创建 Web 应用程序、服务、物联网应用和移动后端，并且无论是云端还是本地环境，都可以通过多种开发工具在 Windows、macOS 和 Linux 操作系统上进行部署。该框架不仅具备专业性和严谨性，同时也提供了良好的可读性。

1.2.1　ASP.NET Core 发展简史


视频讲解

2016 年，微软为了实现跨平台战略，将 .NET Framework 分离出 .NET Core 版本，并在同年发布了 .NET Core 1.0 版本。同时发布的还有 ASP.NET Core RTM 版，正式版本于 2017 年发布。ASP.NET Core 由模块化的组件构成，保持了解决方案的灵活性并最小化开销。与之前基于 System.Web.dll 的 ASP.NET 不同，当前的 ASP.NET Core 基于一系列细粒度且构建良好的 NuGet 包。

ASP.NET Core 是一个完全重写的框架，将之前独立存在的 ASP.NET MVC（Model-View-Controller）和 ASP.NET Web API 整合到一个编程模型中。虽然它是建立在新 Web 栈上的新框架，但与 ASP.NET MVC 具有高度的概念兼容性。ASP.NET Core 应用程序支持并排版本控制，即在同一台机器上运行的不同应用程序可以使用不同版本的 ASP.NET Core 作为目标。

1.2.2　ASP.NET Core 特征


视频讲解

1. 跨平台性

ASP.NET Core 可以在 Windows、macOS 和 Linux 操作系统上构建和运行跨平台的应用程序，这些应用程序可以托管在 IIS、Apache、Docker 中，甚至自主托管在进程中。

2. 统一的 MVC 和 Web API 技术栈

不论是使用 MVC Controller 还是 ASP.NET Web API，在两种情况下所创建的控制器都继承自相同的 Controller 基类，并返回 ActionResult 对象。

3. 依赖注入

ASP.NET Core 内置支持依赖注入，无须额外配置即可使用。

4. 可测试性

通过内置的依赖注入、用于创建 Web 应用程序和 Web API 的统一编程模型，可以轻松地

对 ASP.NET Core 应用程序进行单元测试和集成测试。

1.3　ASP.NET Core Web 项目开发

1.3.1　第一个 ASP.NET Core Web 应用程序

在本节中，将详细介绍如何使用 Visual Studio 快速创建第一个 ASP.NET Core Web 应用程序。后续所有步骤适用于使用 Visual Studio 2022 进行操作。

【例 1-1】在“D:\ ASP.NET Core 项目”目录中创建 chapter1 文件夹，将其作为网站根目录，创建一个名为 example1-1 的项目，设计页面显示“第一个 ASP.NET Core Web 应用程序 2023”。

具体实现步骤如下。

（1）在“所有应用”中双击打开 Visual Studio 2022，如图 1-3 所示。

图 1-3　Visual Studio 2022 程序图标

（2）选择“创建新项目”→“ASP.NET Core Web 应用”选项，如图 1-4 所示。

图 1-4　新建项目操作

（3）单击“下一步”按钮，在“配置新项目”对话框中，设置项目名称为 example1-1，保存位置为“D:\ASP.NET Core 项目 \chapter1”文件夹，将解决方案和项目放于同一目录中，单击“下一步”按钮，如图 1-5 所示。

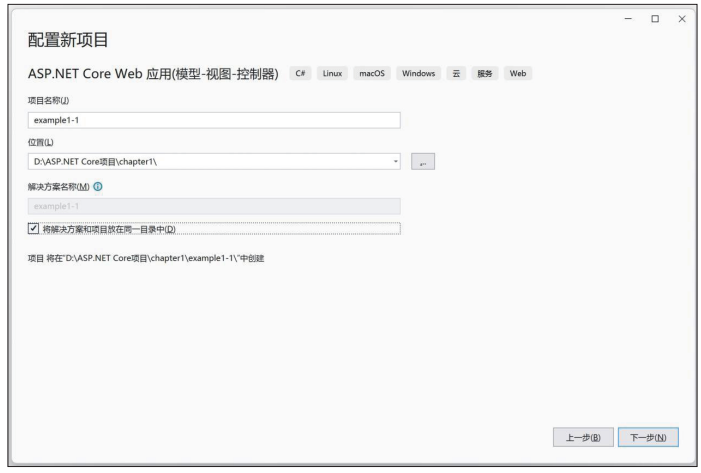

图 1-5　“配置新项目”对话框

（4）在“其他信息”对话框中，将框架设置为 .NET6.0，其他信息为默认项，单击“创建”按钮，如图 1-6 所示。

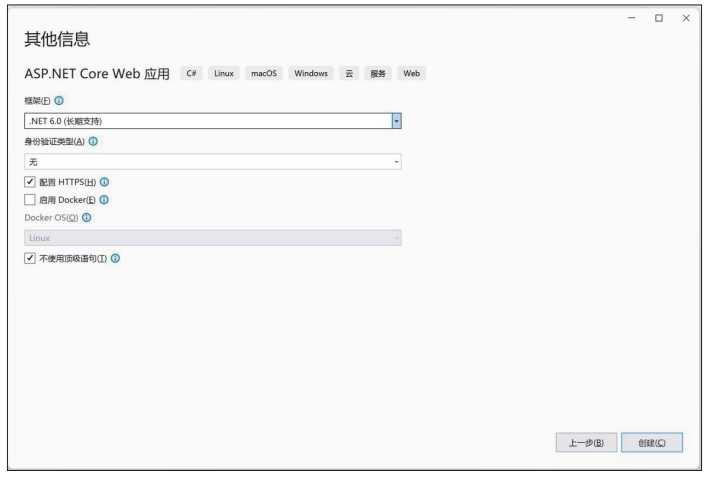

图 1-6　“其他信息”对话框

（5）网站创建后出现如图 1-7 所示的界面，右侧为“解决方案资源管理器”目录，包含 Pages 文件夹、appsettings.json 配置文件等基本项。

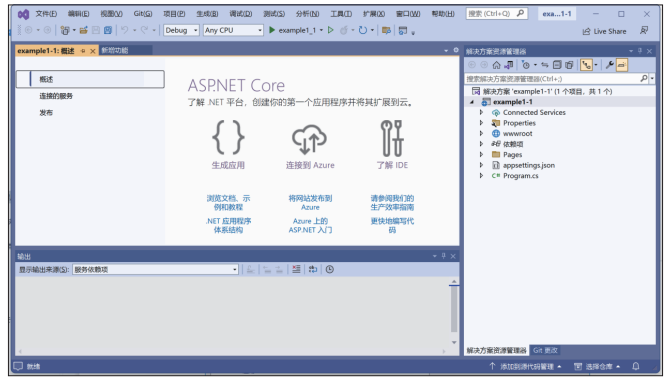

图 1-7　初始空网站

（6）双击 example1-1 项目下的 Pages 文件夹，单击打开 Index.cshtml 页面，编辑代码如下。

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>第一个 ASP.NET Core Web 应用程序 @DateTime.Now.Year</p>
    <p>Learn about <a href="https://docs.microsoft.com/aspNET/
core">building Web apps with ASP.NET Core</a>.</p>
</div>
```

（7）再右击 Index.cshtml 文件，在弹出的快捷菜单中选择“在浏览器中查看（Microsoft Edge）”选项，如图 1-8 所示。

（8）页面运行，显示 ViewBag.Title 的属性值和段落中“第一个 ASP.NET Core Web 应用程序 2023”文字，如图 1-9 所示。

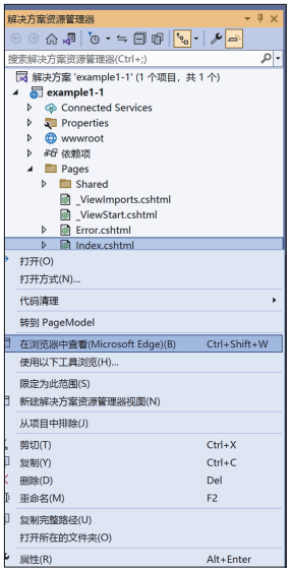

图 1-8 在浏览器中查看网页的操作

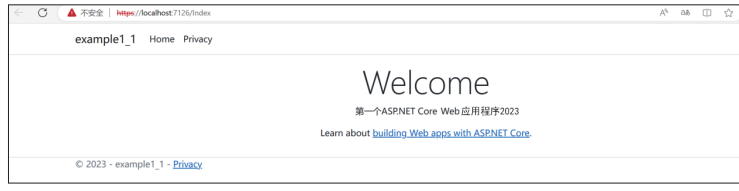

图 1-9 浏览器中显示的初始页面

至此，第一个 ASP.NET Core Web 应用程序就已经实现了，上述操作的详细内容将在后续各章节中展开逐一讲解。

1.3.2 ASP.NET Core Web 应用程序的结构

创建完第一个 ASP.NET Core Web 应用程序后，会自动生成一些目录和文件，如图 1-10 所示，表 1-2 详细介绍了这些目录和文件的主要用途。

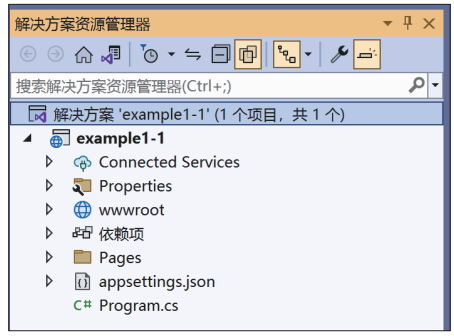

图 1-10　ASP.NET Core 网站的目录结构

表 1-2　ASP.NET Core 网站的目录结构说明

目录和文件	说　明
依赖项	ASP.NET Core 开发、构建和运行过程中的依赖项：包、元包和框架； Microsoft.NETCore.App 是一些包的集合，包含 .NET core 的基础运行时和基础类库； ASP.NET Core 共享框架（Microsoft.AspNETCore.App）包含由微软开发和支持的程序集
Properties	配置和存放一些 .json 文件用于配置 ASP.NET Core 项目
launchSettings.json	启动配置文件是 ASP.NET Core 项目应用保存特有的配置标准，用于应用的启动准备工作，包括环境变量、开发端口等
wwwroot	网站根目录，存放类似 CSS、JS、图片和 HTML 等静态资源文件的目录
Program.cs	包含 ASP.NET Core 应用的 Main() 方法，负责配置和启动应用程序

1.3.3　ASP.NET Core 中的文件类型

在创建一个网站之后，打开网站文件时，会看到各种不同类型的文件。特别是当浏览一个大型工程目录时，可能会感到有些困惑，因为在 .NET 中存在许多不同的文件类型。本节将详细解释 ASP.NET Core 中的不同文件类型和它们的扩展，对这些文件进行逐一讲解，以便更好地理解它们的作用和功能。

1. VS.NET 的文件类型

首先打开项目目录，认识一下 VS.NET 网站项目中使用的文件类型，表 1-3 提供了有关 VS.NET 使用的文件类型 (以 C# 项目为例)。

表 1-3　VS.NET 的文件类型

文件类型	后缀名	说　明
解决方案文件	.sln	存储在解决方案中的项目信息，以及通过属性窗口访问全局构建设置
用户选项文件	.suo	存储特定用户的设置。VS.NET 中的源控制集成包使用此类文件存储 Web 项目的转换表、项目的离线状态，以及其他项目构建的设置
C# 项目文件	.csproj	存储项目细节，如参考内容、名称、版本等
C# 项目的用户选项	.csproj.user	记录存储用户的相关信息

2. 通用开发文件类型

回到 Visual Studio 2022，在“解决方案资源管理器”中，右击 example1-1 项目，在弹出的快捷菜单中，选择“添加”→“新建项”选项，弹出“添加新项”对话框，单击“显示所有模板”按钮，弹出的“添加新项”窗口如图 1-11 所示。窗口中显示网站项目中可以使用的所有文件类型，其中 Windows 服务和 Web 应用程序开发通用的文件类型说明如表 1-4 所示。

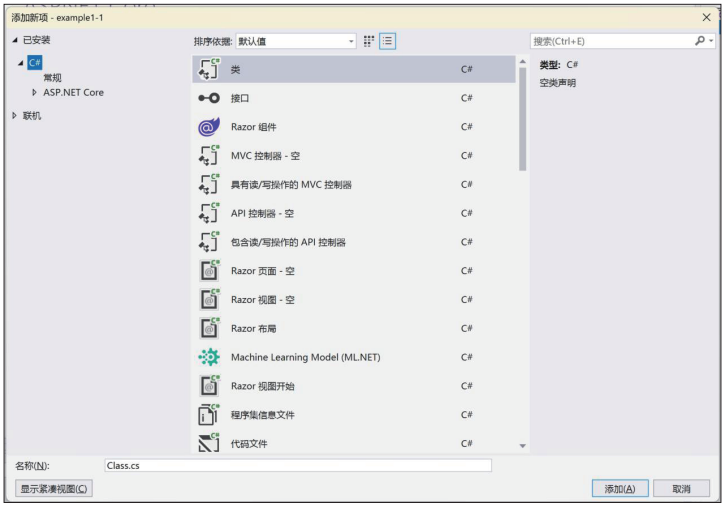

图 1-11　添加新项模板

表 1-4　VS.NET 通用开发文件类型

文件类型	后缀名	说　明
C# 文件	.cs	C# 源代码的文件
MVC 视图文件	.cshtml	MVC 视图文件
XML 文件	.xml	XML 文件与数据标准文件
数据库文件	.mdf	SQL Server 数据库文件
类图文件	.cd	类图表文件
脚本文件	.js	JavaScript 代码的文件
图标文件	.ico	图标样式的图像文件
文本文件	.txt	简单文本文件

3. ASP.NET Core 的文件类型

ASP.NET Core 开发还可以使用一些特定的文件类型，如表 1-5 所示。

表 1-5　ASP.NET Core 的文件类型

文件类型	后缀名	说　明
MVC 视图文件	.cshtml	只能在 MVC 3 或更高版本等支持 Razor 的框架中使用
Web 窗体文件	.aspx	代码分离（code-behind）文件的 Web 窗体
全局程序文件	.asax	全局应用程序类，允许编写代码以处理全局 ASP.NET 程序事件，一个项目最多只可以包括一个无法更改的 global.asax 文件
静态页面文件	.htm	标准的 HTML 页
样式文件	.css	在网站上设置外观使用的层叠样式表

续表

文件类型	后缀名	说　明
站点地图文件	.sitemap	Web 应用程序表示页面间层次关系的站点地图
皮肤文件	.skin	用于指定服务器控件的主题
用户控件文件	.ascx	用户自主创建的 Web 控件
浏览器文件	.browser	定义浏览器相关信息的文件

1.4　Visual Studio 2022 开发环境的基本介绍

1.4.1　菜单栏和工具栏

Visual Studio 2022 的菜单栏继承了 Visual Studio 早期版本的所有选项功能，其中包括“文件”“编辑”“视图”“窗口”“帮助”等核心功能。此外，还提供了一些编程专用的功能菜单，如“生成”“调试”“测试”。菜单栏下方是工具栏，它包含了一些常用功能的工具按钮，如图 1-12 所示。通过菜单栏和工具栏，开发人员可以方便地使用和导航 Visual Studio 提供的各种功能。

图 1-12　菜单和工具栏

（1）显示工具箱及属性窗口：在“视图”菜单中，可以选择显示“解决方案资源管理器”“属性窗口”“工具箱”“错误列表”等窗口的选项，除了“帮助”窗口，其他所有窗口及内容的显示都可以通过“视图”菜单设置，如图 1-13 所示。

图 1-13　“视图”菜单

（2）程序执行及断点调试：通过“调试”菜单可以进行程序调试、执行等编译操作，在代码内部新建、取消断点，对程序进行逐语句、逐过程（直接调用函数、属性的模块，不逐条执行模块内语句）调试，如图 1-14 所示。

（3）代码文本编辑：选择“工具”→“选项”选项，弹出“选项”对话框，在“环境”选项卡中的“字体和颜色”选项可以设置代码编辑区域文本的字体、大小、项前景、项背景等属性，如图 1-15 所示。

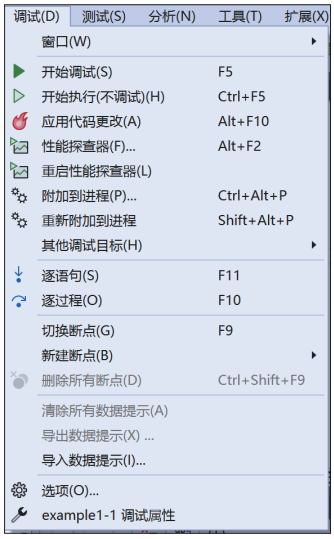

图 1-14　“调试”菜单

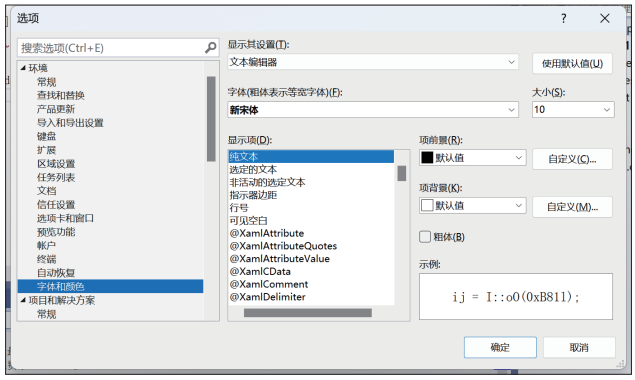

图 1-15　“字体和颜色”选项卡

在“文本编辑器”列表中的 C# 选项卡里可以设置自动换行、显示行号等属性，如图 1-16 所示。

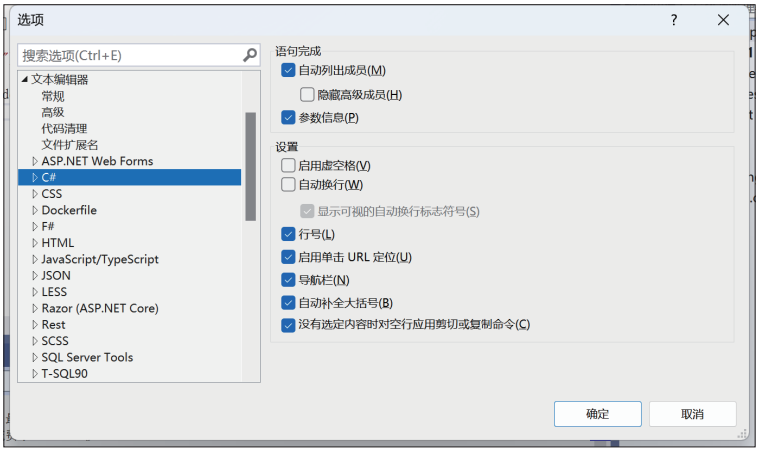

图 1-16　C# 选项卡

1.4.2　工具箱窗口

在 Visual Studio 2022 中，左侧通常是“工具箱”窗口。“工具箱”窗口可以列出用于开发 Web 应用程序的基本 HTML 标签，如图 1-17 所示。当开发人员需要使用某个控件时，只需从

“工具箱”窗口中拖拽该控件到界面上即可，这极大地节省了编写代码的时间，提高了程序设计的效率。

在“工具箱”窗口中右击，弹出快捷菜单，该菜单提供了对选项卡的添加、删除、重命名等操作选项，如图 1-18 所示。如果选择“选择项”选项，那么将会弹出“选择工具箱项”对话框，通过该对话框可以为“工具箱”添加其他可选控件和第三方组件，如图 1-19 所示。这些额外的选项可以丰富“工具箱”的功能，让开发人员能够更灵活地选择和使用不同的工具和组件。

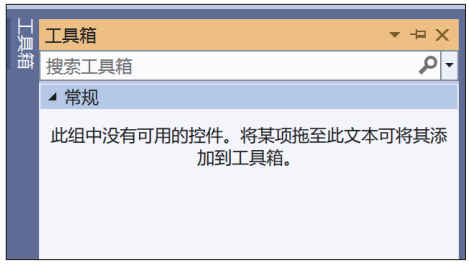

图 1-17　Visual Studio 2022“工具箱”窗口

图 1-18　“工具箱”窗口的快捷菜单

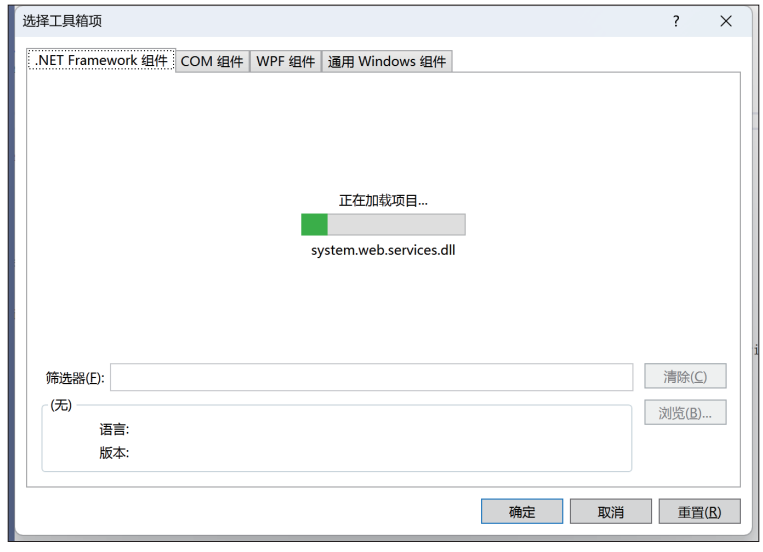

图 1-19　“选择工具箱项”对话框

1.4.3　解决方案资源管理器

在 Visual Studio 2022 中，右侧通常为“解决方案资源管理器”窗口。该窗口提供了网站项目和文件的组织结构视图，以便于导航和管理，如图 1-20 所示。通过“解决方案资源管理器”窗口，可以清晰地查看项目的层次结构，包括各个类库、数据库文件和系统配置文件等。此外，还可以在此处添加或删除文件，并且能够添加系统或用户文件夹，以实现对文件的管理。通过“解决方案资源管理器”窗口，开发人员能够更方便地组织和管理项目中的各类资源。

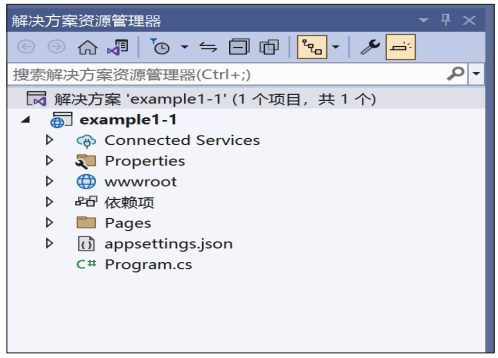

图 1-20　“解决方案资源管理器”窗口

1.4.4　属性窗口

Visual Studio 2022 的右下角是“属性”窗口，可以查看属性，还可以对页面及页面中的控件进行量值化的属性值设置。“属性”窗口最顶部的下拉列表，可以选择要进行属性设置的对象，图标表示属性列表按字母排序，图标表示属性列表按分类排序。当修改某个对象的属性值，会将该属性值自动添加到 HTML 源代码中，实现两者同步，反之亦然，如图 1-21 所示。

图 1-21　“属性”窗口

1.5　综合实验一：Visual Studio 2022 的安装

1. 主要任务

安装开发环境，开始使用 Visual Studio 2022 进行网站开发。

2. 实验步骤

（1）确认配置。

为了确保计算机能够支持使用 Visual Studio 进行开发，本书以 Visual Studio 2022 作为示例。表 1-6 列出了安装 Visual Studio 2022 的相关系统和最低配置要求。如果使用其他版本，可以参考官网提供的配置说明以确保符合要求。这样做可以保证在使用 Visual Studio 进行开发时，系统具备足够的硬件和软件支持，以获得更好的性能和稳定性。

表 1-6　安装 Visual Studio 2022 的系统和最低配置要求

支持的操作系统	硬件最低配置要求	支持的语言
Windows 11 版本 21H2 或更高版本； Windows 10 版本 1909 或更高版本； Windows Server Core 2022； Windows Server Core 2019； Windows Server 核心 2016； Windows Server 2022、2019、2016：标准和数据中心	ARM64 或 x64 处理器，不支持 ARM32 处理器； 至少 4 GB RAM； 至少 2 个 vCPU 和 8 GB RAM； 硬盘 850 MB~210 GB 可用空间； 支持最低显示分辨率 WXGA（1366 像素 ×768 像素）的显卡	英语、简体中文、繁体中文、捷克语、法语、德语、意大利语、日语、韩语、波兰语、葡萄牙语（巴西）、俄语、西班牙语和土耳其语等 14 种语言

（2）文件下载。

从微软的官网（https://visualstudio.microsoft.com/zh-hans/downloads/）下载 Visual Studio 的安装程序，如图 1-22 所示。确定要安装的 Visual Studio 版本和版次，本书选择（community 社区）版，下载引导程序文件。

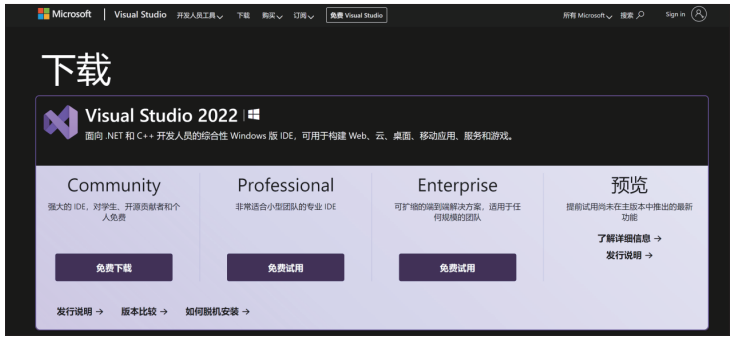

图 1-22　Visual Studio 2022 版本选择

（3）启动安装。

找到文件下载目录，双击 VisualStudioSetup.exe 引导程序进行安装。安装过程会要求确认 Microsoft 软件许可条款和隐私声明，单击“继续”按钮，如图 1-23 所示。

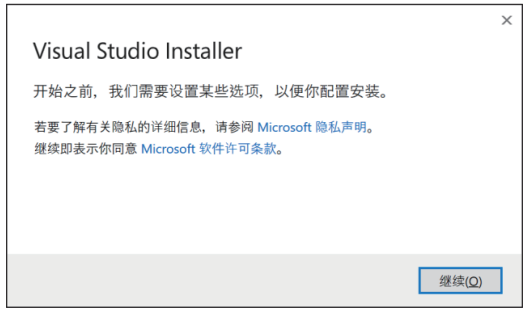

图 1-23　Microsoft 软件许可条款和隐私声明

（4）选择工作负荷。

安装 Visual Studio 的安装程序后，可以通过选择所需的工作负荷进行自定义安装。 例如，“ASP.NET 和 Web 开发”工作负荷可以使用 Web Live Preview 编辑 ASP.NET 网页，或者使用 Blazor 生成响应式 Web 应用程序，在安装程序中选择“ASP.NET 和 Web 开发”的工作负荷，如图 1-24 所示。

图 1-24　工作负荷选项卡

（5）选择组件（可选）。

如果不想使用工作负荷来完成 Visual Studio 安装，或者需要添加比工作负荷更多的组件，可通过“单个组件”选项卡来完成此操作。选择所需组件，然后按照提示进行操作，如图 1-25 所示。

图 1-25　单个“组件”选项卡

（6）安装语言包（可选）。

默认情况下，安装程序首次运行时会尝试匹配操作系统语言。若要以其他语言进行安装，请在 Visual Studio 安装程序中单击“语言包”标签，进入“语言包”选项卡，然后按照提示进行操作，如图 1-26 所示。

图 1-26　“语言包”选项卡

（7）选择安装位置（可选）。

选择非系统盘安装可减少系统驱动器上 Visual Studio 的安装占用，安装位置选择如图 1-27 所示。

图 1-27　“安装位置”选项卡

（8）开始开发。

安装完成后，单击“启动”按钮，进行个性化设置后。开始使用 Visual Studio 进行开发，如图 1-28 所示，至此 Visual Studio 2022 安装成功。

图 1-28　Visual Studio 2022 初始运行界面

1.6 本章小结

本章旨在全面介绍 .NET Core 的基本特征，并与 .NET Framework 进行比较，以分析各自的优缺点。同时，详细解释了开发 ASP.NET Core Web 应用程序的每个步骤，以及使用的各种文件夹和文件。此外，还对 Visual Studio 2022 作为集成开发环境进行了基本介绍。通过本章的学习，读者将掌握 .NET Core 的核心概念和技术，从而在实际开发中有效地利用这些知识。

1.7 习题

一、选择题

1. .NET Core 的特点是（　　）。

A. 跨平台

B. 完全开源

C. 提供适用于构建现代跨平台应用程序的 API

D. 所有选项都正确

2. .NET Core 3.0 引入了（　　）改进。

A. Windows 桌面应用程序开发　　B. WPF 和 Windows Forms 的现代化和改进

C. Visual C# 8.0 的语言特性　　D. 所有选项都正确

3. 在 2021 年，微软推出的统一、长期支持版本是（　　）。

A. .NET Framework　B. .NET Core　C. .NET 6.0　D. .NET 7.0

4. 在 2023 年 2 月，微软发布的预览版是（　　）。

A. .NET 6.0　B. .NET 7.0　C. .NET 8.0　D. .NET MAUI

5. 各种 .NET 实现遵循（　　）的 .NET Standard 进行版本控制。

A. 最高版本　B. 最低版本　C. 任意版本　D. 特定版本

6. 通过遵循 .NET Standard 规范，开发人员可以实现（　　）。

A. 代码的可移植性　B. 代码的高效性　C. 代码的安全性　D. 代码的稳定性

7. 微软为了实现跨平台战略，将 .NET Framework 分离出 .NET Core 版本的时间是（　　）。

A. 2016 年　B. 2017 年　C. 2018 年　D. 2019 年

8. ASP.NET Core 框架整合了之前存在的（　　）。

A. ASP.NET MVC 和 ASP.NET Web API

B. ASP.NET Core MVC 和 ASP.NET Web Forms

C. ASP.NET Core Web API 和 ASP.NET Web Pages

D. ASP.NET Core MVC 和 ASP.NET Web Pages

9. ASP.NET Core 开发、构建和运行过程中的依赖项主要包括（　　）。

A. 包、元包和框架　B. 包和元包　C. 元包和框架　D. 包和框架

10. 启动配置文件 launchSettings.json 用于（　　）。

A. 存放 ASP.NET Core 项目的配置文件

B. 存放 ASP.NET Core 项目的静态资源文件

C. 存放 ASP.NET Core 项目的启动准备工作配置

D. 存放 ASP.NET Core 应用的 Main() 方法配置

11. Visual Studio 2022 的菜单栏继承了（　　）早期版本的核心功能。

A.“文件”“编辑”“视图”“窗口”“帮助”

B.“生成”“调试”“测试”

C.“解决方案管理器”“属性窗口”“工具栏”“错误列表”

D.“视图”“帮助”“解决方案管理器”

12. 在 Visual Studio 2022 中，通过（　　）显示“工具箱”及“属性”等窗口。

A. 单击“文件”菜单　　B. 单击“编辑”菜单

C. 单击“视图”菜单　　D. 单击“帮助”菜单

二、填空题

1. 从 .NET Framework4.5 版本开始支持_____。

2. .NET Framework 适用于开发_____应用程序。

3. .NET Core 具有_____、跨平台、现代、灵活、轻量级、快速、友好、可共享的特点，并为未来的软件开发而构建。

4. .NET Core 可以运行在_____、macOS 和 Linux 操作系统上，对于每个操作系统有不同的 .NET Core 运行时，执行代码，生成相同的输出。

5. ASP.NET Core 可以在_____、_____和_____操作系统上构建和运行应用程序。

6. 由 ASP.NET Core 统一的 MVC 和 Web API 技术栈中，控制器继承自_____基类，并返回_____类型的对象。

7. 在 VS.NET 网站项目中，解决方案文件的后缀名是_____。

8. 在 VS.NET 网站项目中，C# 项目的用户选项文件的后缀名是_____。

9. ASP.NET Core 开发、构建和运行过程中的依赖项包括_____、_____和_____。

10. 启动配置文件是 ASP.NET Core 项目应用保存特有的配置标准的文件名是_____。

三、简答题

1. 简述 .NET Framework 和 .NET Core 在应用模板上的差异。

2. 简述 .NET Framework 和 .NET Core 在 API 上的差异。

3. 简述 .NET Core 的特点。

4. 简述支持 .NET Core 运行的操作系统。