HITE 7.0 软件开发与应用工程师

使用.NET 技术开发 Web 应用程序

(第 2 版)

张家界航空工业职业技术学院
广西科技职业学院
武昌首义学院
武汉厚溥数字科技有限公司

编著

清華大學出版社
北　京

内容简介

本书是一本全面介绍如何使用.NET 6 来构建 Web 应用的教材。本书从搭建第一个 ASP.NET Core 项目开始，帮助开发者逐步了解和掌握 ASP.NET Core MVC、数据传递、视图与模型、中间件与过滤器、Entity Framework Core、Web API 开发、身份认证与授权、发布与部署等重要知识点，并最终完成一个功能完整的综合项目——设备管理系统。

本书可作为高等院校计算机相关专业的教材，也可作为 Web 程序开发人员提升专业技能的参考资料。

图书在版编目(CIP)数据

使用.NET 技术开发 Web 应用程序 / 张家界航空工业职业技术学院等编著. -- 2 版. -- 北京 : 清华大学出版社, 2025. 7. -- (HITE 7.0 软件开发与应用工程师).

ISBN 978-7-302-69526-4

Ⅰ. TP393.092.2

中国国家版本馆 CIP 数据核字第 2025MS9683 号

责任编辑：刘金喜

封面设计：王　晨

版式设计：思创景点

责任校对：成凤进

责任印制：沈　露

出版发行：清华大学出版社

网　　址：https://www.tup.com.cn，https://www.wqxuetang.com

地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084

社 总 机：010-83470000　　邮　　购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市人民印务有限公司

经　　销：全国新华书店

开　　本：185mm×260mm　　**印　　张：**18　　**字　　数：**438 千字

版　　次：2019 年 4 月第 1 版　　2025 年 8 月第 2 版　　**印　　次：**2025 年 8 月第 1 次印刷

定　　价：68.00 元

产品编号：108429-01

编 委 会

前　言

随着互联网的迅猛发展，Web 应用已经成为现代软件开发的核心组成部分之一。在现今数字化和信息化时代，Web 开发者不仅需要掌握传统的前端技术，还需要具备强大的后端开发能力，以实现功能丰富、性能高效、易于扩展的 Web 应用。而.NET 6 作为微软推出的最新版本的跨平台开发框架，凭借其高效、灵活和强大的特性，已经成为构建现代 Web 应用的理想选择。

本书的编写旨在为广大开发者提供一本系统、全面、实用的学习资料，帮助他们深入理解和掌握.NET 6 中的 Web 开发技术。通过本书的学习，读者可以从零开始，逐步掌握构建 ASP.NET Core Web 应用所需的核心技术和最佳实践，并最终能够独立完成复杂的 Web 项目的开发。

本书全面覆盖了.NET 6 Web 应用开发的各个方面，内容按照学习的难度和实际应用场景编排，详细讲解如何使用.NET 6 和 ASP.NET Core 构建高效、可扩展的 Web 应用。

本书具体内容如下。

单元一　初识 ASP.NET Core：介绍.NET 6 和 ASP.NET Core 的基础概念，为后续学习打下坚实的基础。

单元二　第一个 ASP.NET Core MVC 应用：带领读者从实践入手，构建第一个简单的 ASP.NET Core MVC 应用，理解 MVC 模式的核心理念。

单元三　页面之间的数据传递：介绍如何在 Web 页面之间传递数据。

单元四　视图与模型：讲解如何通过视图和模型实现数据绑定与显示，提升 Web 应用的用户体验及交互性。

单元五　中间件与过滤器：深入讲解中间件和过滤器的概念与应用，帮助读者掌握请求处理的高级技巧。

单元六　Entity Framework Core：通过详细的实例讲解如何使用 Entity Framework Core 进行数据访问和数据库操作。

单元七　ASP.NET Core Web API：介绍如何开发 RESTful API，实现前后端分离架构，并提供高效的 API 服务。

单元八　ASP.NET Core 角色与授权：深入讲解如何实现基于角色和权限的用户认证与授权管理，确保 Web 应用的安全性。

单元九　ASP.NET Core 发布与部署：介绍应用的部署流程。

单元十　综合项目——设备管理系统：通过一个完整的项目案例，带领读者整合所学知识实现一个设备管理系统。

本书的编写注重实践与理论结合，每一单元都以实例为主线，结合实际开发中常见的场景，逐步讲解关键技术和工具的应用。通过代码示例、图解与步骤说明，帮助读者清晰地理解每个技术点的实现原理与应用方式。在单元末尾，我们还提供了单元自测和上机实战，帮助读者巩固知识并加深理解。

本书适用于希望掌握.NET 6 Web 开发的开发者，尤其适合以下几类读者。

(1) 初学者：对 Web 开发有一定兴趣，并希望通过学习 ASP.NET Core 快速入门。

(2) 有经验的开发者：已经掌握一些 Web 开发技术，想进一步提高自己在.NET 环境下开发 Web 应用的能力。

(3) 求职者与转行者：希望转向.NET Web 开发方向，提升职场竞争力。

(4) 团队与企业开发者：希望在团队或企业级应用中引入.NET 6，通过学习本书的内容来提升项目开发效率和质量。

本书由一支经验丰富的.NET 开发团队编写，每一位作者都有着多年的 Web 开发经验，并在多个企业级项目中实践过.NET 技术。为了确保本书内容的准确性与实用性，我们在编写过程中采用了精心设计的内容组织结构，并通过真实的案例进行详细解析。在每一单元的编写中，团队成员通力合作，确保内容从技术讲解到实际应用的全面覆盖，力求让每位读者都能够轻松理解并掌握。

由于编者水平有限，书中难免存在欠妥和疏漏之处，敬请广大读者批评指正。

本书 PPT 教学课件和案例源文件可通过扫描下方二维码，将链接地址推送到邮箱进行下载。

教学资源

服务邮箱：476371891@qq.com

编　者

2025 年 1 月

目　录

单元

一

初识ASP.NET Core

课程目标

项目目标

- 搭建 ASP.NET Core 的开发环境
- 搭建第一个 ASP.NET Core 项目
- 理解 ASP.NET Core 项目各个文件夹的意思

技能目标

- 了解 ASP.NET Core 的基本概念
- 熟悉安装和配置 ASP.NET Core 环境
- 掌握构建 ASP.NET Core 项目的应用方法

素养目标

- 培养创业创新的意识
- 培养敏锐的观察力和思考能力
- 具有持续学习和提升的能力

简介

ASP.NET Core 是 Microsoft 开发的一个跨平台的开源 Web 应用程序框架。它是 ASP.NET 的下一代版本，主要提高了 Web 应用程序的性能和可扩展性，其支持在 Windows、Linux 和 MacOS 等多个平台上运行。ASP.NET Core 是基于.NET Core 运行时环境构建的，可以与多种编程语言(如 C#、F#和 VB.NET)一起使用。ASP.NET Core 具有轻量级、高性能、模块化和可测试性等特点，它的出现体现了科技创新的重要性，并为企业创新及提高生产力和市场竞争力提供了强大的支持。同时，ASP.NET Core的开源和跨平台特点符合社会主义市场经济的原则，帮助开发者探索具有高性能、高安全性和健康可持续的 Web 应用程序的开发之路。本单元主要阐述了从.NET Framework 到.NET Core 的发展历程，以及如何使用 ASP.NET Core 构建 Web 应用。

ASP.NET MVC 介绍

1.1.1 任务描述

如果想要开发 ASP.NET Core Web 应用，需先搭建好开发环境。ASP.NET Core 是一个开源的、高性能、跨平台的框架，它支持多种平台环境，目前比较常用的有 Windows 或 MacOS 系统，以及日渐受欢迎的 Docker 和 Linux 系统。本次任务，我们主要了解从.NET Framework 到.NET Core 的发展历程、ASP.NET Core 是什么，以及如何在 Windows 系统上安装和配置.NET Core 的开发环境。

1.1.2 知识学习

1. .NET Framework 发展史

.NET Framework 的发展历程可以说是一段充满挑战与机遇的传奇故事。它告诉我们，只有不断学习、勇于探索和团队协作，才能在技术的快速发展中立足。.NET Framework 是 Microsoft 公司开发并于 2000 年发布的一个全面的应用程序平台，旨在帮助开发人员构建基于 Windows 平台的应用程序。随着技术的不断发展和演变，Microsoft 公司在 2006 年发布了.NET Framework 3.5，该版本在当时受到了广泛的关注和使用。

从 2000 年开始，经过多年的苦心经营，Microsoft 已经在 Windows 平台中构建了一个完整的支持多种设备的.NET 系统。

微软是全球较大的计算机软件供应商之一，为了占据开发市场，在 2002 年，Microsoft 发布了.NET Framework 1.0，这是一个全新的开发框架，它将 Windows 操作系统与 Web 应用程序的开发融合在一起。.NET Framework 1.0 包括了一个运行时环境(Common Language Runtime，CLR)和一组类库，这些类库提供了许多常用的功能，如文件操作、网络通信、XML 处理等。另外，.NET

Framework 1.0 也支持多语言开发。为了吸引更多的开发者涌入平台，微软在 2002 年宣布推出了一个特性强大且与.NET 平台无缝集成的编程语言，即 C# 1.0 正式版。使用.NET 支持的编程语言，开发者就可以通过.NET 平台提供的工具服务和框架支持便捷地开发应用程序。

2003 年，Microsoft 发布了.NET Framework 1.1，该版本增加了许多新的类库及功能，同时也修复了一些漏洞和问题。

2005 年，Microsoft 发布了.NET Framework 2.0，该版本也增加了许多新的类库及功能，如 Windows Presentation Foundation(WPF)、Windows Communication Foundation(WCF)和 Windows Workflow Foundation(WF)。.NET Framework 2.0 也包括了对 LINQ(Language Integrated Query)的支持，其是一种强大的数据查询语言。

2007 年，Microsoft 发布了.NET Framework 3.0，该版本没有增加新的 CLR 版本，而是增加了一些新的类库及功能，如 Windows CardSpace、WPF 和 WCF。

2008 年，Microsoft 发布了.NET Framework 3.5，该版本增加了许多新的类库及功能，如 LINQ to SQL、ADO.NET Entity Framework 和 ASP.NET AJAX。

2010 年，Microsoft 发布了.NET Framework 4.0，该版本增加了许多新的类库及功能，如 Parallel Extensions、Dynamic Language Runtime(DLR)和 Code Contracts。.NET Framework 4.0 也支持多框架开发，这允许开发人员在同一个应用程序中使用不同版本的.NET Framework。

2012 年，Microsoft 发布了.NET Framework 4.5，该版本增加了许多新的类库及功能，如 Async/Await、HttpClient 和 WebSocket。.NET Framework 4.5 也支持 Windows 8 和 Windows Server 2012。

此外，.NET Framework 为开发者提供了新的编程接口(Application Programming Interface，API)和开发工具。这些创新使得程序设计人员可以同时开发 Windows 应用程序、组件和 Web 服务(Web Service)。.NET Framework 还引入了面向对象的反射编程接口，进一步提升了开发效率和灵活性。

.NET Framework 是以一种采用系统虚拟机运行的编程平台，以公共语言运行时为基础，支持多种语言(C#、VB.NET、C++、J#等)的开发。

随着时代的发展及框架版本的不断升级，.NET Framework 变得越来越强大，但也在其他方面变得越来越“臃肿”，由于自身的这些束缚和限制，它想要做一些快速的迭代和更新，就变成了一件艰难的事情，就像一条宽阔的大河挡在了 Microsoft 发展的道路上。因此，.NET 的用户数较 Java 大幅度减少，其已经到了不得不做出改变的时候，于是在 2016 年 6 月 27 日，.NET Core 1.0 项目正式发布，彻底改变了 Windows Only 的场景，开始拥抱开源和跨平台。.NET Core 的开发目标是跨平台的.NET 平台，因此.NET Core 包含了.NET Framework 的类库，但与.NET Framework 打包式安装方法不同的是，.NET Core 采用包化(Packages)的管理方式，应用程序只需要获取需要的组件即可，同时各包亦有独立的版本线(Version Line)，不再硬性要求应用程序跟随主线版本。随后 Microsoft 推出了.NET 5/6/7 版本，其中.NET 6 是.NET 统一计划的关键里程碑。这一统一计划通过统一的 SDK、基本库和运行时(Runtime)，将跨平台、桌面、物联网(IoT)和云应用整合到一个统一的开发环境中，极大地提高了开发效率和平台兼容性。

2021 年 11 月 8 日发布的.NET 6，是.NET 团队和社区历经一年多努力所取得的成果。其中 C#10 和 F#6 提供了语言的改进功能，使代码变得简单、易用，其性能有了巨大的提升。.NET 6 首次发布了对本地化 Apple Silicon(Arm64)的支持，并且改进了 Windows Arm64 的相关性能。.NET 6

构建了一个新的动态配置文件导向优化(PGO)系统，该系统提供了仅在运行时才可能实现的深度优化功能。云诊断已改进与 dotnet monitor 和 Open Telemetry 的集成。Web Assembly 的支持更强大，性能也得到了提升。新增的 API 不仅支持 HTTP/3 协议，能够显著提升网络传输效率，还强化了对 JSON 数据的处理能力，同时借助新的内存操作机制，开发人员可以直接操作内存，有效提升数据处理速度和资源利用率。许多开发人员已积极将应用程序升级到 .NET 6，在生产环境中，这些升级后的应用程序在性能、响应速度等方面都取得了显著的优化和提升。

2022 年 2 月 17 日，Microsoft 发布了 .NET 7.0.0-preview.1，目前最新版是 .NET 7.0.0-rc1。Microsoft 在开源的道路上一直秉持着坚决且认真的态度。自.NET Core 3.0 的核心组件完善以后，.NET Core 的项目虽然已基本完成，但依然保持着快速更新迭代的节奏。从 Microsoft 近几年的迭代规划来看，以后每年都会推出一个大版本，其中部分版本将被指定为长期支持(LTS)版本。.NET Core 的发展历程如图 1-1 所示。

图 1-1　.NET Core 的发展历程

如今，.NET Core 已经取得了巨大的进步和发展，告别了之前臃肿的体态，完成了华丽的蜕变，从一个闭源的、单一的.NET Framework 框架，变成了一个积极开源并大力拓展生态社区的、跨平台的、轻量级的.NET Core 便捷框架。

2. ASP.NET Core 概述

ASP.NET Core 是基于.NET Core 的一个 Web 应用程序框架，它提供了一系列的工具和组件，可以帮助开发人员快速构建高性能、可扩展、安全的 Web 应用程序。与传统的 ASP.NET 框架相比，ASP.NET Core 更加轻量级、灵活和跨平台，可以在 Windows、Linux 和 MacOS 上运行，并且支持多种 Web 服务器，包括 Kestrel、IIS、Apache 和 Nginx 等。此外，ASP.NET Core 还提供了一系列的中间件，可以用来处理 HTTP 请求和响应、认证和授权、日志记录和异常处理等方面的问题。

ASP.NET Core 是一种跨平台、高性能、开源的 Web 开发框架，具有图 1-2 所示的亮点和优势。

图 1-2　ASP.NET Core 的亮点和优势

具体介绍如下。

- 跨平台：ASP.NET Core 可以在 Windows、Linux 和 MacOS 等多个操作系统上运行，开发者可以根据自己的需求选择适合的操作系统进行开发和部署。
- MVC 与 Web API 统一的编程模型：ASP.NET Core 将 MVC(模型-视图-控制器)与 Web API 统一在同一个编程模型中，简化了开发流程，开发者可以通过相同的控制器来处理 Web 应用和 API 请求，减少了重复的代码和配置。
- 依赖注入：ASP.NET Core 内置了依赖注入(DI)功能，简化了对象之间的依赖关系管理，提高了代码的灵活性和可维护性，便于开发者实现解耦。
- 可测试性：通过内置的依赖注入和模块化结构，ASP.NET Core 使得开发的应用更加容易进行单元测试和集成测试，提升了系统的可测试性。
- 开源：ASP.NET Core 是开源的，源代码托管在 GitHub 上，开发者可以自由查看、修改和贡献代码。这增加了开发者的透明度和灵活性。
- 模块化：ASP.NET Core 采用模块化设计，通过 NuGet 包管理，可以按需选择和添加必要的组件，减少了内存占用并提升了应用性能。模块化结构使得项目更易于维护和扩展。

在 2021 年的 Stack Overflow 开发者年度调查报告中，ASP.NET Core 被评为最受欢迎的开发框架之一。

2021 年最受欢迎的 Web 框架排名，如图 1-3 所示。

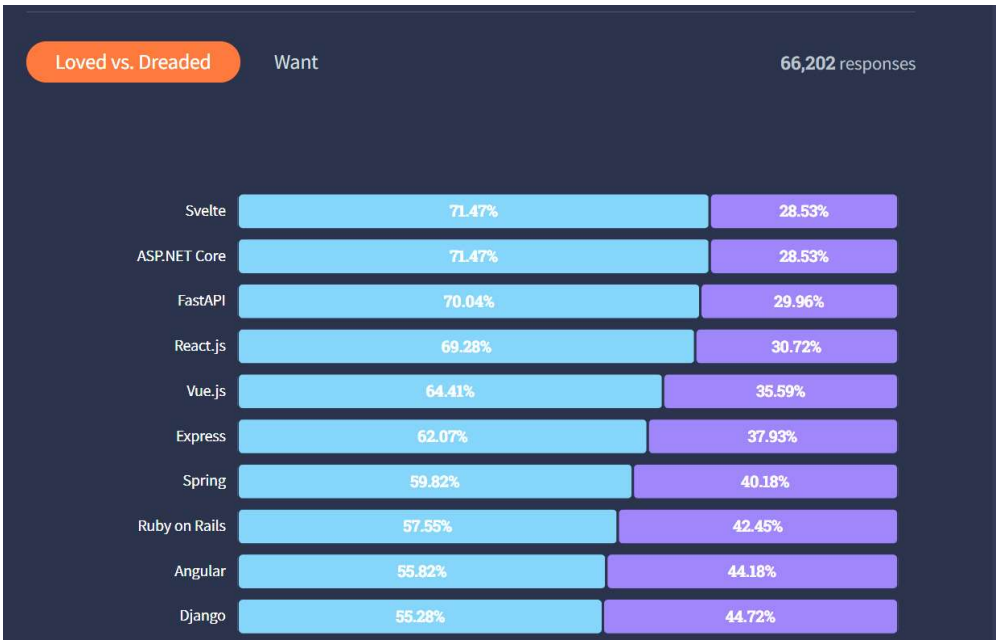

图 1-3　2021 年最受欢迎的 Web 框架排名

此外，Stack Overflow 上有众多使用 ASP.NET Core 进行编程的问答及讨论，这些资源为 ASP.NET Core 初学者和更加高级的开发人员提供了丰富的信息与经验分享。

1.1.3 任务实施

1. 环境下载

我们已经了解了 ASP.NET Core 的含义及作用，现在可以开始准备开发第一个项目了。若想开发 ASP.NET Core Web 应用，需先搭建好开发环境。在之前的内容中，我们提到了 ASP.NET Core 是一个开源的、高性能、跨平台的框架，所以它支持多种平台环境，目前比较常用的有 Windows 或 MacOS 系统，以及日渐受欢迎的 Docker 和 Linux 系统。在这里，我们主要讲解如何在 Windows 系统上进行环境安装，后续部署项目时，再对 Linux 的部署操作进行说明。若支持跨平台，则每一个平台都要搭建一个可以运行 ASP.NET Core 项目的环境，我们称之为运行时(Runtime)，以帮助我们能够在各个平台上顺利运行。如果要开发项目，还需要安装 ASP.NET Core SDK 组件，它包含了运行时及其他的基础库和构建 ASP.NET Core 应用程序的命令行界面(Command Line Interface，CLI)。SDK 可以安装到 Windows、Mac、Linux 等平台上。如果使用的是 Visual Studio 2019/2022 版本，则可以不用安装。本书采用最新的开发工具 Visual Studio 2022，可以在浏览器中搜索“下载 Visual Studio 2022”进行获取，也可以直接从官方网站下载，地址为：https://dotnet.microsoft.com/zh-cn/download，进入官网之后，界面如图 1-4 所示，我们可以根据自己的操作系统选择所需的安装包和版本。

图 1-4　Visual Studio 下载界面

Visual Studio 2022 常见的版本有社区版(Community)、专业版(Professional)和企业版(Enterprise)，具体功能如下。

- Visual Studio 2022 Community：适用于个人开发者及小规模团队，提供基本的开发工具和功能，免费使用。
- Visual Studio 2022 Professional：适用于中小型团队及专业开发者，提供更多的开发工具和功能，包括代码分析、测试工具等。
- Visual Studio 2022 Enterprise：适用于大型企业及团队，提供高级的开发工具和功能，包括团队协作、应用程序生命周期管理等。

我们可根据自身需求选择版本，个人可以使用社区版。

2. 环境安装

安装包下载完成之后，直接双击 VisualStudioSetup.exe 开始安装，稍等片刻会出现图 1-5 所示的功能选择界面，这里选择 ASP.NET 和 Web 开发、.NET 桌面开发即可，Visual Studio 可以通过仅选择所需的组件来节省安装时间和磁盘空间，并且始终可以根据需要随时以增量方式添加更多组件。

安装过程大约需要 30 到 40 分钟，具体时间取决于计算机的配置及网络状况。安装完后，我们就可以开始一个示例项目了。

图 1-5　Visual Studio 2022 功能选择

构建 ASP.NET Core 应用

1.2.1　任务描述

一般，学习新语言或新框架都是从 Hello World 示例项目开始，下面我们一起来开始构建第一个 ASP.NET Core 项目。

1.2.2　任务实施

打开安装好的 Visual Studio 2022，在起始页的对话框中单击【创建新项目】，如果已创建过项目，左侧【打开最近使用的内容】会显示相应的项目列表，操作如图 1-6 所示。

在弹出的【创建新项目】对话框的搜索框中，输入“Asp Net Web”关键字搜索，或者直接通过如图 1-7 所示的下拉框进行筛选。

图 1-6　创建新项目

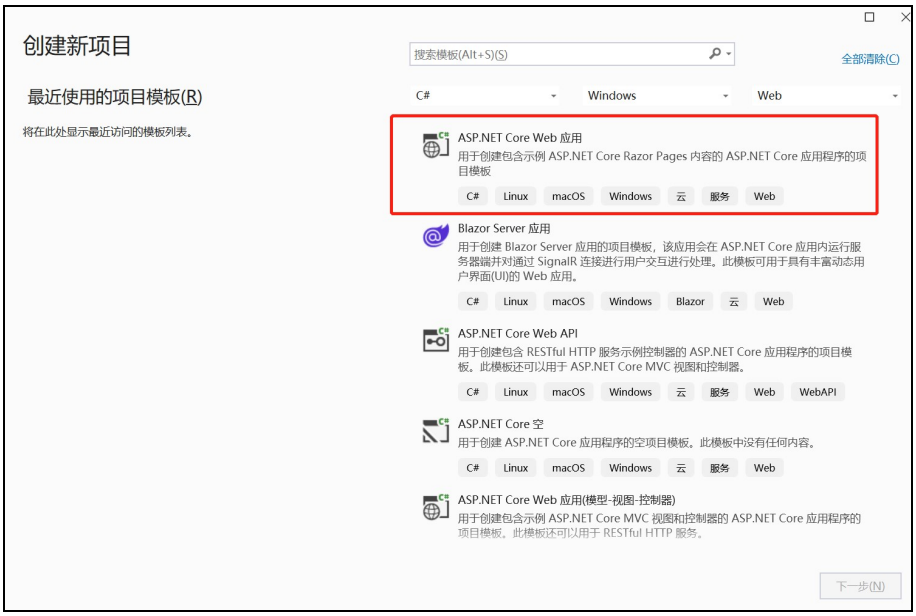

图 1-7　.Net Core 模板搜索与选择

单击【下一步】按钮，弹出【配置新项目】对话框，输入【项目名称】并选择文件存放的【位置】，我们可以自定义【解决方案名称】，默认与项目名称一致，也可以不一致，如果选择或填写错误，也可以单击【上一步】按钮，界面如图 1-8 所示。

单击【下一步】按钮，弹出【其他信息】的输入和选择对话框，由上至下，分别是框架、身份验证类型、配置 HTTPS 和启用 Docker。

- 框架：选择框架和版本，默认是.NET 6.0(长期支持)。
- 身份验证类型：集成了微软提供的认证方案，包括 Windows 认证、匿名认证等，基于 Identity 类库的实现，可以取消选择。

- 配置 HTTPS：为项目支持 HTTPS 访问做准备，可以取消选择。
- 启用 Docker：为项目配置 Docker 容器化，生成镜像做准备，可以暂时取消选择。

其他信息配置界面如图 1-9 所示。

单击【创建】按钮，默认初始化项目正式完成。单击菜单栏中的【调试】按钮，选择【开始执行不调试】，即可看到默认生成的 Web 页面，如图 1-10 所示。

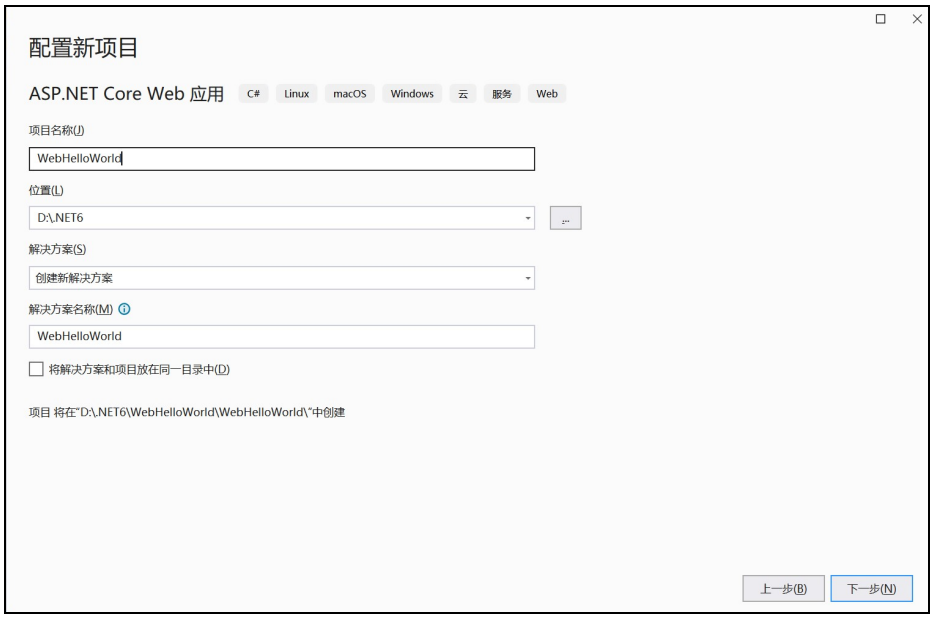

图 1-8　配置项目

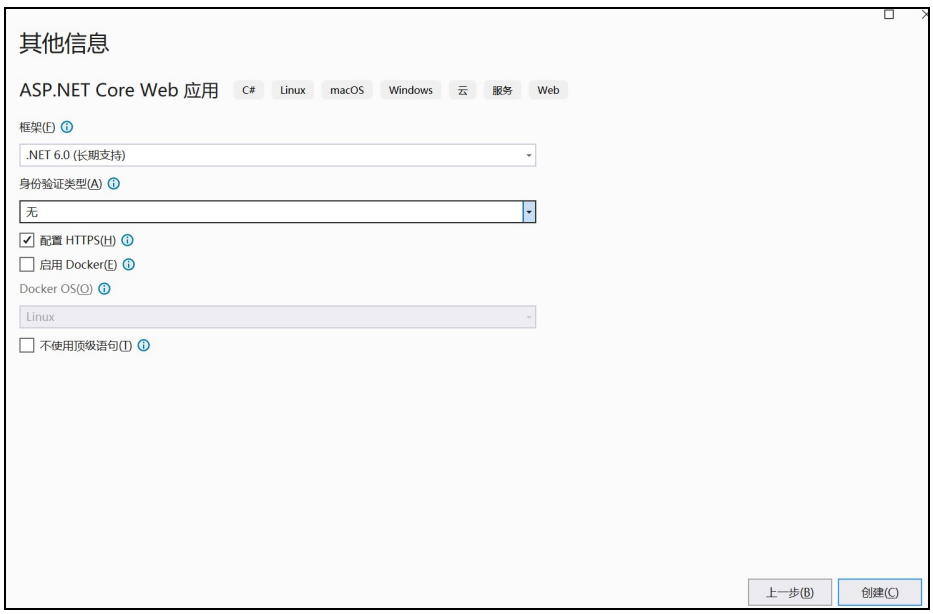

图 1-9　其他信息配置界面

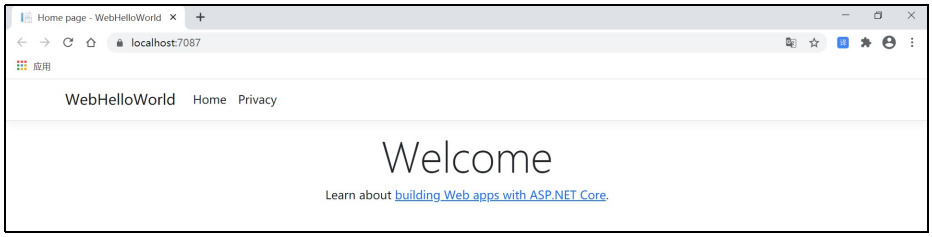

图 1-10　Web 页面运行效果

认识 ASP.NET Core 项目结构

1.3.1　任务描述

前面我们掌握了如何使用 Visual Studio 2022 来创建一个 ASP.NET Core 项目，接下来我们一起来了解一下 ASP.NET Core 项目文件的目录信息。通过对项目结构的认识，我们可以进一步理解 ASP.NET Core 的运行机制和项目框架。

1.3.2　知识学习

ASP.NET Core 项目结构如图 1-11 所示，其目录/文件说明如表 1-1 所示。

图 1-11　ASP.NET Core 项目结构

表 1-1　ASP.NET Core Web 目录/文件说明

目录/文件	说明
appsettings.json	用于存储应用程序配置信息的 JSON 格式文件。该文件通常包含一组键值对，它们用于为应用程序提供各种自定义配置选项
Program.cs	初始化应用程序的所有信息，启动网站应用程序，定义程序配置
wwwroot	用于存储 Web 应用程序和 Web 站点的各种静态资源(如图像、CSS、JavaScript、HTML 文件等)的默认位置。在运行时，这些文件可以被浏览器直接访问和加载，以便为用户提供更丰富、更优秀的 Web 内容
Properties	存放一些.json 文件，用于配置 ASP.NET Core 项目
Pages	用于存储 Razor Pages 的默认目录。Razor Pages 是一种基于类似 MVC 模式组织代码的 Web 应用程序模型，通常用于轻量级的 Web 应用程序和 Web 页面的构建

1. Program.cs 文件

在.NET 6 中，虽然依然支持使用 startup.cs 文件来配置应用程序的服务和中间件，但默认情况下不再创建该文件。相反，ASP.NET Core 在.NET Generic Host 中提供了更简单、灵活的方式来配置应用程序的服务和中间件。在.NET 6 中，使用新的通用主机模式，可以通过 Program 类中的 Main()方法来进行应用程序的启动和配置。具体来说，我们可以在 Program 类中调用 CreateHostBuilder()函数来创建主机实例，并在主机实例上调用 ConfigureServices()和 Configure()方法来配置应用程序的服务和中间件。

Program.cs 文件中的代码如下所示。

```
var builder = WebApplication.CreateBuilder(args);
// 添加服务
builder.Services.AddRazorPages();
var app = builder.Build();
// 中间件配置
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
}
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();
app.MapRazorPages();
app.Run();
```

总之，在.NET 6 中，主机程序从 Program.cs 开始，我们可以使用 CreateHostBuilder()方法来创建主机实例，并在应用程序上下文中直接配置服务和中间件，而不需要显式创建和配置 startup.cs 文件。

2. 依赖项

在.NET 6 中，依赖项(Dependency)是指程序开发中用到的其他的程序集或库。例如，若我们在编写一个 ASP.NET Core Web 应用程序并需要使用 EF Core ORM 来进行数据库操作时，则需要将 Microsoft.EntityFrameworkCore NuGet 包添加为依赖项，以便从代码中引用和调用 EF Core 相关的类及方法。

依赖项管理是现代软件开发过程中不可或缺的一部分。使用各种第三方库和框架可以显著提高代码复用性与开发速度，同时也很容易出现版本冲突、组件更新等问题。因此，在.NET 6 中，可以使用以下几种方法来有效地管理依赖项。

- NuGet 包管理器：NuGet 是.NET 生态系统中最常用的包管理器，它可以让我们方便地添加、移除和更新相关的组件。在 Visual Studio 中，可以使用 NuGet 包管理器 UI(NuGet Package Manager UI)或包管理器控制台(Package Manager Console)来搜索、安装和更新 NuGet 包。
- .NET SDK 项目文件：.NET SDK 项目文件是.NET 6 新的项目文件格式之一，它使用 XML/MSBuild 语法，并将项目、目录和项目的基本信息全部集成在同一个文件中。通过编写.csproj 文件，我们可以清楚地定义项目中所有的依赖项和构建选项。
- .NET 工具：.NET 工具是指轻量级的 CLI(Command Line Interface)扩展，它们可以用于完成各种开发相关的任务。例如，可以使用 dotnet add package 命令添加 NuGet 包，使用 dotnet build 命令来构建代码，使用 dotnet test 命令进行单元测试等。

总之，依赖项在.NET 6 的开发中具有重要作用，通过优秀的依赖项管理方式可以提高应用程序的代码质量、可维护性和可扩展性，推动整个开发过程工作效率的提升。

3. Properties 目录

Properties 目录用于存放程序集信息、运行配置、内部资源等文件。该目录在创建时，会默认创建一个 launchSettings.json 文件，该文件包含了一些程序启动时的信息。代码如下：

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:45697",
      "sslPort": 0
    }
  },
  "profiles": {
    "WebHelloWorld": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "http://localhost:7087",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

在.NET 6 中，Properties 文件夹主要用于指定应用程序的属性和设置相关信息。

具体来说，Properties 文件夹常包括以下文件。

- launchSettings.json：它定义应用程序在运行时的启动配置信息，如 Web 应用程序的 URL 地址、环境变量等。
- assemblyinfo.cs：它存储有关应用程序的元数据信息，包括版本号、版权信息等。
- resources.resx：它是一个资源管理器，用于存储所有应用程序所使用的字符串及其他资源，便于国际化和本地化。

另外，还可以通过添加自己的Settings.settings文件以实现自定义设置。此外，Properties 文件夹还可以扩展到其他自定义配置文件，以保存应用程序所需的所有属性信息。

总之，Properties 文件夹在 ASP.NET Core 实现中提供了一种集中管理应用程序所需信息的方式，并帮助开发人员更好地进行应用程序的管理及代码的维护。

素养园地

科技之光，照亮未来之路

党的二十大报告明确提出，教育、科技和人才是社会主义现代化建设的基础性支撑，必须深入实施科教兴国战略、人才强国战略、创新驱动发展战略。这一战略方针为我国的科技发展和创新提供了坚实的政策保障。正是在这种战略引领下，许多企业开始在自主创新和前沿技术领域取得突破，其中，深圳微芯生物科技有限公司便是一个典型代表。

微芯生物依托生命科学和新药研发领域的最新进展，搭建了“基于化学基因组学的集成式药物发现与早期评价平台”。这一平台不仅整合了分子医学、计算机辅助药物设计和高通量筛选等技术，还有效降低了新药研发中的风险，推动原创新药的诞生。通过这一平台，微芯生物为全球药物研发带来了新的突破，并展示了中国企业在全球创新药物领域的竞争力。

通过微芯生物的实践，我们更加深刻地认识到，只有通过不断的创新和探索，才能实现科技进步并推动社会发展。树立正确的科技发展观，培养创新意识，对于推动科技进步及满足人类健康需求具有深远的意义。

单元小结

- ASP.NET Core 是一个开源的、高性能、跨平台的 Web 应用程序框架，它是 ASP.NET 的下一代版本。该框架可以在 Windows、MacOS 和 Linux 等操作系统上运行。
- ASP.NET Core Web 应用采用 Program.cs 文件来加载配置和启动代码。
- 搭建 ASP.NET Core 开发环境。
- ASP.NET Core 具有轻量级、高性能、模块化和可测试性等特点。它的出现体现了科技创新的重要性，为企业创新及提高生产力和市场竞争提供了强大的支持。
- 学习 ASP.NET Core Web 项目各个文件的意义，可以帮助我们培养系统思维、协作精神、适应性和可持续发展的理念。

单元自测

■ 选择题

1. 下列不是 ASP.NET Core 的特征的是(　　)。

A. 可以运行在 Windows、Linux 和 MacOS 操作系统上

B. 它是一种跨平台开发框架

C. 它仅能使用 C#语言进行编程

D. 采用了与前身 ASP.NET 完全不同的架构

2. 下列中是 ASP.NET Core Web 应用程序使用到的页面类型的是(　　)。

A. dll pages　　B. css pages　　C. razor pages　　D. jquery pages

3. 在 ASP.NET Core 项目的文件夹中，下列中包含了项目的静态资源文件的文件夹是(　　)。

A. views　　B. controllers　　C. models　　D. wwwroot

4. 下列中是 ASP.NET Core Web 应用的启动文件的是(　　)。

A. Web.Config　　B. Globle.axax　　C. Startup.cs　　D. Program.cs

5. 在 ASP.NET Core 项目的文件夹中，下列中包含了项目的视图页面的文件夹是(　　)。

A. Models　　B. Properties　　C. Pages　　D. Services

■ 问答题

1. 什么是 ASP.NET Core？
2. ASP.NET Core 的主要特点有哪些？
3. .NET Core 与 ASP.NET Core 的区别是什么？

上机实战

■ 上机目标

掌握.NET Core 开发环境的安装和配置。

■ 上机练习

练习：搭建.NET Core 开发环境。

【问题描述】

通过官网下载 Visual Studio 2022 的任意一个版本，并安装 ASP.NET 和 Web 开发、.NET 桌面开发所需要的组件。

【问题分析】

(1) 通过官网下载 Visual Studio 2022 的任意一个版本。

(2) 安装 ASP.NET 和 Web 开发、.NET 桌面开发所需要的组件。

【参考步骤】

请参考本单元【任务 1.1　安装和配置环境】～【1.1.3　任务实施】内容中的步骤完成练习。

单元

二

第一个ASP.NET Core MVC应用

课程目标

项目目标

- 搭建任务管理系统基本架构
- 完成任务管理系统登录页面
- 任务管理系统登录功能实现

技能目标

- 理解 MVC 的开发模式
- 掌握基于模型-控制器-视图的程序编写
- 掌握 Razor 视图语法
- 熟练掌握开发环境的使用

素养目标

- 培养高效的沟通技巧和协作能力
- 养成精益求精的职业态度和做事风格
- 具备程序员的基本素养

简介

通过前面的介绍，我们已经知道 ASP.NET Core 的作用，以及如何创建一个 ASP.NET Core Web 应用。微软在 ASP.NET Core 的基础上开发了 ASP.NET Core MVC 和 ASP.NET Core Web API 两个框架。本单元我们将主要讲述如何使用 MVC 开发模式开发 ASP.NET Core MVC 项目，深入了解其开发模式和原理，提高程序的可扩展性和可维护性，保证项目后期的扩展和更新。在深入了解的过程中，我们应养成分工明确、爱岗敬业、勇于进取的职业精神。

TaskMaster 任务管理系统

2.1.1 任务描述

在传统的开发模式中，应用程序常常被耦合在一起，这使得代码难以维护和扩展。MVC 模式通过将用户界面、数据处理和用户交互逻辑分离，提供了一种更具组织性和可维护性的设计思想，同时也使得小团队的开发工作更加容易协同操作。本次任务我们将认识什么是 MVC 及 MVC 的相关知识，并创建第一个 ASP.NET Core MVC 项目——任务管理系统，实现简单的登录功能。

2.1.2 知识学习

1. MVC 模式

在 20 世纪 80 年代，MVC 模式最初由 Xerox PARC 实验室的 Trygve Reenskaug 提出，旨在解决 Smalltalk 应用程序的设计问题，后来被广泛应用于各种编程语言和框架中，成为一种经典的软件架构模式。如图 2-1 所示，MVC 将一个应用程序分为 Model(模型，即应用程序的数据和业务逻辑)、View(视图，即用户界面)和 Controller(控制器，即处理用户交互逻辑) 3 个相互协作的部分。

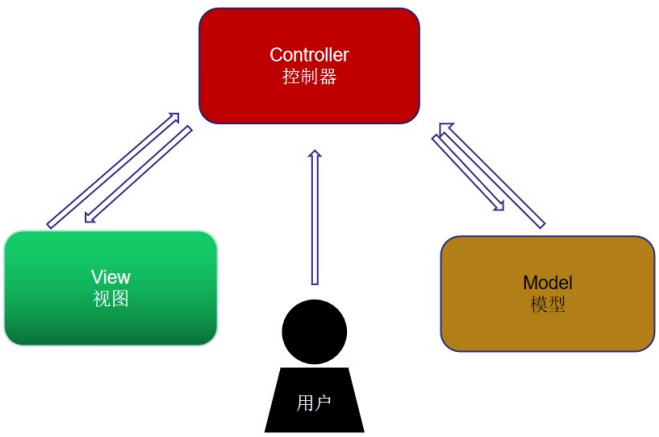

图 2-1　MVC 架构

在 MVC 模式应用程序的 3 个组件中，Model 负责表示应用程序中的数据和业务逻辑，View 负责显示应用程序的用户界面，而 Controller 则负责协调 Model 和 View 之间的交互，并处理用户的输入事件。这 3 个组件通过特定的规则进行通信，以实现应用程序的运行和交互。其中，Model 和 View 是相互独立的，它们没有直接的联系，而是通过 Controller 进行交互。这种结构使得代码易于维护和修改，同时也提高了开发效率和代码的可复用性。

从 ASP.NET MVC 到 ASP.NET Core MVC，最大的改进在于其跨平台性能的增强和开发体验的提高。

2. ASP.NET Core MVC

ASP.NET Core MVC 是 Microsoft 推出的基于.NET Core 的一种 Web 应用程序开发框架，是 ASP.NET MVC 框架的后续版本。在 ASP.NET Core MVC 中，控制器由 Controller 类实现，视图一般是扩展名为 cshtml 的文件，模型则是只有属性的普通 C#类，控制器类一般直接或间接继承自 Controller 类，控制器的名字一般以 Controller 结尾，并且放到 Controllers 文件夹下，模型则放到 Models 文件夹下，视图文件则放在 Views 文件夹下。按照这些规则存放对文件管理非常方便，这就像是一个约定好的规则。

该框架的主要目的是增强开发人员的工作效率和开发体验，并且在性能、安全性、可靠性和跨平台方面都有所改进。相对于早期版本的基于.NET Framework 的 ASP.NET MVC，ASP.NET Core MVC 平台不一定需要使用 IIS 来运行，可自由选择与 Kestrel 或其他支持 ASP.NET Core 应用程序的 Web 服务器集成使用，其可以在多种操作系统上运行，具有跨平台的能力。

ASP.NET Core MVC 框架还提供了许多其他功能，如数据绑定、表单验证、区域路由、安全认证、缓存等，使得开发人员能够更加轻松地构建高质量的 Web 应用程序。同时，ASP.NET Core 还支持依赖注入等面向对象编程的最佳实践，允许开发人员写出更加优雅和可测试的代码。

3. ASP.NET Core MVC 模式的特性

随着软件项目复杂度的增加及软件项目分工的细化，前后端分离已经成了主流的开发模式，ASP.NET Core MVC 既支持基于视图的 MVC 模式开发，也支持 Web API 和 Razor Pages 开发。在 ASP.NET Core MVC 开发模式下，后端人员也需要写一部分前端代码。

作为一种技术框架，ASP.NET Core MVC 具有以下优点。

(1) 分离关注点：MVC 模式将应用程序分为 Model、View 和 Controller，它们之间相互独立。Model 明确了应用程序的数据和业务逻辑，View 管理用户界面，而 Controller 负责协调 Model 和 View 之间的交互。这使得代码的逻辑更加清晰，方便维护和修改。

(2) 可扩展性良好：由于 ASP.NET Core 的模块化设计，应用程序中每个部分都可以进行自我管理并封装在其自己的 DLL 中。这种分层架构基于接口解耦，能够使开发人员轻松地添加新的功能或改进现有功能，而不需要对整个应用程序进行大规模修改。

(3) 可定制性强：ASP.NET Core MVC 提供了丰富的插件和组件，开发人员可以针对特定需求进行自定义开发，并完整地支持依赖注入(DI)，让代码更加灵活和可定制。

(4) 高度灵活性：ASP.NET Core MVC 引入了特性路由(Attribute Routing)机制，使得我们能够更加灵活地定义请求与响应之间的关系，并能使用 RESTful 风格的 API 进行开发。

(5) 支持多种客户端技术：ASP.NET Core MVC 可以轻松集成其他不同的客户端技术，如

Angular、React 等，实现前后端分离的设计模式，开发人员可以使用任何其他前端工具栈来构建动态 Web 应用程序。

在具备以上优点的情形下，ASP.NET Core MVC 同时具有如下一些缺点。

(1) 较为复杂：相对于 ASP.NET Web Forms 模式而言，MVC 模式的应用程序结构较为复杂，需要掌握更多的概念和技术，处理客户端和服务端之间的通信也更加复杂，因此入门较为困难。

(2) 学习曲线陡峭：ASP.NET Core MVC 模式是一种全新的开发模式，在学习过程中需要掌握其核心思想、概念及与生态系统相关联的各种组件和工具，并且往往需要在多个领域(如数据库、Web 服务、HTML 网页等)中有深入的了解才能够进行开发。

(3) 开发效率相对较低：在许多偏向快速上线的项目中，使用 ASP.NET Core MVC 虽然可行但开发效率可能没有其他框架高，这主要是由于应用程序的分层和拆分导致的额外复杂性所决定的。

(4) 代码量相对较大：相对其他框架而言，使用 ASP.NET Core MVC 编写应用程序的代码量有可能会比较庞大，随着功能增加和应用程序规模扩大，代码量会进一步增加。

总体而言，ASP.NET Core MVC 是一款优秀的 Web 框架，尤其适合分工合作。

2.1.3 任务实施

1. 项目实现

打开【Visual Studio 2022】对话框，单击【创建新项目】选项，选择【ASP.NET Core Web 应用(模型-视图-控制器)】选项，如图 2-2 所示。

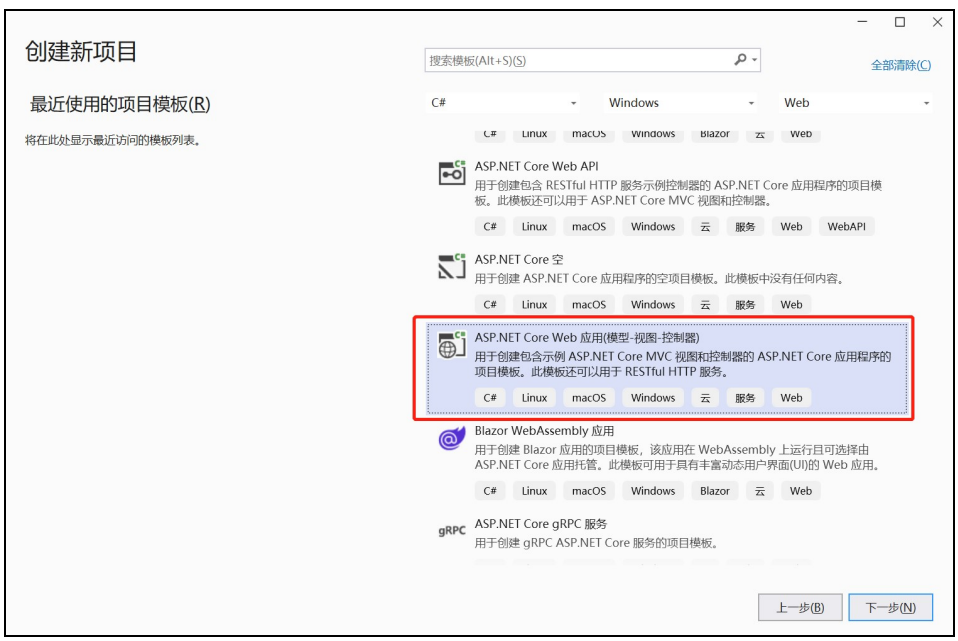

图 2-2　创建 MVC 项目

选择模板之后，单击【下一步】按钮，输入【项目名称】和【解决方案名称】，如图 2-3 所示。默认情况下，解决方案名称会与项目名称一致，可以选择修改，也可以保持默认。

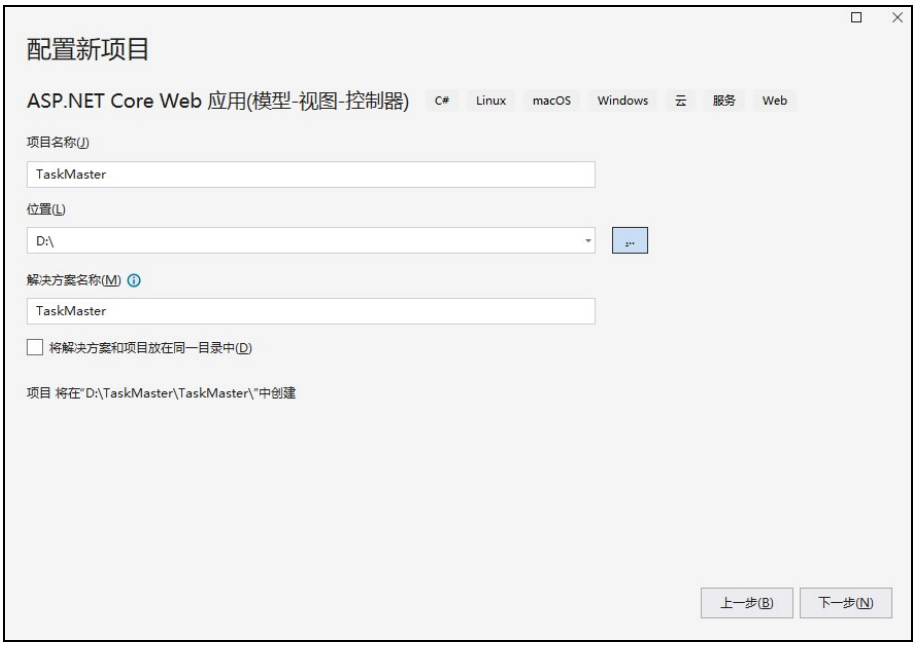

图 2-3　设置项目名称和解决方案名称

确定项目名称和解决方案名称后，单击【下一步】按钮，选择框架为【.NET 6.0】,【配置HTTPS】可根据需求选择，如图 2-4 所示，然后单击【创建】按钮。

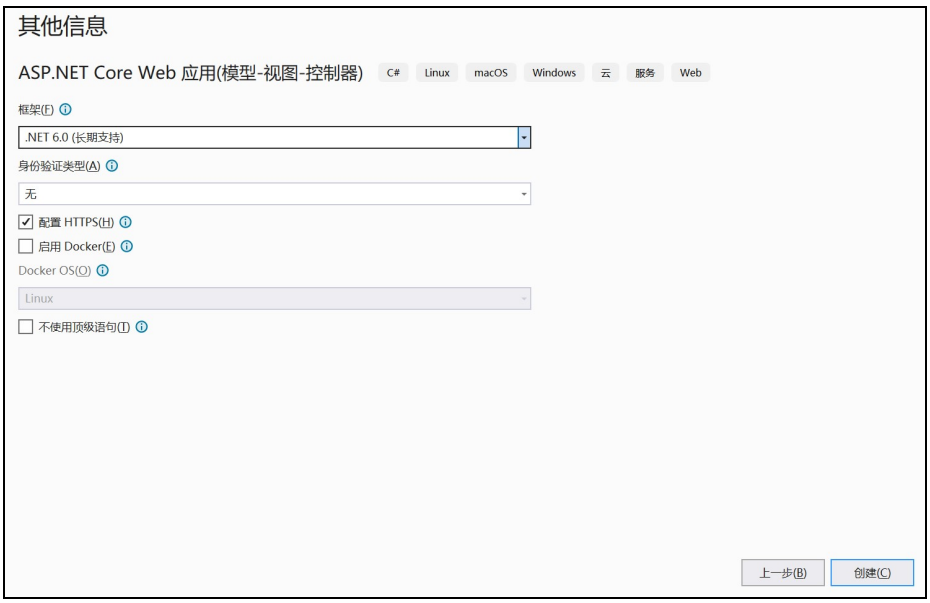

图 2-4　项目配置

创建完成之后，可在工具右侧看到如图 2-5 所示的项目结构。

在上述文件结构中，编辑器会自动创建好 Controllers、Models 和 Views 等文件夹。现在，我们需要实现 TaskMaster 任务管理系统的登录页面。

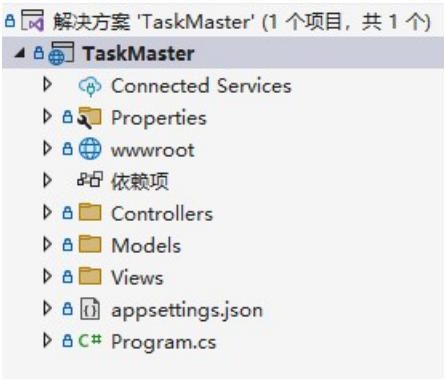

图 2-5　MVC 项目结构

首先，在项目的 Models 文件夹中创建一个名为 LoginUser.cs 的类文件，然后在该类中存放用户的登录名与密码信息，代码如下所示。

```
namespace TaskMaster.Models
{
    public class LoginUser
    {
        /// <summary>
        /// 用户名
        /// </summary>
        public string UserName { get; set; }
        /// <summary>
        /// 密码
        /// </summary>
        public string LoginPwd { get; set; }
    }
}
```

其次，在项目结构的 Controller 文件夹中，选择【右键】→【添加】→【控制器】选项，在弹出的对话框中，选择【MVC 控制器-空】选项，如图 2-6 所示。

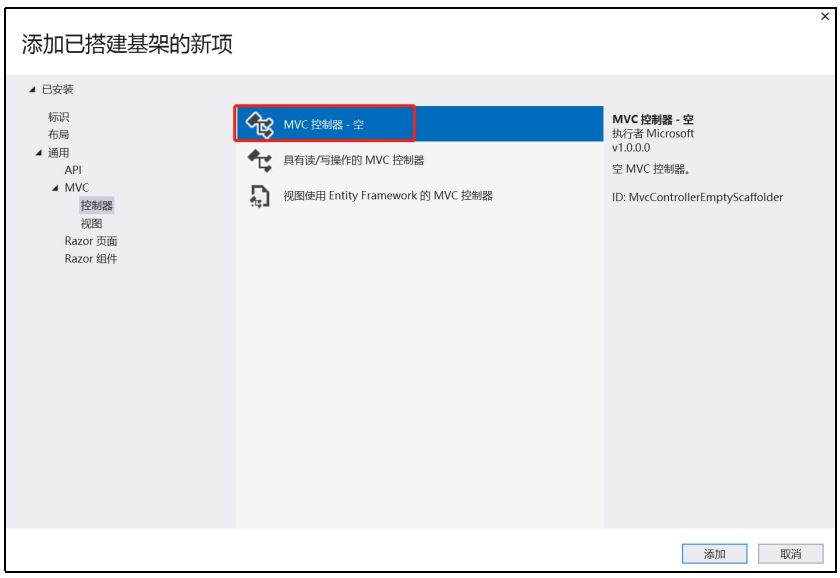

图 2-6　添加控制器

单击【添加】按钮，在弹出的对话框中更改控制器的名称，如图 2-7 所示。需要注意的是，MVC 中的控制器名称只需要修改前半部分，后半部分保持以 Controller 结尾，遵循“约定大于配置”的原则，修改后单击【添加】按钮。

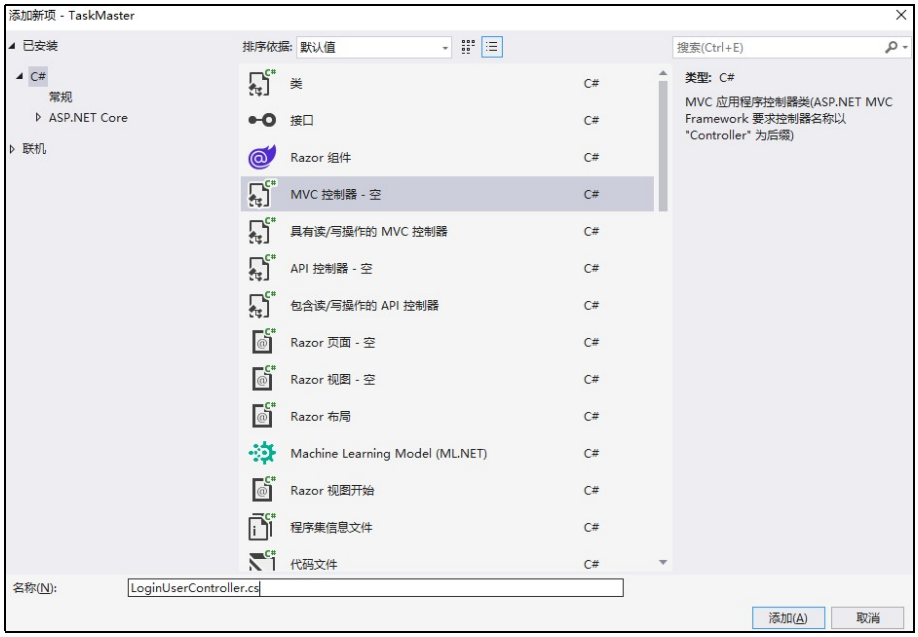

图 2-7　修改控制器名称

在新添加控制器中编写代码，如下所示。

```
using Microsoft.AspNetCore.Mvc;
using TaskMaster.Models;
namespace TaskMaster.Controllers
{
    public class LoginUserController : Controller
    {
        [HttpGet]
        public IActionResult Index()
        {
            return View();
        }
        [HttpPost]
        public IActionResult Index(LoginUser user)
        {
            if (user.UserName.Equals("admin") && user.LoginPwd.Equals("pwd123"))
            {
                return Content("登录成功"); //跳转到登录成功页面
            }
            return View();
        }
    }
}
```

完成控制器代码之后，接下来我们开始编写视图。将鼠标光标聚焦在 Index 方法上，选择【右键】→【添加视图(D)】选项，在弹出的新对话框中，选中【Razor 视图-空】选项，单击【添加】按钮，如图 2-8 所示。

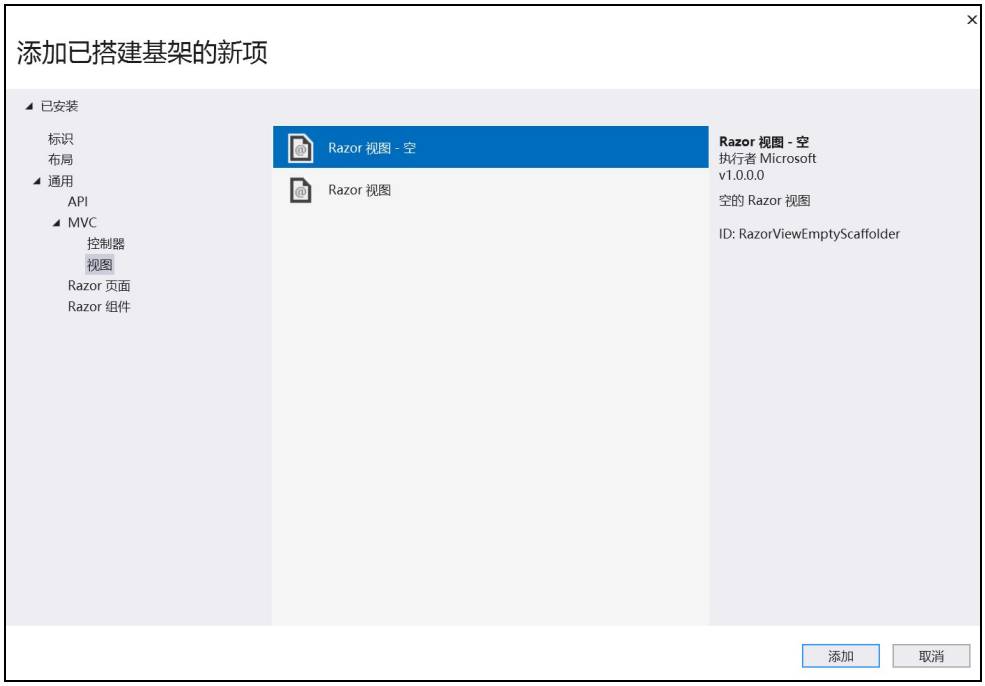

图 2-8　创建视图

视图创建好后，项目的 Views 文件夹中会自动生成 LoginUser 控制器文件夹，同时在该文件夹下还会自动创建一个 Index.cshtml 文件，该文件就是登录页面对应的视图文件。编写视图文件代码如下。

```
@model TaskMaster.Models.LoginUser;
<h2>登录</h2>
<form asp-action="Index" method="post" class="form-horizontal" role="form">
    <div asp-validation-summary="All" class="text-danger"></div>
    <div class="form-group">
        <label asp-for="UserName" class="col-md-2 control-label">
            用户名
        </label>
        <div class="col-md-5">
            <input asp-for="UserName" class="form-control" />
            <span asp-validation-for="UserName" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="LoginPwd" class="col-md-2 control-label">
            密码
        </label>
        <div class="col-md-5">
            <input asp-for="LoginPwd" type="password" class="form-control" />
            <span asp-validation-for="LoginPwd" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-5">
            <input type="submit" class="btn btn-primary" value="登录" />
        </div>
    </div>
</form>
```

完成上述代码之后，可以直接单击菜单栏中的【调试】按钮，选择【开始执行不调试】选项运行查看结果。此时浏览器会打开 https://localhost:7068/这类网址，如果我们需要访问刚刚自定义的视图，则需要在浏览器中直接访问 LoginUser 视图中的 Index 方法，写法为/LoginUser/Index，即将地址改成 https://localhost:7068/LoginUser/Index 进行访问，显示结果如图 2-9 所示。

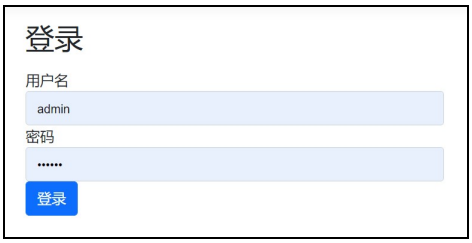

图 2-9　登录运行效果

2. MVC：约定大于配置

在上述登录功能实现的过程中，我们发现了一些特殊的地方。例如，在添加控制器类时后缀保持 Controller 不变，而且都放在 Web 文件夹下的 Controllers 文件夹中，控制器类继承 Controller 基类；视图文件必须放在名称为 Views 的文件夹中，并且在该文件夹下按照控制器名称进行组织；以下画线开头命名的视图一般作为布局视图，放在 shared 文件夹下面，这些我们称为“约定大于配置”。ASP.NET Core MVC 中的“约定大于配置”是一种设计哲学，旨在减少开发人员的代码量。简而言之，“约定大于配置”是指使用默认设置和惯例(约定)来降低需要进行显式配置的数量。通过利用这些默认约定，我们不需要在应用程序中显式地指定每个功能和组件如何工作，这样可以简化任务并使代码更加易于阅读和维护。如果我们需要修改或扩展默认约定，大多数情况下可以通过调整它们的行为方式来满足特定需求。然而，即便如此，我们仍然可以从众多默认值中获得诸多帮助。

3. 热重载

在上述运行效果中，使用过之前开发工具的开发者应该都知道，如果中途修改了服务器代码，则需要重启服务器再查看代码，而在 Visual Studio 2022 中新添加了热重载的功能，其是一种开发工具的功能，允许在不停止应用程序的情况下对代码更改进行实时查看。启用热重载后，可以在编辑器中进行代码更改，然后将更改内容立即反映到正在运行的应用程序中，无须重新启动应用程序。这为开发人员带来了极大的便利，并提高了开发效率。

ASP.NET Core 是一个支持热重载的框架，其原理是将应用程序加载到 ASP.NET Core 管道的进程运行时中，然后通过监视本地文件系统(如项目的源文件)来检测任何更改。如果更改了代码，它会编译并重新加载“运行时”代码，以便我们能够立即看到更改的结果。使用 ASP.NET Core 热重载功能进行开发可节省大量时间，尤其是在需要频繁更改代码或 UI 设计的开发过程中。这项功能非常适合进行快速迭代和测试。但需要注意的是，在生产环境中我们应该禁用此特性，因为在这种模式下运行应用程序可能会导致性能下降和其他问题。

在 Visual Studio 2022 中，热重载的用法很简单，只需要单击 Visual Studio 工具栏中的热重载图标(如图 2-10 所示)，修改的代码即可生效。

图 2-10　热重载图标

4. Razor 视图语法

在上述代码中，@model TaskMaster.Models.LoginUser;表明该视图文件是接收 LoginUser 类数据的强类型视图，控制器传递过来的对象在 cshtml 视图中可以用 Model 属性来获取。cshtml 文件中大部分是普通的前端代码，其中以@开头的为标准的 C#表达式，代表把 C#表达式的内容传输到表达式所在的位置。在上述代码中以@开头的语法格式我们称为 Razor 语法。

Razor 语法中的“@”符号是 Razor 视图中一个非常重要的符号，它被定义为 Razor 服务器代码块的开始符号。如果我们希望在页面中输出一个变量，则可以使用如下代码；如果要输出“@”符号，则可以使用“@@”符号。

```
<span>@userName</span>
<span>@sex </span>
```

在 Razor 视图中，我们可以采用“@{code}”来定义一段代码块，如下所示。

```
@{
int a=5；
        int b=10;
int sum=a+b;
@sum;
}
```

Razor 支持代码混写，如下所示。

```
@if(value%2==2){
<p>这个数是偶数</p>
}
```

同时，Razor 还支持 c#和 html 的注释，如//单行注释、/*多行注释*/及<!-- htmlcode-->，在 Razor 中也可以使用“@*多行注释*@”这种写法来进行注释。另外，在 Razor 中还有许多常用的指令，可以在后续的学习中逐步熟悉。

素养园地

分工合作，实事求是，精益求精

党的二十大报告明确提出了大会主题：“高举中国特色社会主义伟大旗帜，全面贯彻新时代中国特色社会主义思想，弘扬伟大建党精神，自信自强、守正创新，踔厉奋发、勇毅前行，为全面建设社会主义现代化国家、全面推进中华民族伟大复兴而团结奋斗。”这句话深刻体现了中国特色社会主义道路的核心精神。正如报告所强调的，旗帜决定方向，党的信仰和理论方向将引领我们朝着全面建设社会主义现代化国家的目标前进。

在这一伟大目标指引下，全国各地积极践行新时代中国特色社会主义思想，推动社会经济各领域的蓬勃发展。例如，江西省宜春市万载县通过采取间作套种、林下种植等农业模式，因地制宜地发展迷迭香、百合、香薷、艾草等中草药种植。这不仅为农民带来了可观的收入，也为实现乡村振兴战略贡献了力量。

在这一过程中，万载县的农民与企业通过密切合作，解决了技术、成本、产品质量等一系列问题，展现了“实事求是”和“精益求精”的精神。通过团队合作，他们不断改进生产技术，提

高了产品质量和市场竞争力，进一步推动了地方经济的高质量发展。

这一实践充分体现了党的二十大报告提出的合作共赢、实事求是的精神。只有通过分工合作、互相支持、发挥个人优势，团队才能克服困难，共同实现目标。正如新时代中国特色社会主义思想所强调的，只有坚持不懈、不断创新，才能在实践中取得更大成就，最终推动社会和谐进步。

单元小结

- MVC 是一种常见的 Web 应用程序开发模式，其将应用程序分为 Model、View 和 Controller 3 个部分。
- 创建 ASP.NET Core MVC 项目。
- Razor 语法设计简洁，易于编写，它允许开发人员直接在 HTML 中编写服务器端代码，无须使用特殊的标记语言，使代码更加清晰和易于维护。
- MVC 的分工合作模式彰显了分工协作、分工明确的实事求是、精益求精的工作态度和工作作风。

单元自测

选择题

1. ASP.NET Core MVC 中视图文件通常存放在(　　)目录下。

   A. wwwroot　　B. Controllers

   C. Views　　D. Models

2. ASP.NET Core MVC 中的 Action 是(　　)。

   A. 控制器内响应请求的方法　　B. 模型类的工厂方法

   C. 普通方法　　D. 视图呈现方法

3. 在 Razor 视图中，以下用于标记代码块的字符是(　　)。

   A. #　　B. $

   C. @　　D. %

4. 在 Razor 视图中定义循环结构的语句是(　　)。

   A. @(for…)　　B. <%foreach…%>

   C. <do..while>　　D. <%=for(…)%>

5. 在 ASP.NET Core MVC 中，以下可以在视图中接收强类型数据的类型是(　　)。

   A. ViewData　　B. ViewBag

   C. Model　　D. TempData

上机实战

上机目标

- 掌握 ASP.NET Core MVC 项目的创建。
- 掌握 Razor 视图的基本语法。

上机练习

练习：创建一个 ASP.NET Core MVC 项目，实现如图 2-11 所示的【添加课程】界面，输入课程信息后，单击【添加课程】按钮，返回所添加的课程信息，如图 2-12 所示。

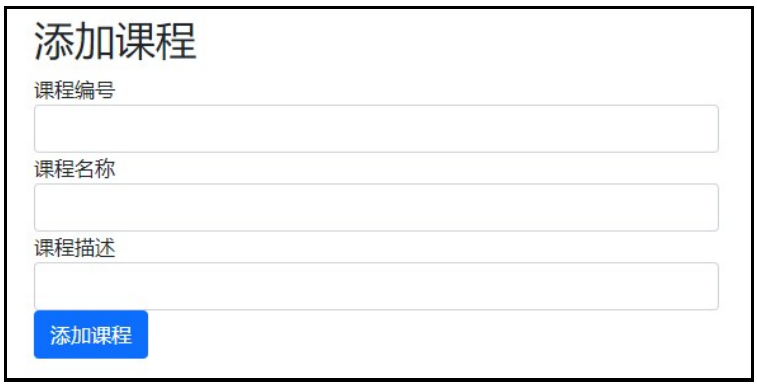

图 2-11　添加课程界面

图 2-12　添加结果界面

【问题描述】

我们正在开发一个在线学习平台，该平台需要一个课程管理功能来管理课程资源。请创建一个名称为 CourseManagement 的 ASP.NET Core MVC 项目，添加课程信息的 Model 类 Course.cs (包含课程编号 CourseNo、课程名称 CourseName、课程描述 Description 属性)和控制器 CourseController，实现如图 2-11 所示的视图界面，在界面中输入课程信息后，单击【添加课程】按钮，返回所添加的课程信息界面，如图 2-12 所示。

【问题分析】

(1) 创建一个 ASP.NET Core MVC 项目，名称为 CourseManagement。

(2) 在项目下的 Models 文件夹中，添加一个 Course.cs 类，并在类中声明 CourseNo、CourseName、Description 属性。

(3) 在项目下的 Controllers 文件夹中，添加一个 CourseController.cs 控制器类，并实现功能：

在界面(见图 2-11)中输入课程编号、课程名称后，单击【添加课程】按钮，返回所添加的课程信息(见图 2-12)。

【参考步骤】

(1) 打开【Visual Studio 2022】界面，单击【创建新项目】按钮，选择【ASP.NET Core Web 应用(模型-视图-控制器)】选项，设置项目名称为 CourseManagement。

(2) 在 CourseManagement 项目下的 Models 文件夹中，添加一个 Course.cs 的类文件，并在类中声明 CourseNo、CourseName、Description 属性，代码如下所示。

```
namespace CourseManagement.Models
{
    public class Course
    {
        /// <summary>
        /// 课程编号
        /// </summary>
        public string CourseNo { get; set; }

        /// <summary>
        /// 课程名称
        /// </summary>
        public string CourseName { get; set; }

        /// <summary>
        /// 课程描述
        /// </summary>
        public string Description { get; set; }
    }
}
```

(3) 在 Controllers 文件夹中，添加一个 CourseController.cs 的控制器类，代码如下所示。

```
namespace CourseManagement.Controllers
{
    public class CourseController : Controller
    {
        [HttpGet]
        public IActionResult Create()
        {
            return View();
        }
        [HttpPost]
        public IActionResult Create(Course course)
        {
            if (!string.IsNullOrEmpty(course.CourseNo)&&!string.IsNullOrEmpty(course.CourseName))
            {
                return Content("课程编号：" + course.CourseNo+"\n 课程名称："+course.Course Name+"\n 课程
                        描述："+course.Description);
            }
            return View();
        }
    }
}
```

(4) 将光标聚焦在 CourseController.cs 中的 Create 方法上，右击，选择【添加视图(D)】，在弹出的对话框中选中【Razor 视图-空】选项，单击【添加】按钮。

(5) 在 Views 文件夹中会自动生成 Course 文件夹，在该文件夹下的 Create.cshtml 文件中，

实现如图 2-11 所示的界面，代码如下所示。

```
@model CourseManagement.Models.Course
<h2>添加课程</h2>
<form asp-action="Create" method="post" class="form-horizontal" role="form">
    <div class="form-group">
        <label asp-for="CourseNo" class="col-md-2 control-label">
            课程编号
        </label>
        <div class="col-md-5">
            <input asp-for="CourseNo" class="form-control" />
            <span asp-validation-for="CourseNo" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="CourseName" class="col-md-2 control-label">
            课程名称
        </label>
        <div class="col-md-5">
            <input asp-for="CourseName" class="form-control" />
            <span asp-validation-for="CourseName" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Description" class="col-md-2 control-label">
            课程描述
        </label>
        <div class="col-md-5">
            <input asp-for="Description" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-5">
            <input type="submit" class="btn btn-primary" value="添加课程" />
        </div>
    </div>
</form>
```