HITE 7.0 软件开发与应用工程师

ASP.NET MVC 实战教程

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

本书按照高等院校计算机课程的基本要求，以案例驱动的形式组织内容，突出计算机课程的实践性特点。本书共包括11个单元：ASP.NET MVC介绍，使用Entity Framework操作数据库，View(视图)，Controller(控制器)，Model(模型)，ASP.NET MVC项目实战，系统登录与注销，会员等级管理，用户信息管理，会员信息管理，会员消费。

本书内容安排合理，结构清晰，所讲内容通俗易懂，实例丰富，突出理论与实践的结合，可作为各类高等院校、培训机构的教材，也可供广大Windows程序设计人员参考。

图书在版编目(CIP)数据

ASP.NET MVC 实战教程 / 武汉厚溥数字科技有限公司编著. —北京：清华大学出版社，2023.2
(HITE 7.0 软件开发与应用工程师)
ISBN 978-7-302-61982-6

I. ①A… II. ①武… III. ①网页制作工具—程序设计—教材 IV. ①TP393.092.2

中国版本图书馆 CIP 数据核字(2022)第 182771 号

责任编辑：刘金喜
封面设计：王 晨
版式设计：孔祥峰
责任校对：成凤进
责任印制：朱雨萌

出版发行：清华大学出版社
网 址：http://www.tup.com.cn，http://www.wqbook.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：**15 **彩 插：**2 **字 数：**309 千字
版 次：2023 年 2 月第 1 版 **印 次：**2023 年 2 月第 1 次印刷
定 价：79.00 元

产品编号：099254-01

编 委 会

主　编：

邓卫红　　寇立红

副主编：

何　爽　　孙　俊　　史晓磊　　韦素云
姜有奇　　杨　晓　　孙　玮

编　委：

闫俊杰　　杨　文　　罗秋菊　　程　胜
杨婷婷　　赖松兆　　李娟娟　　张　乔

主　审：

李　杰　　王　伟

前　言

ASP.NET MVC 是 Windows 系统下面的 Web 开发框架，由 Microsoft 提供。MVC 由 Model、View、Controller 3 个单词的首字母组合而成，是 UI 端分层的三层模式，跟三层架构有着本质区别。.NET MVC 彻底分离了前后端，以及抽象层结构的依赖注入，横切编程模式。用于模型架构的 Model Metadata、用于模型验证的 Validate Provider、用于数据提供的 Value Provider、用于数据绑定的 Model Binder 及用于视图绑定的 View Engine 引擎等，构成了 ASP.NET MVC 架构的模式。

本书是“工信部国家级计算机人才评定体系”中的一本专业教材。“工信部国家级计算机人才评定体系”是由武汉厚溥数字科技有限公司开发，以培养符合企业需求的软件工程师为目标的 IT 职业教育体系。在开发该体系之前，我们对 IT 行业的岗位序列做了充分的调研，包括研究从业人员技术方向、项目经验和职业素养等方面的需求，通过对所面向学生的特点、行业需求的现状及项目实施等方面的详细分析，结合我公司对软件人才培养模式的认知，按照软件专业总体定位要求，进行软件专业产品课程体系设计。该体系集应用软件知识和多领域的实践项目于一体，着重培养学生的熟练度、规范性、集成和项目能力，从而达到预定的培养目标。

本书共包括 11 个单元：ASP.NET MVC 介绍，使用 Entity Framework 操作数据库，View(视图)，Controller(控制器)，Model(模型)，ASP.NET MVC 项目实战，系统登录与注销，会员等级管理，用户信息管理，会员信息管理，会员消费。

我们对本书的编写体系做了精心设计，按照“理论学习—知识总结—上机操作—课后习题”这一思路进行编排。“理论学习”部分描述通过案例要达到的学习目标与涉及的相关知识点，使学习目标更加明确；“知识总结”部分概括案例所涉及的知识点，使知识点能够完整系统地呈现；“上机操作”部分对案例进行了详尽分析，通过完整的步骤帮助读者快速掌握该案例的操作方法；“课后习题”部分帮助读者理解章节的知识点。本书在内容编写方面，力求细致全面；在文字叙述方面，注意言简意赅、重点突出；在案例选取方面，强调案例的针对性和实用性。

本书凝聚了编者多年来的教学经验和成果，可作为各类高等院校、培训机构的教材，也可供广大程序设计人员参考。

本书由武汉厚溥数字科技有限公司编著，由寇立红、李杰、王伟、何爽、罗秋菊、程胜、杨婷婷等多名企业实战项目经理和培训讲师编写。本书编者长期从事项目开发和教学实施，并且对当前高校的教学情况非常熟悉，在编写过程中充分考虑到不同学生的特点和需求，加强了项目实战方面的教学。在本书的编写过程中，得到了武汉厚溥数字科技有限公司各级领导的大力支持，在此对他们表示衷心的感谢。

参与本书编写的人员还有：张家界航空工业职业技术学院邓卫红、杨文，松山职业技术学院孙俊，张家口机械工业学校史晓磊、闫俊杰，包头钢铁职业技术学院韦素云，陕西机电职业技术学院姜有奇，湖北国土资源职业学院杨晓，陕西国际商贸学院孙玮、李娟娟、张乔，闽西职业技术学院赖松兆等。

限于编写时间和编者的水平有限，书中难免存在不足之处，希望广大读者批评指正。

服务邮箱：476371891@qq.com。

编　者

2022 年 9 月

目 录

单元

一

ASP.NET MVC概述

课程目标

- 了解 ASP.NET MVC 开发模式
- 了解 ASP.NET MVC 请求流程
- 创建第一个 ASP.NET MVC 应用程序
- 掌握路由配置方法

简介

在 ASP.NET MVC 开发模式诞生之前，一直流行的开发模式是 ASP.NET WebForms，MVC 成为计算机科学领域重要的构建模式已有多年历史。自从引入以来，MVC 已经在数十种框架中得到应用，在 Java 和 C++语言中，在 macOS 和 Windows 操作系统中以及很多架构内部都用到了 MVC。

MVC 模式是一种流行的 Web 应用架构技术，它被命名为模型-视图-控制器(Model-View-Controller)。在分离应用程序内部的关注点方面，MVC 是一种强大而简洁的方式，尤其适合应用在 Web 应用程序中。

1.1 ASP.NET MVC 介绍

MVC 模式是一种软件架构模式。它把软件系统分为三部分：模型(Model)、视图(View)和控制器(Controller)。MVC 模式最早由 Trygve Reenskaug 在 1974 年提出，是施乐帕罗奥多研究中心(Xerox PARC)在 20 世纪 80 年代为程序语言 Smalltalk 发明的一种软件设计模式。MVC 模式的目的是实现一种动态的程序设计，使后续对程序的修改和扩展简化，并使对程序某一部分的重复利用成为可能。除此之外，此模式通过简化复杂度，使程序结构更加直观。软件系统通过对自身基本部分分离的同时也赋予了各个基本部分应有的功能。

ASP.NET MVC 框架是 MVC 设计模式在 ASP.NET 中的具体应用。2009 年微软发行 ASP.NET MVC 1.0 版，而 ASP.NET MVC 这种开发模式开始流行起来是在 2011 年 ASP.NET MVC 3.0 版本发行之后。这个版本的推出改进了许多非常实用的特性，深受广大开发者喜爱。随后几乎每一年迭代一次新的版本，而今 ASP.NET MVC 已经成为.NET 平台下首选的 Web 应用程序开发方式。

1.1.1 什么是 ASP.NET MVC 开发模式

MVC(Model-View-Controller)模式，强制性地使应用程序输入、处理和输出分开。通常 MVC 应用程序分为 3 个核心部件：模型、视图、控制器。它们各自处理自己的任务。

从宏观上来说，MVC 是框架分层的一种搭建思想，在最原始的项目中，没有什么框架分层之说，所有的项目代码都在一个层里，这样会导致代码冗杂，耦合性强，项目迭代升级困难。MVC 是一种分层思想，将一个项目代码分为几类，分别放到不同的层里，Model 层存储一些数据和业务逻辑；View 层处理页面问题；Controller 层用来接收人机交互指令。

MVC 分层思想和传统的三层(数据库访问层、业务逻辑层、表现层)还是有区别的。

Model(模型)：代表一系列类，用来描述业务逻辑，比如业务模型以及数据访问操作，再比如数据模型。同时也定义了对数据进行处理的业务规则。

View(视图)：代表的是 UI 部分，类似 CSS、jQuery、HTML 等。它的主要职责是展现从 Controller 接收到的数据或模型。

Controller(控制器)：其职责在于处理传入的请求。它接收用户通过视图的输入，然后对用户输入的数据模型进行处理，最终通过视图将结果渲染给用户。通常来讲，控制器在视图和模型之间扮演着桥梁(协调者)的角色。

1.1.2 ASP.NET MVC 的优点

通过 ASP.NET MVC 这种模式开发 Web 应用程序有如下优点：

- 很容易将复杂的应用分成 M、V、C 三个组件模型。通过 Model、View 和 Controller 有效地简化复杂的架构，体现了很好的隔离原则。
- 因为没有使用 Server-based Forms，所以程序的控制更加灵活，页面更加干净。
- 可以控制生成自定义的 URL，对于 SEO(搜索引擎优化)友好的 URL 更不在话下，而 Web Forms 要额外做路由重写来实现伪静态的形式。
- 强类型 View 实现，更安全、更可靠、更高效。
- 让 Web 开发可以专注于某一层，有利于开发中的分工，更利于分工配合，适用于大型架构开发。
- 很多企业已经使用 MVC 作为项目开发框架，招聘时明确要求应聘者熟悉 MVC 开发模式，因为有些项目架构就是 MVC+EF+Web API+…。
- 松耦合、易于扩展和维护。
- 有利于组件的重用。
- ASP.NET MVC 能够更好地支持单元测试(Unit Test)。
- 在团队开发模式下表现更出众。
- MVC 将代码和页面彻底分离，分离到了两个文件中，即视图和控制器。而 Web Forms 的 Codebehind 技术没有完全对代码和前台页面进行分离，耦合度太高。ASPX 和 ASPX.cs 是典型的继承关系。

1.2 创建 ASP.NET MVC 应用程序

1.2.1 创建项目

了解 ASP.NET MVC 开发模式的基本概念之后，可以发现它和之前的 Web Forms 有很大的不同。学习任何新的技术一般都从最简单的创建项目开始，下面就开始创建第一个 ASP.NET MVC 应用程序。

提示：本书中所有的案例和项目演示都以 Visual Studio 2019 和 SQL Server 2014 版本为主，为了更好地学习本书内容，请安装相同版本的开发工具。

(1) 打开 Visual Studio 2019，在“开始使用”中选择“创建新项目”，如图 1-1 所示。

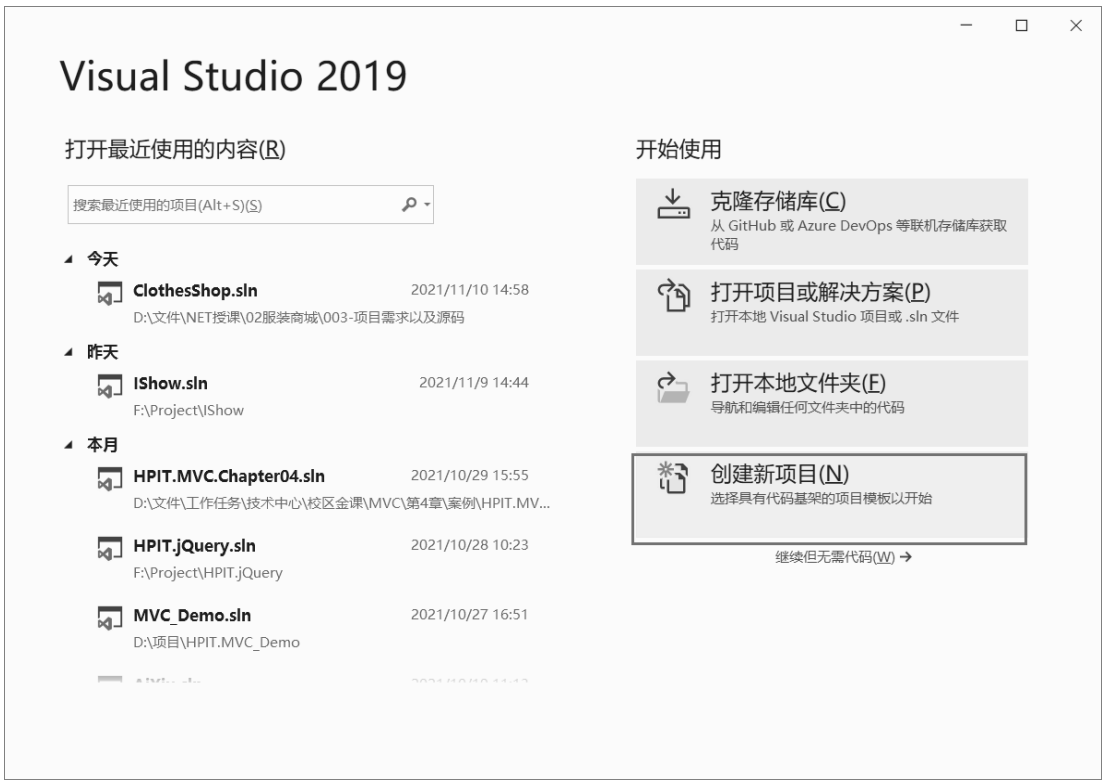

图 1-1　创建新项目

(2) 在弹出的菜单中依次选择项目类型“C#”→“Windows”→“Web”，在下方的菜单中选择“ASP.NET Web 应用程序(.NET Framework)”，然后单击“下一步”按钮，如图 1-2 所示。

图 1-2　创建 ASP.NET Web 应用程序(.NET Framework)

(3) 在弹出的“配置新项目”界面中，依次输入项目名称、位置、解决方案名称，并选择运行时的框架版本，然后单击“创建”按钮，如图 1-3 所示。

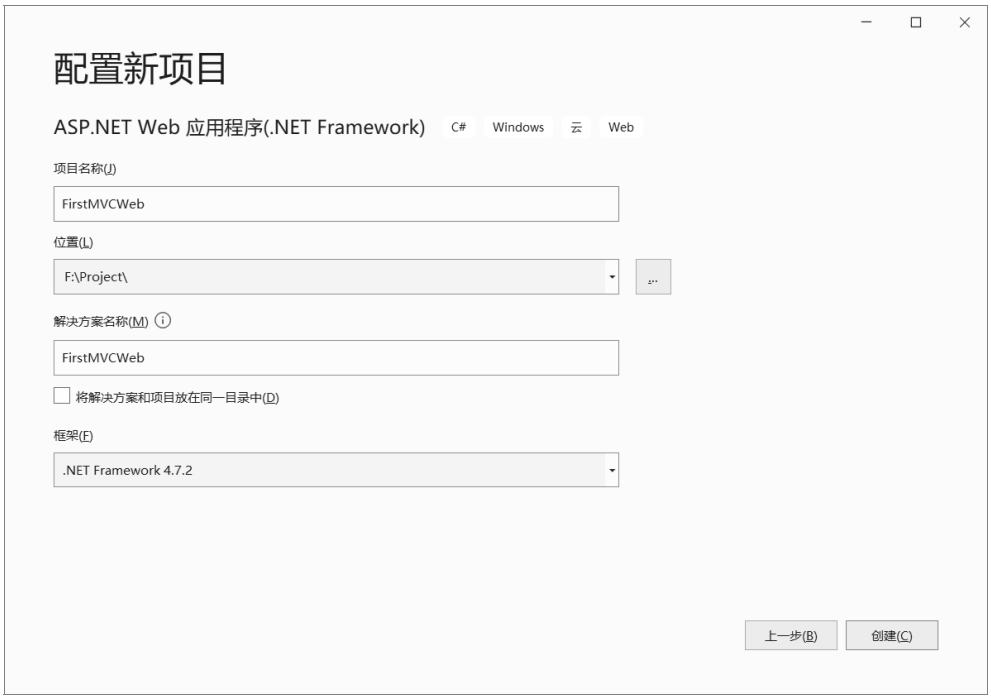

图 1-3　配置新项目

(4) 在弹出的“创建新的 ASP.NET Web 应用程序”界面中，选择 MVC 选项，并取消选中右侧“高级”栏目中的“为 HTTPS 配置”前的复选框(目前学习阶段暂时用不到)，然后单击“创建”按钮，如图 1-4 所示。

图 1-4　创建 MVC 项目

(5) 稍等片刻，Visual Studio 会自动创建好一个 ASP.NET Web 应用程序，如图 1-5 所示。

图 1-5　创建好的 ASP.NET MVC 项目

1.2.2　ASP.NET MVC 项目结构

ASP.NET MVC 项目建立完成后会自动创建一些标准的目录结构，默认的目录结构如图 1-6 所示。

图 1-6　ASP.NET MVC 项目结构

目录结构中部分项目的含义如下。

- App_Data：用来存储数据库文件，如 xml 文件或者应用程序需要的一些其他数据。
- Content：用来存放应用程序中需要用到的一些静态资源文件，如图片和 CSS 样式文件。
- Controllers：用于存放所有控制器类，控制器负责处理请求，并决定哪一个 Action 执行。它充当一个协调者的角色。
- Models：用于存放应用程序的核心类、数据持久化类或者视图模型。如果项目比较大，可以把这些类单独放到一个项目中。
- Scripts：用于存放项目中用到的 JavaScript 文件，默认情况下，系统自动添加了一系列的 js 文件，包含 jQuery 和 jQuery 验证等 js 文件。
- Views：包含许多用于用户界面展示的模板，这些模板都是使用 Razor 视图展示的，子目录对应着控制器相关的视图。
- Global.asax：存放在项目根目录下，代码中包含应用程序第一次启动时的初始化操作，诸如路由注册。
- Web.config：同样存在于项目根目录下，包含 ASP.NET MVC 正常运行所需的配置信息。

至此，已经完成了第一个 ASP.NET MVC 应用程序的创建，接下来可以通过菜单导航去浏览每个页面的内容。

1.2.3 ASP.NET MVC 的请求流程

创建一个新的 ASP.NET MVC 应用程序之后，会发现默认已经生成了一些文件，比如在 Controllers 文件夹中有一个 HomeController，Views 下的 Home 文件夹中有 3 个默认的视图，分别是 About.cshtml、Contact.cshtml 和 Index.cshtml。有了这些默认的文件就可以先试运行，一睹 ASP.NET MVC 的庐山真面目。

(1) 单击运行按钮▶，可以浏览创建好的 ASP.NET MVC Web 应用程序，如图 1-7 所示。

图 1-7　浏览 ASP.NET MVC 项目

(2) 运行后，默认页面如图 1-8 所示。

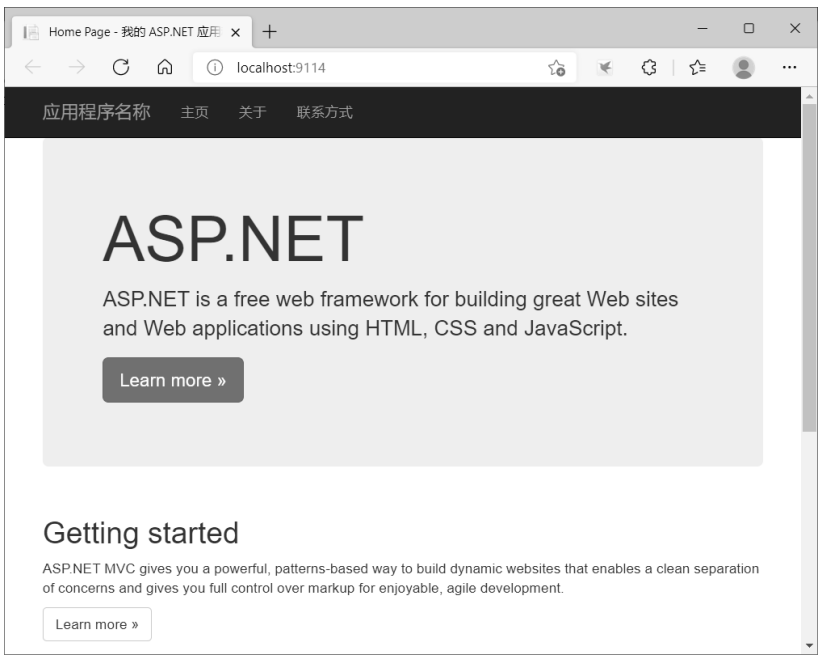

图 1-8　ASP.NET MVC 项目首页

我们可以通过单击菜单导航链接来切换到其他的页面中。

- About 页面(如图 1-9 所示)。

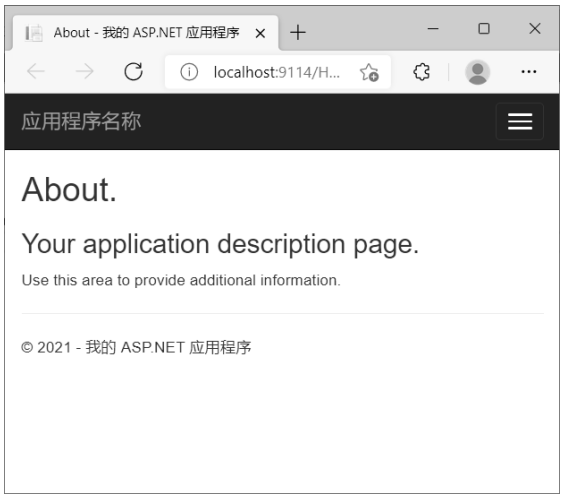

图 1-9　About 页面

- Contact 页面(如图 1-10 所示)。

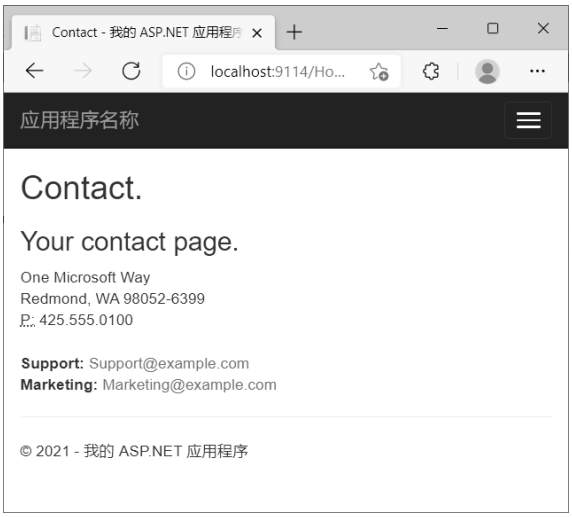

图 1-10　Contact 页面

通过运行项目，我们可以总结一下 URL 地址、控制器和视图之间的关系。在 ASP.NET MVC 中控制器充当的是处理请求的职责，URL 地址 http://localhost: 9114/Home/Index 请求的即为 HomeController 中的 Index 方法。在 HomeController 控制器中，默认有 3 个方法：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.Mvc;

namespace FirstMVCWeb.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

其返回值类型为 ActionResult，在每个方法中默认返回 return View()，在 ASP.NET MVC 中这类方法被称为“Action”，当前的 Action 都返回一个“视图”方法。ASP.NET MVC 最终会去加载和它名称一致的视图，最终便看到了视图中所呈现的内容。

通常情况下，ASP.NET MVC 开发模式的请求模型如图 1-11 所示。

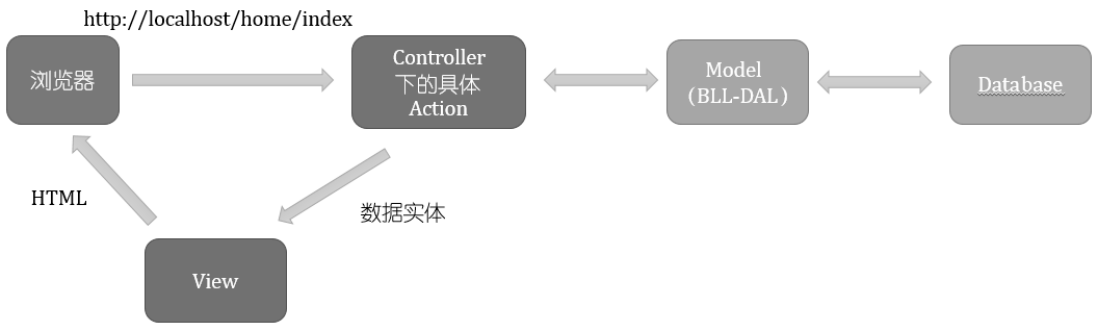

图 1-11　ASP.NET MVC 请求模型

(1) 用户打开浏览器，在地址栏输入某个网址 URL 并按 Enter 键，浏览器便开始向该 URL 指向的服务器发送 HTTP 请求(一般是 GET 方式)。

(2) 服务器端的网站服务系统(IIS)接收到该请求，先检查自己是否认识该类请求，如果认识就直接处理并发回响应，否则就将该类型的请求发给对应的 HTTP 处理程序(在此是 ASP.NET MVC)。

(3) MVC 路由系统收到请求后，根据 HTTP 请求的 URL，把请求定向到对应的控制器。

(4) 如果控制器是 MVC 内置的标准 Controller，则启动 Action 机制；否则，根据自定义的控制器逻辑，直接向浏览器发回响应。

(5) MVC 路由把 HTTP 请求定向到具体的 Controller/Action，如果 Action 没有使用视图引擎，则根据自定义逻辑发回响应；否则返回 ActionResult 给视图引擎(WebForms 或 Razor)，由视图引擎渲染呈现 HTML，并返回给浏览器。

1.2.4　添加新的控制器和视图

明白了 ASP.NET MVC 的基本请求流程之后，接下来就可以创建自己的控制器和视图，并能够最终请求到自己的页面中。

(1) 右击文件夹 Controllers，选择“添加”→“控制器”→“MVC5 控制器”命令，打开如图 1-12 所示对话框。选择“MVC 5 控制器-空”选项，然后单击“添加”按钮。

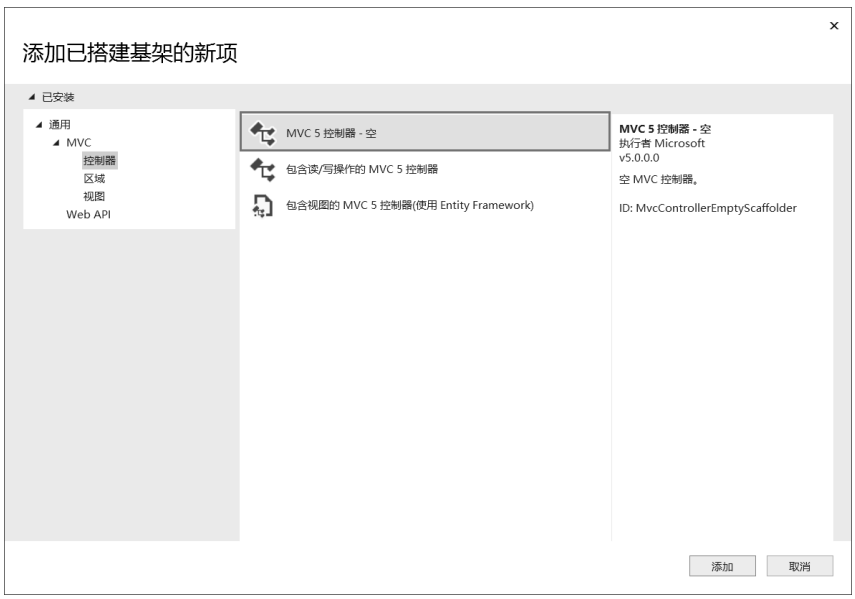

图 1-12　添加控制器

(2) 在打开的对话框中填写新的控制器名称，如“DemoController”，然后单击“添加”按钮，如图 1-13 所示。

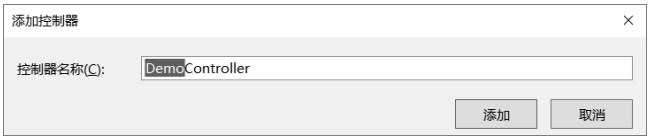

图 1-13　填写控制器名称

添加完成后，会在 DemoController 中默认添加一个 Index 方法：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace FirstMVCWeb.Controllers
{
    public class DemoController : Controller
    {
        // GET: Demo
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

同时，在 Views 文件夹中自动创建了一个 Demo 文件，用来存放和 DemoController 相关的视图文件，如图 1-14 所示。

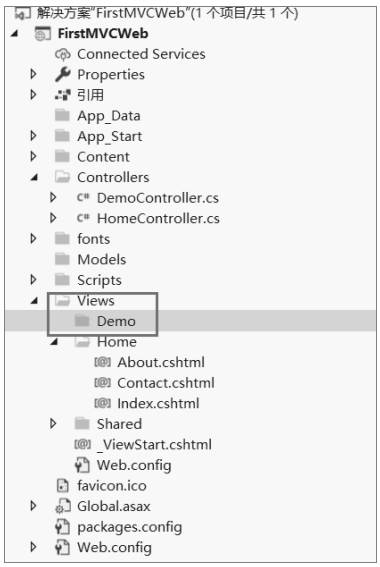

图 1-14　Demo 文件夹

(3) 添加视图。添加视图的方法有两种，一种是在 Demo 文件夹上单击右键，然后选择“添加”→“视图”菜单，如图 1-15 所示。

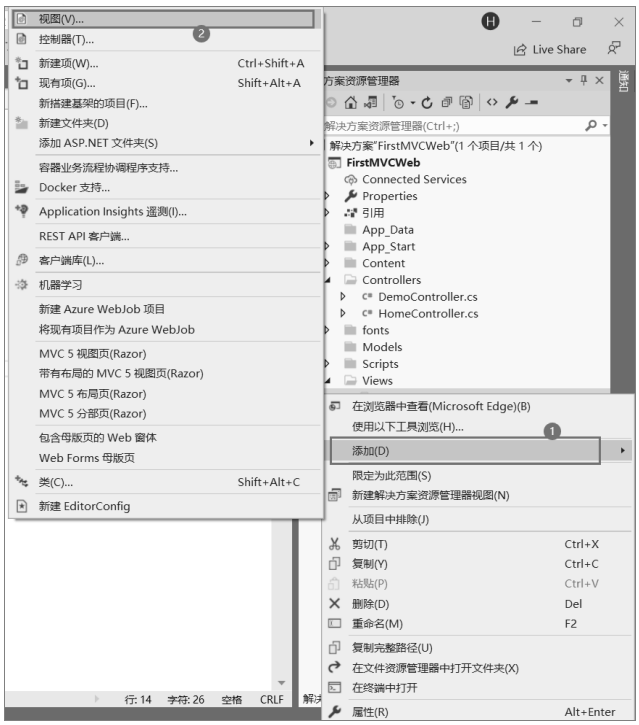

图 1-15　在文件夹中添加视图

另一种方法是在当前 Action 方法中单击右键，选择“添加视图”命令，如图 1-16 所示。

图 1-16　在控制器中添加视图

(4) 在弹出的对话框中选择“MVC5 视图”选项，然后单击“添加”按钮，如图 1-17 所示。

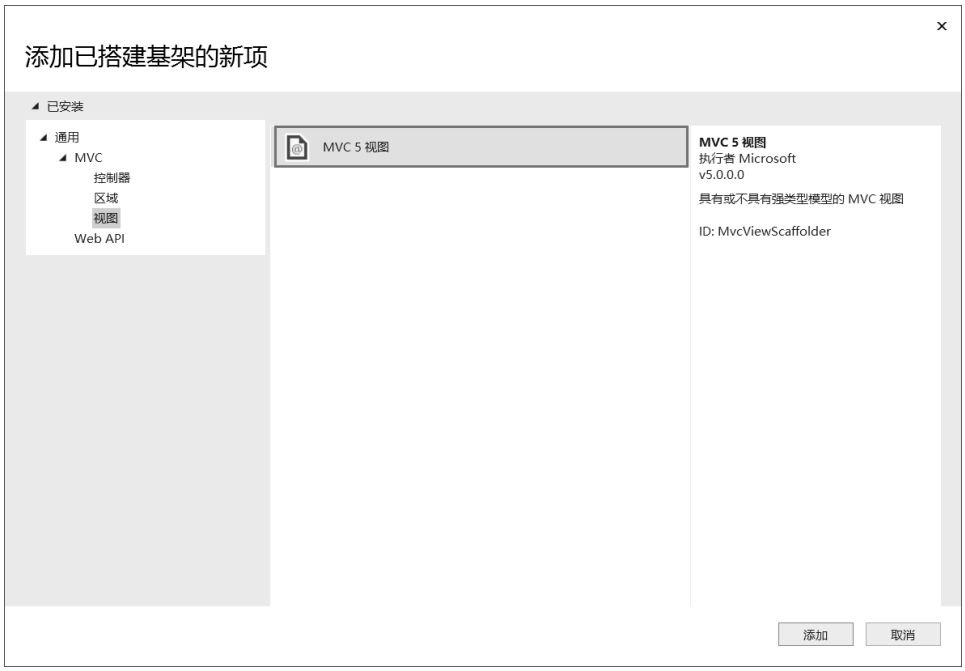

图 1-17　选择视图

(5) 在“添加视图”对话框中输入视图的名称，通常情况下视图的名称默认和当前 Action 的名称一致，我们用默认名称即可，然后单击“添加”按钮，如图 1-18 所示。

图 1-18　配置视图

(6) 在添加好的视图中，可以编写自己想要的页面效果。

```
@{
    ViewBag.Title = "Index";
}
```

```
<h2>Index</h2>

<h3>这是我的新视图</h3>
```

(7) 编写完成后，可以运行查看当前的页面，如图 1-19 所示。

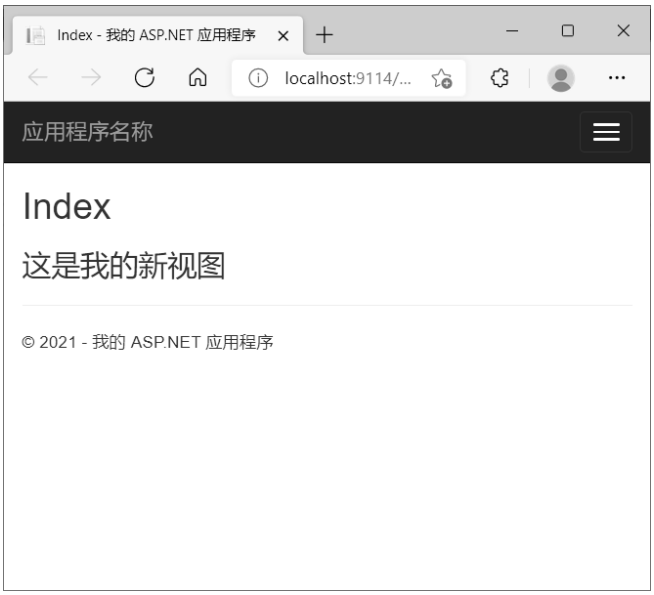

图 1-19　新视图页面

小　结

模型(Model)：指对业务流程/状态的处理以及业务规则的规定。业务流程的处理过程对其他层来说是不透明的，模型接受视图数据的请求，并返回最终的处理结果。业务模型的设计可以说是 MVC 的核心。

视图(View)：代表用户交互界面，对 Web 来说是 HTML 界面。

控制器(Controller)：可以理解为一个分发器，它决定选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求。控制层并不做任何的数据处理，它负责接受用户的请求，将模型与视图匹配在一起，共同完成用户请求。

1.3 路由

对于 MVC 项目，可以发现 URL 的地址并不像以前的 WebForms 那样指向的是一个实实在在的 aspx 页面文件，那么它是如何寻找所请求的具体行为方法的呢？下面我们就来一探究竟。

1.3.1 映射 URL 到 Action

在 Web 应用中，用户都会通过 URL(统一资源定位符，俗称网址)来发送对页面的请求。打开浏览器，输入将要访问网站的网址，然后等待浏览器加载我们期待的页面。

- 传统的 WebForm 开发，URL 映射到的是一个具体的处理程序、磁盘上的物理文件，如一个 aspx 文件。
- MVC 中多数情况下是将 URL 映射到 Controller 和 Controller 下的 Action。

ASP.NET MVC 引入了路由，以消除将每个 URL 映射到物理文件的需求。路由使我们能够定义映射到请求处理程序的 URL 模式。这个请求处理程序可以是一个文件或类。

路由定义了 URL 模式和处理程序信息。存储在 RouteTable 中的应用程序的所有已配置路由将由路由引擎为传入请求确定适当的处理程序类或文件。

假如用户请求的 URL 是 http://localhost:9114/Home/Index，则路由的执行过程如图 1-20 所示。

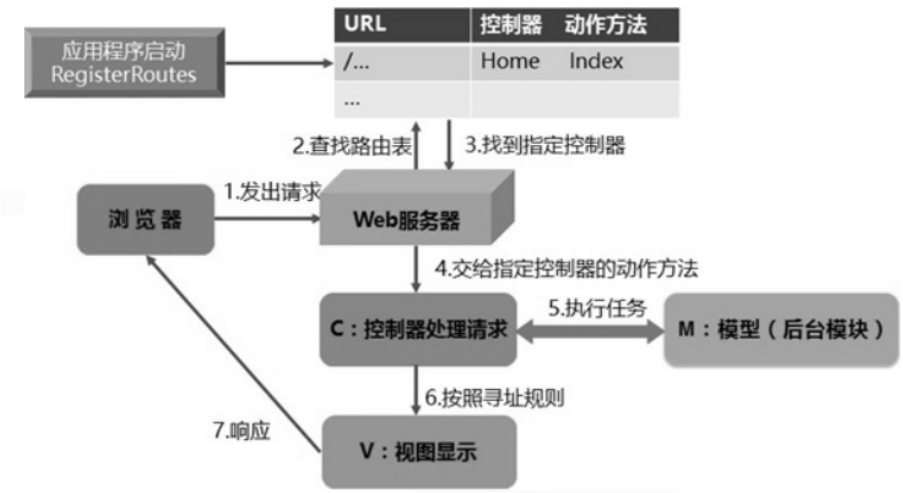

图 1-20　路由执行过程

1.3.2 路由配置

在 ASP.NET MVC 项目中，打开 Global.asax 文件，其中 Application_Start 方法在应用程序启动时执行，RouteConfig.RegisterRoutes(RouteTable.Routes)用来注册路由信息。

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
```

```
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
```

RouteConfig 为路由的详细配置，位于 App_Start 文件夹下的 RouteConfig.cs 文件中。

```
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home"，action = "Index"，id = UrlParameter.Optional }
            );
        }
    }
```

- name 为路由名称，可以注册多个路由，但是必须保证 name 名称是唯一的。
- default 为默认路由，当 URL 中没有出现完整的请求内容时，会按照默认路由定义的规则来进行请求。

上面的 URL 中的参数值是"{controller}/{action}/{id}"，称之为 URL 模式。该模式是一种字符串，包括一些固定的“字符字面量”和“占位符”，占位符用大括号{}表示。URL 模式规定了 URL 路径的定义规则。

- 占位符：可以是一个字符串或字符，比如“x”“id”“year”等。
- 字符字面量：可能是一个比较固定的字符，比较常见的是斜杠“/”；也可以是字符串。

在匹配方面要求必须严格匹配，即实际请求的 URL 中的字符串和路由模式中的字面量字符串必须完全一致。

- 大小写：URL 模式匹配不区分大小写。

定义多个路由的方法，可参考图 1-21 所示的代码。

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
    );
    routes.MapRoute(
        name: "Test1",
        url: "{first}/{second}/{third}",
        defaults: new { controller = "Work", action = "Index", id = UrlParameter.Optional }
    );
}
```

图 1-21　定义路由

路由的匹配原则：如果一个 URL 能够在多个路由中匹配，则默认使用第一个匹配的路由。

UrlParameter.Optional 参数的作用是什么？该参数可以作为路由参数的默认值，当需要让/Home/Index 或/Home 能正常匹配，但又不希望赋一个无意义的值时，可以使用该参数。

小　结

路由匹配总结：

(1) 关于{controller}/{action}

必不可少：在一个实际的 MVC 系统中，{controller}和{action}必不可少。如果缺少，就会因找不到路径而出错。

约定规则：这个占位符是 MVC 中约定的，并且会被解析成控制器和对应的方法。

位置灵活：这两个约定的占位符可以在任意位置。

(2) 其他占位符

仅仅占位：其他占位符只起占位的作用，比如{aa}/{bb}/{cc}不能把 aa 解析成控制器、bb 解析成动作方法。默认要求：一个路由中，如果没有规定{controller}和{action}，或者只是规定其中之一，则没有规定的部分都将使用默认值。

(3) 匹配顺序

优先使用：多个路由匹配一个 URL，则会使用优先匹配的路由。

尽量避免：定义多个路由时，尽量避免出现多匹配。

单元小结

- MVC 应用程序具有 3 个核心部件：模型、视图、控制器。
- 模型接收视图数据的请求，并返回最终的处理结果。
- 视图代表用户交互界面，对 Web 来说是 HTML 界面。
- 控制器负责接受用户的请求，将模型与视图匹配在一起，共同完成用户请求。
- 在 MVC 中 cshtml 文件和 cs 文件是分离的，一个控制器对应一组页面。
- MVC 通过路由映射查找 Controller 下的某个方法，约定大于配置。

单元自测

选择题

1. MVC 中视图文件的扩展名是(　　)。

   A. aspx　　B. cshtml

   C. html　　D. asp

2. 以下关于 MVC 的说法中，不正确的是(　　)。

   A. MVC 中显示页面和逻辑分离

   B. MVC 通过路由映射查找 Controller 下的某个方法

   C. MVC 中约定大于配置

   D. MVC 中 Action 的返回结果必须是 View

3. 关于路由，以下描述中不正确的是(　　)。

   A. 路由配置信息在 RouteConfig 文件中

   B. 一个 MVC 项目中只能有一个路由规则

   C. 可以在一个项目中添加多个路由规则

   D. 我们所请求的 URL 地址不区分大小写

4. MVC 中的路由配置在(　　)文件中。

   A. Web.config　　B. Global.asax

   C. FilterConfig.cs　　D. RouteConfig.cs

5. MVC 中控制器必须以(　　)结尾。

A. Service　　B. Controller

C. Control　　D. 可以以任意名称结尾

问答题

1. 简述 ASP.NET MVC 的请求流程。

2. 简述 ASP.NET MVC 和 ASP.NET WebForms 的区别。