

.NET 开发经典名著

ASP.NET Core 3

高级编程(第 8 版)

(上册)

[英] 亚当·弗里曼(Adam Freeman) 著

杜静芬 程凤娟

译

清华大学出版社

北 京

北京市版权局著作权合同登记号 图字：01-2021-3335

EISBN: 978-1-4842-5439-4

First published in English under the title

Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages,
Eighth Edition

By Adam Freeman

Copyright © Adam Freeman, 2020

This edition has been translated and published under licence from Apress Media, LLC, part of Springer
Nature.

本书中文简体字版由 Apress 出版公司授权清华大学出版社出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。举报：010-62782989, beiqinquan@tup.tsinghua.edu.cn。

图书在版编目(CIP)数据

ASP.NET Core 3高级编程：第8版 / (英)亚当·弗里曼 (Adam Freeman) 著；杜静芬，程凤娟译. — 北京：
清华大学出版社，2021.6

(.NET 开发经典名著)

书名原文：Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor
Pages, Eighth Edition

ISBN 978-7-302-58271-7

I. ①A… II. ①亚… ②杜… ③程… III. 网页制作工具—程序设计 IV. ①TP393.092.2

中国版本图书馆 CIP 数据核字(2021)第 101201 号

责任编辑：王 军 韩宏志

装帧设计：孔祥峰

责任校对：成凤进

责任印制：宋 林

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：010-62770175 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：三河市国英印务有限公司

经 销：全国新华书店

开 本：170mm×240mm 印 张：73 字 数：2011 千字

版 次：2021 年 7 月第 1 版 印 次：2021 年 7 月第 1 次印刷

定 价：268.00 元(全二册)

产品编号：089796-01

译者序

ASP.NET Core 是一款由微软创建的，用于构建 Web 应用、API、微服务的 Web 框架。它使用常见的模式，如 MVC(Model-View-Controller)、依赖注入和一个由中间件构成的请求处理管道。ASP.NET Core 是重新设计的 ASP.NET 4.x，更改了体系结构，形成了更精简的模块化框架。

ASP.NET Core 运行在微软的 .NET 运行时库上，有几种语言(C#、Visual Basic 和 F#)可用来编写 ASP.NET Core 程序。C#是最常见的选择，可在 Windows、macOS 和 Linux 上构建并运行 ASP.NET Core 应用。

为什么要用 ASP.NET Core 开发应用程序？现存的 Web 框架选项已经很多了，如 Node/Express、Spring、Ruby on Rails、Django、Laravel 等，数不胜数。ASP.NET Core 具有如下优点。

- 快速：因为 .NET Core 是编译运行的，执行速度远高于解释执行的语言，如 JavaScript 或 Ruby，ASP.NET Core 也已为多线程和异步任务做了专门优化。与使用 Node.js 写的代码相比，执行速度高出 5~10 倍是很正常的。
- 生态：在 NuGet(.NET 的包管理系统，如 NPM、RubyGems 或 Maven)中有成千上万的软件包。有现成的包可用来完成 JSON 反序列化、数据库连接、PDF 生成等。
- 安全：通过 ASP.NET Core，开发者可轻松配置和管理应用的安全性。ASP.NET Core 的功能包括管理身份验证、授权、数据保护、HTTPS 强制、应用机密、请求防伪保护及 CORS 管理。通过这些安全功能，可生成安全可靠的 ASP.NET Core 应用。
- 跨平台：能在 Windows、macOS 和 Linux 上开发和运行。
- 开源：出现问题时，可阅读其源代码，来获取解决问题的方法。

本书深入浅出地介绍 ASP.NET Core 基础及实战等方面的知识，共分 4 个部分。第 I 部分介绍 ASP.NET Core。除了设置开发环境和创建第一个应用程序外，还介绍对 ASP.NET Core 开发最重要的 C#特性和如何使用 ASP.NET Core 开发工具。第 II 部分描述 ASP.NET Core 平台的主要特性，解释如何处理 HTTP 请求，如何创建和使用中间件组件，如何创建路由，如何定义和使用服务，以及如何与 Entity Framework Core 一起工作。第 III 部分解释如何创建不同类型的应用程序，包括 RESTful Web 服务以及使用控制器和 Razor Pages 的 HTML 应用程序。第 IV 部分解释如何使用 Blazor 服务器创建应用程序，如何使用实验性的 Blazor WebAssembly，以及如何使用 ASP.NET Core 验证用户身份和授予访问权限。

本书对于任何一名 C#开发者来说都是一本宝贵的指导书。书中所涉及的对于代码设计的实用建议是非常宝贵的。高效率的 .NET 开发者需要对他所选择的语言有很深的理解。本书作者以令人惊叹的能力，把极其复杂的问题拆解为可消化的、易理解的一个个小问题，进行合理探讨，以一定的洞察力将知识传授给读者，并教给读者如何书写实践性强、干净简单且更容易理解的代码。无论是 C#新手还是资深开发者，都能通过阅读本书而有所收获。

本书文字简洁明快、流畅，既适合初学者及具有.NET 基础的开发者阅读，还可作为大中专院校计算机、通信、电子信息、自动化等相关专业的教材；也可供软件项目管理人员、开发团队成员学习参考。

这里要感谢清华大学出版社的编辑，他们为本书的翻译投入了巨大热情并付出了很多心血。没有他们的帮助和鼓励，本书不可能顺利付梓。

对于这本经典之作，译者本着“诚惶诚恐”的态度，在翻译过程中力求“信、达、雅”，但鉴于译者水平有限，失误在所难免，如果你有任何意见和建议，欢迎指正。

作者简介

Adam Freeman 是一位经验丰富的 IT 专业人士，曾在多家公司担任高级职位，后担任一家全球银行的首席技术官和首席运营官。现在退休了，他把时间花在写作和长跑上。

技术审校者简介

Fabio Claudio Ferracchiati 是一名使用微软技术的高级顾问和高级分析师/开发人员。他为 BluArancio (www.bluarancio.com) 工作。他是一名 .NET 解决方案开发人员 and 应用程序开发人员，也是一名多产的作者和技术评论员。在过去十年里，他为意大利和国际杂志撰写文章，并与他人合著了十多本关于各种计算机主题的书籍。

目 录

第 I 部分 介绍 ASP.NET Core

第 1 章	ASP.NET Core 上下文	3
1.1	了解 ASP.NET Core	3
1.1.1	理解应用程序框架	3
1.1.2	理解实用程序框架	5
1.1.3	了解 ASP.NET Core 平台	5
1.2	理解本书	5
1.2.1	需要什么软件来完成示例?	6
1.2.2	需要什么平台来完成示例?	6
1.2.3	源代码下载	6
1.2.4	如果在执行这些示例时遇到 问题, 怎么办?	6
1.2.5	如果发现书中有错误, 怎么办?	6
1.2.6	本书包含的内容	6
1.2.7	本书未包含的内容	7
1.2.8	如何联系作者?	7
1.2.9	如果你真的喜欢本书?	8
1.2.10	如果本书让人生气, 想要抱怨 该怎么办?	8
1.3	小结	8
第 2 章	入门	9
2.1	选择代码编辑器	9
2.1.1	安装 Visual Studio	10
2.1.2	安装 Visual Studio Code	12
2.2	创建 ASP.NET Core 项目	16
2.2.1	用 Visual Studio 打开项目	16
2.2.2	用 Visual Studio Code 打开项目	17
2.3	运行 ASP.NET Core 应用程序	18
2.3.1	理解端点	20
2.3.2	了解路由	21
2.3.3	理解 HTML 渲染	22
2.3.4	内容综述	26
2.4	小结	26

第 3 章	第一个 ASP.NET Core 应用程序	27
3.1	设置场景	27
3.2	创建项目	27
3.2.1	添加数据模型	29
3.2.2	创建第二个操作和视图	29
3.2.3	连接操作方法	31
3.2.4	构建表单	32
3.2.5	接收表单数据	34
3.2.6	添加 Thanks 视图	36
3.2.7	显示响应	37
3.2.8	添加验证	39
3.2.9	内容的样式化	44
3.3	小结	49
第 4 章	使用开发工具	51
4.1	创建 ASP.NET Core 项目	51
4.1.1	使用命令行创建项目	52
4.1.2	使用 Visual Studio 创建项目	54
4.2	向项目中添加代码和内容	57
4.3	构建和运行项目	59
4.3.1	使用命令行构建和运行项目	60
4.3.2	使用 Visual Studio Code 构建和运行项目	60
4.3.3	使用 Visual Studio 构建和 运行项目	60
4.4	管理包	61
4.4.1	管理 NuGet 包	61
4.4.2	管理工具包	62
4.4.3	管理客户端包	63
4.4.4	使用 Visual Studio 管理包	64
4.4.5	使用 Visual Studio 管理 客户端包	65
4.5	调试项目	65
4.6	小结	66

第 5 章 C#的基本特点	67
5.1 准备工作	67
5.1.1 打开项目	68
5.1.2 启用 MVC 框架	68
5.1.3 创建应用程序组件	69
5.1.4 选择 HTTP 端口	70
5.1.5 运行示例应用程序	71
5.2 使用 null 条件运算符	71
5.2.1 链接 null 条件运算符	72
5.2.2 结合条件运算符和合并运算符	74
5.3 使用自动实现的属性	76
5.3.1 使用自动实现的属性初始化器	77
5.3.2 创建自动实现的只读属性	77
5.4 使用字符串插值	79
5.5 使用对象和集合初始化器	80
5.6 模式匹配	82
5.7 使用扩展方法	84
5.7.1 将扩展方法应用到接口	86
5.7.2 创建过滤扩展方法	88
5.8 使用 lambda 表达式	89
5.8.1 定义函数	91
5.8.2 使用 lambda 表达式方法和属性	94
5.9 使用类型推断和匿名类型	96
5.10 在接口中使用默认实现	98
5.11 使用异步方法	101
5.11.1 直接处理任务	101
5.11.2 应用 async 和 await 关键字	102
5.11.3 使用异步枚举	104
5.12 获取名称	107
5.13 小结	109
第 6 章 测试 ASP.NET Core 应用程序	111
6.1 准备工作	112
6.1.1 打开项目	112
6.1.2 选择 HTTP 端口	112
6.1.3 启用 MVC 框架	113
6.1.4 创建应用程序组件	113
6.1.5 运行示例应用程序	115
6.2 创建单元测试项目	115
6.3 编写和运行单元测试	116

6.3.1 使用 Visual Studio Test Explorer 运行测试	118
6.3.2 使用 Visual Studio Code 运行测试	119
6.3.3 从命令行运行测试	119
6.3.4 纠正单元测试	120
6.3.5 为单元测试隔离组件	121
6.3.6 使用模拟包	126
6.3.7 创建模拟对象	126
6.4 小结	128

第 7 章 SportsStore: 一个真正的应用程序	129
7.1 创建项目	130
7.1.1 创建单元测试项目	130
7.1.2 创建应用程序项目文件夹	130
7.1.3 打开项目	131
7.1.4 准备应用程序服务和请求管道	132
7.1.5 配置 Razor 视图引擎	133
7.1.6 创建控制器和视图	134
7.1.7 启动数据模型	135
7.1.8 检查和运行应用程序	135
7.2 向应用程序添加数据	136
7.2.1 安装 Entity Framework Core 包	136
7.2.2 定义连接字符串	136
7.2.3 创建数据库上下文类	137
7.2.4 配置 Entity Framework Core	138
7.2.5 创建存储库	139
7.2.6 创建数据库迁移	141
7.2.7 创建种子数据	142
7.3 显示产品列表	145
7.3.1 准备控制器	145
7.3.2 更新视图	147
7.3.3 运行应用程序	148
7.4 添加分页	148
7.4.1 显示页面的链接	150
7.4.2 改善 URL	158
7.5 内容的样式化	160
7.5.1 安装 Bootstrap 包	161
7.5.2 应用 Bootstrap 风格	161
7.5.3 创建部分视图	164
7.6 小结	165

第 8 章 SportsStore: 导航和购物车	167
8.1 添加导航控件	167
8.1.1 筛选产品列表	167
8.1.2 优化 URL 方案	172
8.1.3 构建一个类别导航菜单	176
8.1.4 更正页数	183
8.2 构建购物车	186
8.2.1 配置 Razor Pages	186
8.2.2 创建 Razor Pages	189
8.2.3 创建 Add To Cart 按钮	189
8.2.4 启用会话	191
8.2.5 实现购物车功能	193
8.3 小结	203
第 9 章 SportsStore: 完成购物车	205
9.1 使用服务改进 Cart 模型	205
9.1.1 创建支持存储的 Cart 类	205
9.1.2 注册服务	207
9.1.3 简化购物车 Razor Pages	209
9.2 完成购物车的功能	211
9.2.1 从购物车中删除商品	211
9.2.2 添加购物车摘要小部件	214
9.3 提交订单	217
9.3.1 创建模型类	217
9.3.2 添加付款过程	218
9.3.3 创建控制器和视图	218
9.3.4 实现订单处理	221
9.3.5 完成订单控制器	224
9.3.6 显示验证错误	227
9.3.7 显示摘要页面	229
9.4 小结	230
第 10 章 SportsStore: 管理	231
10.1 准备 Blazor 服务器	231
10.1.1 创建导入文件	233
10.1.2 创建 Startup Razor Pages	233
10.1.3 创建路由和布局组件	234
10.1.4 创建 Razor 组件	235
10.1.5 检查 Blazor 的设置	235
10.2 管理订单	236
10.2.1 增强模型	236
10.2.2 向管理员显示订单	238

10.3 添加目录管理	241
10.3.1 扩展存储库	241
10.3.2 将验证属性应用到数据模型	242
10.3.3 创建列表组件	243
10.3.4 创建细节组件	245
10.3.5 创建编辑器组件	246
10.3.6 删除产品	249
10.4 小结	251
第 11 章 SportsStore: 安全与部署	253
11.1 确保管理功能的安全	253
11.1.1 创建身份数据库	253
11.1.2 添加常规的管理特性	259
11.1.3 应用基本授权策略	260
11.1.4 创建账户控制器和视图	262
11.1.5 测试安全策略	266
11.2 准备进行部署	266
11.2.1 配置错误的处理	266
11.2.2 创建生产配置设置	268
11.2.3 创建 Docker 映像	268
11.2.4 运行容器化应用程序	271
11.3 小结	272

第 II 部分 ASP.NET Core 平台

第 12 章 了解 ASP.NET Core 平台	275
12.1 准备工作	276
12.2 了解 ASP.NET Core 平台	277
12.2.1 理解中间件和请求管道	277
12.2.2 了解服务	277
12.3 了解 ASP.NET Core 项目	278
12.3.1 理解入口点	279
12.3.2 理解 Startup 类	280
12.3.3 理解项目文件	281
12.4 创建自定义中间件	283
12.4.1 使用类定义中间件	286
12.4.2 理解返回管道路径	289
12.4.3 请求管道短路	290
12.4.4 创建管道分支	292
12.4.5 创建终端中间件	294
12.5 配置中间件	297
12.6 小结	301

第 13 章	使用 URL 路由	303
13.1	准备工作	304
13.1.1	理解 URL 路由	307
13.1.2	添加路由中间件、定义端点	307
13.1.3	理解 URL 模式	310
13.1.4	在 URL 模式中使用段变量	311
13.1.5	从路由中生成 URL	315
13.2	管理 URL 的匹配	319
13.2.1	从一个 URL 段匹配多个值	319
13.2.2	为段变量使用默认值	320
13.2.3	在 URL 模式中使用可选段	321
13.2.4	使用 catchall 段变量	323
13.2.5	约束段的匹配	324
13.2.6	定义回退路由	327
13.3	高级路由功能	328
13.3.1	创建自定义约束	328
13.3.2	避免模棱两可的路由异常	330
13.3.3	访问中间件组件中的端点	332
13.4	小结	334
第 14 章	使用依赖注入	335
14.1	为本章做准备	336
14.1.1	创建中间件组件和端点	337
14.1.2	配置请求管道	338
14.2	理解服务位置和紧密耦合	339
14.2.1	理解服务位置问题	340
14.2.2	理解紧密耦合组件的问题	342
14.3	使用依赖注入	344
14.3.1	在中间件类中使用服务	346
14.3.2	在端点中使用服务	347
14.4	使用服务生命周期	352
14.4.1	创建临时服务	353
14.4.2	避免临时服务重用陷阱	354
14.4.3	使用有作用域的服务	357
14.5	其他依赖注入特性	363
14.5.1	创建依赖关系链	363
14.5.2	访问 ConfigureServices 方法中的服务	365
14.5.3	使用服务工厂函数	366
14.5.4	创建具有多个实现的服务	367
14.5.5	在服务中使用未绑定类型	370

14.6	小结	372
第 15 章	使用平台特性(第 1 部分)	373
15.1	准备工作	374
15.2	使用配置服务	375
15.2.1	理解特定于环境的配置文件	376
15.2.2	访问配置设置	377
15.2.3	在服务中使用配置数据	378
15.2.4	理解启动设置文件	381
15.2.5	确定启动类中的环境	387
15.2.6	存储用户的秘密	388
15.3	使用日志服务	392
15.3.1	生成日志消息	392
15.3.2	配置最小日志级别	395
15.4	使用静态内容和客户端包	397
15.4.1	添加静态内容中间件	397
15.4.2	使用客户端包	401
15.5	小结	404
第 16 章	使用平台特性(第 2 部分)	405
16.1	准备工作	405
16.2	使用 cookie	406
16.2.1	启用 cookie consent 检查	409
16.2.2	管理 cookie consent	411
16.3	使用会话	413
16.3.1	配置会话服务和中间件	413
16.3.2	使用会话数据	415
16.4	使用 HTTPS 连接	417
16.4.1	启用 HTTP 连接	417
16.4.2	检测 HTTPS 请求	419
16.4.3	执行 HTTPS 请求	420
16.4.4	启用 HTTP 严格传输安全性	422
16.5	处理异常和错误	425
16.5.1	返回 HTML 错误响应	427
16.5.2	富集状态码响应	429
16.6	使用 Host 头过滤请求	431
16.7	小结	433
第 17 章	处理数据	435
17.1	准备工作	436
17.2	缓存数据	438

17.2.1 缓存数据值	440	19.5.3 使用操作的结果	493
17.2.2 使用共享和持久的数据 缓存	443	19.5.4 验证数据	499
17.3 缓存响应	447	19.5.5 应用 API 控制器属性	501
17.4 使用 Entity Framework Core	449	19.5.6 忽略 Null 属性	502
17.4.1 安装 Entity Framework Core	450	19.6 小结	505
17.4.2 创建数据模型	451	第 20 章 高级 Web 服务特性	507
17.4.3 配置数据库服务	452	20.1 准备工作	507
17.4.4 创建和应用数据库迁移	453	20.1.1 删除数据库	508
17.4.5 播种数据库	454	20.1.2 运行示例应用程序	508
17.4.6 在端点中使用数据	457	20.2 处理相关数据	509
17.5 小结	460	20.3 支持 HTTP Patch 方法	512
第 III 部分 ASP.NET Core 应用程序		20.3.1 理解 JSON Patch	512
第 18 章 创建示例项目	463	20.3.2 安装和配置 JSON Patch 包	513
18.1 创建项目	463	20.3.3 定义操作方法	514
18.2 添加数据模型	464	20.4 理解内容的格式化	515
18.2.1 向项目中添加 NuGet 包	464	20.4.1 理解默认的内容策略	515
18.2.2 创建数据模型	464	20.4.2 理解内容协商	517
18.2.3 准备种子数据	466	20.4.3 指定操作结果格式	521
18.2.4 配置 Entity Framework Core 服务和中间件	467	20.4.4 在 URL 中请求格式	522
18.2.5 创建和应用迁移	469	20.4.5 限制操作方法接收的格式	524
18.3 添加 CSS 框架	469	20.5 记录和探索 Web 服务	525
18.4 配置请求管道	470	20.5.1 解决操作冲突	526
18.5 运行示例应用程序	472	20.5.2 安装和配置 Swashbuckle 包	527
18.6 小结	472	20.5.3 微调 API 描述	529
第 19 章 创建 RESTful Web 服务	473	20.6 小结	533
19.1 准备工作	474	第 21 章 使用控制器和视图 (第 1 部分)	535
19.2 理解 RESTful Web 服务	474	21.1 准备工作	536
19.2.1 理解请求 URL 和方法	474	21.1.1 删除数据库	537
19.2.2 理解 JSON	475	21.1.2 运行示例应用程序	537
19.3 使用自定义端点创建 Web 服务	475	21.2 开始使用视图	538
19.4 使用控制器创建 Web 服务	478	21.2.1 配置应用程序	538
19.4.1 启用 MVC 框架	479	21.2.2 创建 HTML 控制器	539
19.4.2 创建控制器	480	21.2.3 创建 Razor 视图	542
19.5 改进 Web 服务	489	21.2.4 通过名称选择视图	544
19.5.1 使用异步操作	490	21.3 使用 Razor 视图	548
19.5.2 防止过度绑定	491	21.4 理解 Razor 语法	556
		21.4.1 理解指令	556
		21.4.2 理解内容表达式	557
		21.4.3 设置元素内容	557

21.4.4	设置特性值.....	558	23.5.1	为 Razor Pages 创建布局.....	623
21.4.5	使用条件表达式.....	559	23.5.2	在 Razor Pages 中使用分部视图.....	625
21.4.6	枚举序列.....	563	23.5.3	创建没有页面模型的 Razor Pages.....	627
21.4.7	使用 Razor 代码块.....	565	23.6	小结.....	628
21.5	小结.....	566	第 24 章	使用视图组件.....	629
第 22 章	使用控制器和视图(第 2 部分)...	567	24.1	准备工作.....	629
22.1	准备工作.....	567	24.1.1	删除数据库.....	632
22.1.1	删除数据库.....	569	24.1.2	运行示例应用程序.....	632
22.1.2	运行示例应用程序.....	569	24.2	理解视图组件.....	633
22.2	使用 ViewBag.....	570	24.3	创建和使用视图组件.....	633
22.3	使用临时数据.....	572	24.4	理解视图组件的结果.....	637
22.4	使用布局.....	574	24.4.1	返回一个分部视图.....	638
22.4.1	使用 ViewBag 配置布局.....	576	24.4.2	返回 HTML 片段.....	641
22.4.2	使用 ViewStart 文件.....	578	24.5	获取上下文数据.....	643
22.4.3	覆盖默认布局.....	579	24.5.1	使用实参提供父视图的上下文.....	645
22.4.4	使用布局节.....	583	24.5.2	创建异步视图组件.....	648
22.5	使用分部视图.....	590	24.6	创建视图组件类.....	649
22.5.1	启用分部视图.....	590	24.7	小结.....	655
22.5.2	创建分部视图.....	590	第 25 章	使用标签助手.....	657
22.5.3	应用分部视图.....	591	25.1	准备工作.....	658
22.6	理解内容编码.....	594	25.1.1	删除数据库.....	660
22.6.1	理解 HTML 编码.....	594	25.1.2	运行示例应用程序.....	660
22.6.2	理解 JSON 编码.....	596	25.2	创建标签助手.....	660
22.7	小结.....	597	25.2.1	定义标签助手类.....	661
第 23 章	使用 Razor Pages.....	599	25.2.2	注册标签助手.....	663
23.1	准备工作.....	600	25.2.3	使用标签助手.....	664
23.2	理解 Razor Pages.....	601	25.2.4	缩小标签助手的范围.....	665
23.2.1	配置 Razor Pages.....	601	25.2.5	扩展标签助手的范围.....	666
23.2.2	创建 Razor Pages.....	603	25.3	高级标签助手功能.....	668
23.3	理解 Razor Pages 的路由.....	607	25.3.1	创建快捷元素.....	668
23.3.1	在 Razor Pages 中指定路由模式.....	609	25.3.2	以编程方式创建元素.....	671
23.3.2	为 Razor Pages 添加路由.....	610	25.3.3	追加、附加内容和元素.....	672
23.4	理解页面模型类.....	612	25.3.4	获取视图上下文数据.....	675
23.4.1	使用代码隐藏类文件.....	613	25.3.5	使用模型表达式.....	678
23.4.2	理解 Razor Pages 的操作结果.....	615	25.3.6	标签助手之间的协调.....	682
23.4.3	处理多个 HTTP 方法.....	619	25.3.7	抑制输出元素.....	684
23.4.4	选择处理程序方法.....	621	25.4	使用标签助手组件.....	686
23.5	理解 Razor Pages 视图.....	623			

25.4.1	创建标签助手组件	686	27.5	使用 label 元素	739
25.4.2	展开标签助手的元素选择	688	27.6	使用 select 和 option 元素	741
25.5	小结	690	27.7	处理文本区域	745
第 26 章	使用内置的标签助手	691	27.8	使用防伪功能	746
26.1	准备工作	691	27.8.1	在控制器中启用防伪功能	747
26.1.1	添加图像文件	693	27.8.2	在 Razor Pages 中启用 防伪功能	749
26.1.2	安装客户端包	694	27.8.3	使用 JavaScript 客户端 防伪令牌	750
26.1.3	删除数据库	694	27.9	小结	753
26.1.4	运行示例应用程序	694	第 28 章	使用模型绑定	755
26.2	启用内置的标签助手	695	28.1	准备工作	756
26.3	改变锚元素	695	28.1.1	删除数据库	757
26.4	使用 JavaScript 和 CSS 标签助手	699	28.1.2	运行示例应用程序	757
26.4.1	管理 JavaScript 文件	699	28.2	理解模型绑定	757
26.4.2	管理 CSS 样式表	706	28.3	绑定简单数据类型	759
26.5	处理图像元素	709	28.3.1	绑定 Razor Pages 中的简单 数据类型	760
26.6	使用数据缓存	710	28.3.2	理解默认绑定值	762
26.6.1	设置缓存到期时间	712	28.4	绑定复杂类型	764
26.6.2	设置固定的过期点	713	28.4.1	绑定到属性	766
26.6.3	设置最后使用的有效期	713	28.4.2	绑定嵌套的复杂类型	768
26.6.4	使用缓存的变化	714	28.4.3	选择性的绑定属性	772
26.7	使用宿主环境标签助手	715	28.5	绑定到数组和集合	775
26.8	小结	716	28.5.1	绑定到数组	775
第 27 章	使用表单标签助手	717	28.5.2	绑定到简单集合	778
27.1	准备工作	717	28.5.3	绑定到字典	780
27.1.1	删除数据库	719	28.5.4	绑定到复杂类型的集合	781
27.1.2	运行示例应用程序	719	28.6	指定模型绑定源	784
27.2	理解表单处理模式	720	28.6.1	选择属性的绑定源	786
27.2.1	创建控制器来处理表单	721	28.6.2	使用标头进行模型绑定	787
27.2.2	创建 Razor Pages 来 处理表单	723	28.6.3	使用请求体作为绑定源	788
27.3	使用标签助手改进 HTML 表单	725	28.7	手动模式绑定	789
27.3.1	使用表单元素	725	28.8	小结	791
27.3.2	改变表单按钮	727	第 29 章	使用模型验证	793
27.4	处理 input 元素	728	29.1	准备工作	794
27.4.1	转换 input 元素的类型属性	730	29.2	删除数据库	795
27.4.2	格式化 input 元素值	732	29.3	运行示例应用程序	795
27.4.3	在 input 元素中显示相关 数据的值	735	29.4	理解对模型验证的需要	796
			29.5	显式验证控制器中的数据	796

29.5.1	向用户显示验证错误	799
29.5.2	显示验证消息	802
29.5.3	显示属性级的验证消息	806
29.5.4	显示模型级消息	807
29.6	显式验证 Razor Pages 中的数据	810
29.7	使用元数据指定验证规则	813
29.8	执行客户端验证	821
29.9	执行远程验证	823
29.10	小结	828
第 30 章	使用过滤器	829
30.1	准备工作	829
30.1.1	启用 HTTPS 连接	831
30.1.2	删除数据库	832
30.1.3	运行示例应用程序	833
30.2	使用过滤器	833
30.3	理解过滤器	838
30.4	创建自定义过滤器	839
30.4.1	理解授权过滤器	840
30.4.2	理解资源过滤器	842
30.4.3	理解操作过滤器	846
30.4.4	理解页面过滤器	850
30.4.5	理解结果过滤器	855
30.4.6	理解异常过滤器	859
30.4.7	创建异常过滤器	860
30.5	管理过滤器生命周期	863
30.5.1	创建过滤器工厂	865
30.5.2	使用依赖注入范围来管理过滤器的生命周期	866
30.6	创建全局过滤器	868
30.7	理解和改变过滤器的顺序	870
30.8	小结	874
第 31 章	创建表单应用程序	875
31.1	准备工作	875
31.1.1	删除数据库	878
31.1.2	运行示例应用程序	878
31.2	创建 MVC 表单应用程序	879
31.2.1	准备视图模型和视图	879
31.2.2	读取数据	881
31.2.3	创建数据	883

31.2.4	编辑数据	887
31.2.5	删除数据	890
31.3	创建 Razor Pages 表单应用程序	892
31.3.1	创建常用功能	893
31.3.2	为 CRUD 操作定义页面	896
31.4	创建新的相关数据对象	899
31.4.1	在同一请求中提供相关数据	900
31.4.2	创建新数据	903
31.5	小结	908

第 IV 部分 高级 ASP.NET Core 功能

第 32 章	创建示例项目	911
32.1	创建项目	911
32.2	添加数据模型	912
32.2.1	准备种子数据	914
32.2.2	配置 Entity Framework Core 服务和中间件	916
32.2.3	创建和应用迁移	917
32.3	添加引导 CSS 框架	918
32.4	配置服务和中间件	918
32.5	创建控制器和视图	919
32.6	创建 Razor Pages	922
32.7	运行示例应用程序	924
32.8	小结	925
第 33 章	使用 Blazor 服务器 (第 1 部分)	927
33.1	准备工作	928
33.2	理解 Blazor 服务器	928
33.2.1	理解 Blazor 服务器的优势	929
33.2.2	理解 Blazor 服务器的缺点	930
33.2.3	在 Blazor 服务器和 Angular/React/Vue.js 之间选择	930
33.3	从 Blazor 开始	930
33.3.1	为 Blazor 服务器配置 ASP.NET Core	930
33.3.2	创建 Blazor 组件	933
33.4	理解 Razor 组件的基本特性	938

33.4.1 理解 Blazor 事件和数据 绑定	938	第 36 章 Blazor 表单和数据	1023
33.4.2 使用数据绑定	946	36.1 准备工作	1023
33.5 使用类文件定义组件	951	36.2 使用 Blazor 表单组件	1027
33.5.1 使用代码隐藏类	951	36.2.1 创建自定义表单组件	1029
33.5.2 定义 Razor 组件类	953	36.2.2 验证表单数据	1033
33.6 小结	954	36.2.3 处理表单事件	1036
第 34 章 使用 Blazor 服务器 (第 2 部分)	955	36.3 使用 Entity Framework Core 与 Blazor	1038
34.1 准备工作	955	36.3.1 理解 Entity Framework Core 上下文范围问题	1039
34.2 结合组件	956	36.3.2 理解重复查询问题	1043
34.2.1 利用属性配置组件	958	36.4 执行创建、读取、更新和 删除操作	1049
34.2.2 创建自定义事件和绑定	963	36.4.1 创建 List 组件	1049
34.3 在组件中显示子内容	968	36.4.2 创建 Details 组件	1050
34.3.1 创建模板组件	970	36.4.3 创建 Editor 组件	1052
34.3.2 在模板组件中使用泛型 类型参数	972	36.5 扩展 Blazor 表单特性	1054
34.3.3 级联参数	978	36.5.1 创建自定义验证约束	1055
34.4 处理错误	981	36.5.2 创建只验证提交按钮 组件	1058
34.4.1 处理连接错误	981	36.6 小结	1060
34.4.2 处理未捕获的应用程序 错误	984	第 37 章 使用 Blazor WebAssembly	1061
34.5 小结	986	37.1 准备工作	1062
第 35 章 高级 Blazor 特性	987	37.2 设置 Blazor WebAssembly	1064
35.1 准备工作	988	37.2.1 创建共享项目	1064
35.2 使用组件的路由	988	37.2.2 创建 Blazor WebAssembly 项目	1065
35.2.1 准备 Blazor 页	989	37.2.3 准备 ASPNETCore 项目	1065
35.2.2 向组件添加路由	990	37.2.4 添加解决方案引用	1066
35.2.3 在路由组件之间导航	993	37.2.5 打开项目	1066
35.2.4 接收路由数据	996	37.2.6 完成 Blazor WebAssembly 配置	1067
35.2.5 使用布局定义公共内容	998	37.2.7 测试占位符组件	1069
35.3 理解组件生命周期方法	1000	37.3 创建 Blazor WebAssembly 组件	1070
35.4 管理组件的交互	1005	37.3.1 导入数据模型名称空间	1070
35.4.1 使用子组件的引用	1005	37.3.2 创建组件	1070
35.4.2 与来自其他代码的组件 交互	1008	37.3.3 创建布局	1074
35.4.3 使用 JavaScript 与组件 交互	1012	37.3.4 定义 CSS 样式	1075
35.5 小结	1022	37.4 完成 Blazor WebAssembly 表单 应用程序	1076

37.4.1	创建 Details 组件	1076		
37.4.2	创建 Editor 组件	1077		
37.5	小结	1080		
第 38 章	使用 ASP.NET Core Identity	1081		
38.1	准备工作	1082		
38.2	为 ASP.NET Core Identity 准备项目	1083		
38.2.1	准备 ASP.NET Core Identity 数据库	1083		
38.2.2	配置数据库连接字符串	1083		
38.2.3	配置应用程序	1084		
38.2.4	创建和应用身份数据库 迁移	1086		
38.3	创建用户管理工具	1086		
38.3.1	准备用户管理工具	1087		
38.3.2	枚举用户账户	1088		
38.3.3	创建用户	1090		
38.3.4	编辑用户	1097		
38.3.5	删除用户	1099		
38.4	创建角色管理工具	1100		
38.4.1	为角色管理工具做准备	1101		
38.4.2	枚举和删除角色	1102		
38.4.3	创建角色	1103		
38.4.4	分配角色从属关系	1104		
38.5	小结	1107		
第 39 章	应用 ASP.NET Core Identity	1109		
39.1	验证用户的身份	1111		
39.1.1	创建登录特性	1111		
39.1.2	检查 ASP.NET Core Identity cookie	1113		
39.1.3	创建退出页面	1114		
39.1.4	测试身份验证特性	1115		
39.1.5	启用身份验证中间件	1116		
39.2	对授权端点的访问	1118		
39.2.1	应用授权属性	1118		
39.2.2	启用授权中间件	1119		
39.2.3	创建被拒绝访问的端点	1120		
39.2.4	创建种子数据	1120		
39.2.5	测试身份验证序列	1123		
39.3	授权访问 Blazor 应用程序	1124		
39.3.1	在 Blazor 组件中执行 授权	1125		
39.3.2	向授权用户显示内容	1127		
39.4	对 Web 服务进行身份验证和 授权	1129		
39.4.1	构建简单的 JavaScript 客户端	1132		
39.4.2	限制对 Web 服务的访问	1134		
39.4.3	使用 cookie 验证	1135		
39.4.4	使用令牌认证	1138		
39.4.5	创建令牌	1139		
39.4.6	用令牌验证	1141		
39.4.7	使用令牌限制访问	1144		
39.4.8	使用令牌请求数据	1145		
39.5	小结	1147		

第 I 部分



介绍 ASP.NET Core

- 第 1 章 ASP.NET Core 上下文
- 第 2 章 入门
- 第 3 章 第一个 ASP.NET Core 应用程序
- 第 4 章 使用开发工具
- 第 5 章 C# 的基本特点
- 第 6 章 测试 ASP.NET Core 应用程序
- 第 7 章 SportsStore：一个真正的应用程序
- 第 8 章 SportsStore：导航和购物车
- 第 9 章 SportsStore：完成购物车
- 第 10 章 SportsStore：管理
- 第 11 章 SportsStore：安全与部署

第 1 章



ASP.NET Core 上下文

1.1 了解 ASP.NET Core

ASP.NET Core 是微软的 Web 开发平台。原来的 ASP.NET 是在 2002 年引入的，它经过了几次再创造和转世才成为 ASP.NET Core 3，这是本书的主题。

ASP.NET Core 由一个处理 HTTP 请求的平台、一系列用于创建应用程序的主要框架和提供支持特性的次要实用程序框架组成，如图 1-1 所示。

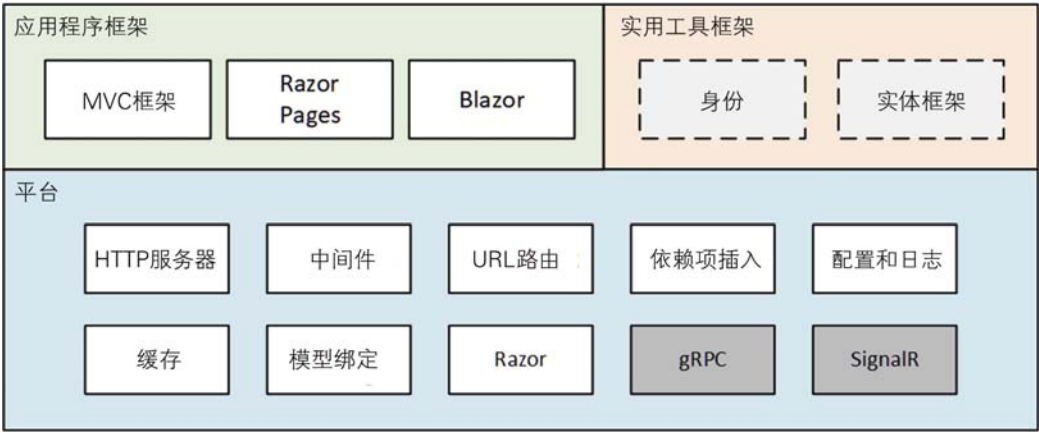


图 1-1 ASP.NET Core 的结构

1.1.1 理解应用程序框架

开始使用 ASP.NET Core，发现有不同的应用框架可用时，可能会感到困惑。其实，这些框架是互补的，可以解决不同的问题，或者，对于某些特性，可采用不同方式解决相同的问题。理解这些框架之间的关系意味着理解 Microsoft 所支持的不断变化的设计模式，如下面各节所述。

1. 理解 MVC 框架

MVC 框架是在 ASP.NET Core 以前的时代引入的。原来的 ASP.NET 依赖于一种称为 Web Pages 的开发模型，这种模型重新创建了编写桌面应用程序的体验，但导致了无法很好扩展的、笨拙的

Web 项目。MVC 框架与 Web 页面一起引入，开发模型包含 HTTP 和 HTML 的特性，而不是试图隐藏它们。

MVC 代表 Model-View-Controller，是一种描述应用程序形状的设计模式。MVC 模式强调关注点分离，其中功能区域是独立定义的，这是对 Web 页面导致的模糊架构的有效解决方案。

早期版本的 MVC 框架建立在 ASP.NET 基础之上，最初是为 Web 页面设计的，这导致了一些笨拙的特性和解决方案。随着向 .NET Core 的迁移，ASP.NET 成为 ASP.NET Core，MVC 框架是在一个开放的、可扩展的、跨平台的基础上重新构建的。

MVC 框架仍然是 ASP.NET Core 的重要组成部分。但是，随着单页应用程序(SPA)的兴起，它通常的使用方式已经发生了变化。在 SPA 中，浏览器发出一个单独的 HTTP 请求，然后接收一个 HTML 文档，该文档将提供一个富客户端，通常是用 JavaScript 客户端编写的，比如 Angular 或 React。SPA 的转变意味着 MVC 框架最初追求的清晰分离并不重要，强调遵循 MVC 模式不再是有必要的，但 MVC 框架仍然是有用的(用于通过 Web 服务支持 SPA，详见第 19 章)。

2. 把模式放在应有的位置上

模式只是其他人在其他项目中遇到的问题的解决方案。如果自己也面临同样的问题，了解以前是如何解决的会很有帮助。但这并不意味着必须完全遵循这个模式，或者根本不遵循，只要明白后果就行。例如，如果模式的目的是使项目可管理，而选择背离该模式，就必须接受项目可能更难管理这一事实。但是盲目地遵循模式可能比没有模式更糟糕，而且没有模式适合每个项目。

建议自由地使用模式，根据需要调整它们，忽略那些混淆模式和戒律的狂热分子。

3. 理解 Razor Pages

MVC 框架的一个缺点是，在应用程序开始生成内容之前，它可能需要进行大量的准备工作。尽管存在结构上的问题，但 Web 页面的一个优点是可以在几小时内创建简单的应用程序。

Razor Pages 采用了 Web 页面的开发风格，并使用最初为 MVC 框架开发的平台特性来实现它。代码和内容混合在一起，形成自包含的页面；这重新提高了 Web 页面的开发速度，而不存在一些潜在的技术问题(尽管扩大复杂项目的规模仍然是个问题)。

Razor Pages 可以与 MVC 框架一起使用，这也是笔者倾向于使用的方式。笔者使用 MVC 框架编写应用程序的主要部分，使用 Razor Pages 编写次要功能，比如管理和报告工具。第 7~11 章就使用了这种方法，那几章开发了一个现实的 ASP.NET Core 应用程序，名为 SportsStore。

4. 理解 Blazor

JavaScript 客户端框架的兴起对 C#开发人员来说是一个障碍，他们必须学习一种不同的、有些特殊的编程语言。笔者已经爱上 JavaScript，它和 C#一样流畅、富有表现力。但是，精通一种新的编程语言需要时间和投入，尤其是与 C#有根本区别的语言。

Blazor 试图通过允许将 C#用于编写客户端应用程序来弥补这一差距。Blazor 有两个版本：Blazor Server 和 Blazor WebAssembly。Blazor Server 是 ASP.NET Core 中稳定且受支持的部分。它通过使用到 ASP.NET Core 的持久 HTTP 连接工作。应用程序的 C#代码在这里执行。Blazor WebAssembly 是一个实验性版本，它更进一步，在浏览器中执行应用程序的 C#代码。如第 33 章所述，Blazor 的两种版本都不适合所有的情况，但它们都为 ASP.NET Core 开发的未来指明了方向。

1.1.2 理解实用程序框架

有两个框架与 ASP.NET Core 密切相关，但不直接用于生成 HTML 内容或数据。Entity Framework Core 是 Microsoft 的对象-关系映射(ORM)框架，它将存储在关系数据库中的数据表示为 .NET 对象。Entity Framework Core 可以在任何 .NET Core 应用程序中使用，它通常用于访问 ASP.NET Core 应用程序中的数据库。

ASP.NET Core Identity 是 Microsoft 的认证和授权框架，用于验证 ASP.NET Core 应用程序中的用户凭证，限制对应用程序特性的访问。

本书只描述了这两个框架的基本特性，重点是大多数 ASP.NET Core 应用程序所需要的功能，但它们都是复杂的框架，太大了，无法在一本关于 ASP.NET Core 的书中详细描述。

未来版本的主题

本书没有足够的篇幅介绍每一个 Entity Framework Core 和 ASP.NET Core Identity 特性，而只关注大多数项目都需要的方面。如果你认为某主题应该包括在本书的下一版或进行深度解析，请把建议发送到 adam@adam-freeman.com。

1.1.3 了解 ASP.NET Core 平台

ASP.NET Core 平台包含接收和处理 HTTP 请求以及创建响应所需的底层特性。有一个集成的 HTTP 服务器、一个处理请求的中间件组件系统和应用程序框架所依赖的核心特性，如 URL 路由和 Razor 视图引擎。

大部分开发时间都花费在应用程序框架上，但为了有效使用 ASP.NET Core，需要理解该平台提供的强大功能；没有这些功能，高层框架就无法运行。本书第 II 部分详细介绍 ASP.NET Core 平台的工作原理，解释它提供的特性是如何支撑 ASP.NET Core 开发的各个方面的。

本书没有描述两个值得注意的平台特性：SignalR 和 gRPC。SignalR 用于在应用程序之间创建延迟通信通道，为本书第 IV 部分中描述的 Blazor 服务器框架提供了基础。但我们很少直接使用 SignalR，对于那些需要低延迟消息传递的项目，有更好的替代方案，如 Azure Event Grid 或 Azure Service Bus。

gRPC 是基于 HTTP 的跨平台远程过程调用(RPC)的新兴标准，最初由谷歌(gRPC 中的 g)创建，提供了效率和可伸缩性的优势。gRPC 可能是 Web 服务的未来标准，但它不能在 Web 应用程序中使用，因为它需要对它发送的 HTTP 消息进行低级控制，而这是浏览器不允许的(有一个浏览器库允许 gRPC 通过代理服务器使用，但这破坏了使用 gRPC 的好处)。直到 gRPC 可以在浏览器中使用，它才会包含在 ASP.NET Core 中，但只适用于使用它进行后端服务器之间通信的项目；对于这些项目，存在许多替代协议。当 gRPC 可以在浏览器中使用或成为主要的数据中心协议，本书的未来版本会介绍它。

1.2 理解本书

为从本书得到最大的收获，应该熟悉 Web 开发的基础知识，理解 HTML 和 CSS 是如何工作的，并掌握 C# 的工作原理。如果没有进行过任何客户端开发(如 JavaScript)，请不要担心。本书的重点是 C# 和 ASP.NET Core，随着章节的进展，读者可以获得需要知道的一切。第 5 章总结了 C#

对 ASP.NET Core 开发最重要的特性,如果读者从 .NET Core 或者 .NET Framework 的早期版本转到 ASP.NET Core, 这些特性都很有用。

1.2.1 需要什么软件来完成示例?

需要一个代码编辑器(Visual Studio 或 Visual Studio Code)、.NET 核心软件开发工具包和 SQL Server LocalDB。所有这些都可以从微软免费使用,第 2 章包含了所需软件的安装指令。

1.2.2 需要什么平台来完成示例?

本书是为 Windows 编写的,使用的是 Windows 10 Pro,但是 Visual Studio、Visual Studio Code 和 .NET Core 支持的任何版本的 Windows 都可以工作。其他平台也支持 ASP.NET Core,但本书中的示例依赖于 SQL Server LocalDB 特性,这是 Windows 特有的。如果试图使用另一个平台,可以通过 adam@adam-freeman.com 联系笔者,笔者会给你一些修改示例的一般指示,但有一点要注意,如果你陷入困境,笔者不能提供详细的帮助。

1.2.3 源代码下载

读者可访问本书的 GitHub 存储库(<https://github.com/apress/pro-asp.net-core-3>)下载源代码,也可扫描封底的二维码下载。

1.2.4 如果在执行这些示例时遇到问题,怎么办?

首先要做的是回到这一章的开头,重新开始。大多数问题是由于缺少一个步骤或没有完全按照清单执行而引起的。请密切关注代码清单中的重点部分,它会突出显示所需的更改。

接下来,检查该书 GitHub 存储库中包含的勘误表/更正列表。技术书籍是复杂的,尽管笔者和编辑尽了最大努力,错误是不可避免的。检查勘误表列表,获得已知错误列表和解决这些错误的指令。

如果仍然有问题,那么从本书的 GitHub 存储库(<https://github.com/apress/pro-asp.net-core-3>)下载正在阅读的章节的项目,并将其与项目进行比较。笔者通过遍历每一章来创建 GitHub 存储库的代码,因此项目中的相同文件应该具有相同的内容。

如果示例仍然不能工作,可以联系笔者(adam@adam-freeman.com)寻求帮助。请在电子邮件中说明你正在阅读的是哪本书,是哪一章/例子导致了这个问题。请记住,笔者会收到很多电子邮件,可能不会立即回复。

1.2.5 如果发现书中有错误,怎么办?

可以通过向 adam@adam-freeman.com 发电子邮件,向笔者报告错误,不过读者应先检查本书的勘误表/修正列表,本书的 GitHub 库可以在如下地址找到:

<https://github.com/apress/pro-asp.net-core-3>

你遇到的错误可能已经在该列表中列出。

1.2.6 本书包含的内容

本书试图涵盖大多数 ASP.NET Core 项目都需要的功能。它分为四个部分,每一部分都涵盖

一系列相关的主题。

第 I 部分：介绍 ASP.NET Core

本书的这一部分——包括本章——介绍了 ASP.NET Core。除了设置开发环境和创建第一个应用程序之外，还将了解对 ASP.NET Core 开发最重要的 C# 特性，和如何使用 ASP.NET Core 开发工具。但是第 I 部分的大部分内容是关于一个名为 SportsStore 的项目的开发，通过这个项目，展示了从开始到部署的实际开发过程，涉及 ASP.NET Core 的所有主要特性，并展示它们是如何结合在一起的。

第 II 部分：ASP.NET Core 平台

该部分的章节描述 ASP.NET Core 平台的主要特性，解释如何处理 HTTP 请求，如何创建和使用中间件组件，如何创建路由，如何定义和使用服务，以及如何与 Entity Framework Core 一起工作。解释 ASP.NET Core 的基础知识。理解它们对于有效的 ASP.NET Core 开发是至关重要的。

第 III 部分：ASP.NET Core 应用程序

该部分的章节解释如何创建不同类型的应用程序，包括 RESTful Web 服务以及使用控制器和 Razor Pages 的 HTML 应用程序。这些章节还描述易于生成 HTML 的特性，包括视图、视图组件和标签助手。

第 IV 部分：高级 ASP.NET Core 功能

本书最后一部分解释如何使用 Blazor 服务器创建应用程序，如何使用实验性的 Blazor WebAssembly，以及如何使用 ASP.NET Core 验证用户身份和授予访问权限。

1.2.7 本书未包含的内容

本书没有涵盖基本的 Web 开发主题，如 HTML 和 CSS，也没有详细讲述 C# (第 5 章描述了对 ASP.NET Core 开发有用的 C# 特性，使用旧版本 .NET 的开发人员可能不熟悉这些特性)。

虽然笔者喜欢在书中钻研细节，但不是每个 ASP.NET Core 特性在主流开发中很有用，笔者必须将本书保持适当的厚度范围内。当笔者决定去掉某个特性时，是因为认为它不重要，或者因为使用所介绍的技术可达到同样的结果。

如前所述，本书没有描述 ASP.NET Core 对 SignalR 和 gRPC 的支持，后续章节提到了没有描述的其他特性，这要么是因为它们不能广泛应用，要么是因为有更好的替代方法可用。每种情况下，笔者都会解释为什么省略了描述，并提供了关于该主题的可供参考的 Microsoft 文档。

1.2.8 如何联系作者？

可通过 adam@adam-freeman.com 给笔者发邮件。从第一次在书中公布电子邮件地址以来，到现在已经有好几年了。不能完全肯定这是个好主意，但很高兴笔者这么做了。笔者收到来自世界各地的电子邮件，读者来自各个行业，大部分邮件都是积极的、礼貌的，收到邮件令人愉悦。

笔者试着迅速回复邮件，但邮件太多，有时还会积压大量邮件，尤其是当笔者埋头写书的时候。笔者总是试图帮助那些被书中的例子困住的读者，尽管读者在联系之前，应按照本章前面描述的步骤来做。

虽然笔者欢迎读者发电子邮件，但对于某些问题，笔者的回答总是否定的。笔者恐怕不会为新公司写代码，帮助完成大学作业，或参与某个开发团队的设计争议。

1.2.9 如果你真的喜欢本书？

请发邮件到 adam@adam-freeman.com，让笔者知道。收到快乐读者的来信总是一件让人高兴的事，笔者也很感激读者花时间发送这些邮件。写书很难，这些电子邮件是笔者前进的基本动力。

1.2.10 如果本书让人生气，想要抱怨该怎么办？

仍然可以通过 adam@adam-freeman.com 发邮件，笔者仍然会尽力提供帮助。请记住，只有在解释问题是什么、想让笔者怎么做的情况下，笔者才能提供帮助。有时候唯一的结果就是接受本书不是专门为每个人写的，只有放下本书，选择另一本书，事情才会结束。笔者会仔细考虑让人心烦的事情，但在写了 25 年书之后，笔者开始明白，不是每个人都喜欢读我的书。

1.3 小结

本章为本书其余部分做了铺垫。简要介绍了 ASP.NET Core，说明了本书的要求和内容，以及如何联系笔者。第 2 章将展示如何准备 ASP.NET Core 开发。