

# Оглавление

---

Часть I	■ НАЧАЛО РАБОТЫ С ASP.NET CORE.....	31
1	■ Начало работы с ASP.NET Core.....	33
2	■ Ваше первое приложение .....	58
3	■ Обработка запросов с помощью конвейера промежуточного ПО.....	95
4	■ Создание веб-сайта с помощью страниц Razor.....	130
5	■ Сопоставление URL-адресов с Razor Pages с использованием маршрутизации .....	164
6	■ Модель привязки: получение и проверка пользовательского ввода.....	203
7	■ Визуализация HTML-кода с использованием представлений Razor ....	239
8	■ Создание форм с помощью тег-хелперов .....	278
9	■ Создание веб-API для мобильных и клиентских приложений с помощью MVC .....	313
Часть II	■ СОЗДАНИЕ ПОЛНОЦЕННЫХ ПРИЛОЖЕНИЙ .....	350
10	■ Конфигурация сервисов с помощью внедрения зависимостей .....	352
11	■ Конфигурирование приложения ASP.NETCore .....	394
12	■ Сохраняем данные с Entity Framework Core .....	432
13	■ Конвейер фильтров MVC и Razor Pages .....	471
14	■ Аутентификация: добавляем пользователей в приложение с помощью ASP.NET Core Identity .....	513
15	■ Авторизация: обеспечиваем защиту приложения .....	553
16	■ Публикация и развертывание приложения .....	590
Часть III	■ РАСШИРЕНИЕ ПРИЛОЖЕНИЙ.....	628
17	■ Мониторинг и устранение ошибок с помощью журналирования ....	630
18	■ Повышаем безопасность приложения .....	667
19	■ Создание специальных компонентов .....	710
20	■ Создание специальных компонентов MVC и Razor Pages .....	745
21	■ Вызов удаленных API с помощью IHttpConnectionFactory .....	775
22	■ Создание фоновых задач и сервисов .....	799
23	■ Тестирование приложения .....	827

# Содержание

---

Предисловие .....	19
Благодарности.....	21
Об этой книге .....	23
Об авторе.....	29
Об иллюстрации на обложке .....	30

## Часть I НАЧАЛО РАБОТЫ С ASP.NET CORE ..... 31

<b>1</b> <b>Начало работы с ASP.NET Core</b> .....	33
1.1 Введение в ASP.NET Core .....	34
1.1.1 Использование веб-фреймворка.....	35
1.1.2 Что такое ASP.NET Core?.....	37
1.2 Когда следует отдать предпочтение ASP.NET Core.....	40
1.2.1 Какие типы приложений можно создавать? .....	40
1.2.2 Если вы новичок в разработке на .NET.....	43
1.2.3 Если вы разработчик, создающий новое приложение .....	45
1.2.4 Перенос существующего ASP.NET-приложения на ASP.NET Core.....	50
1.3 Как работает ASP.NET Core?.....	51
1.3.1 Как работает веб-запрос по протоколу HTTP? .....	52
1.3.2 Как ASP.NET Core обрабатывает запрос? .....	54
1.4 Что вы узнаете из этой книги .....	56
Резюме .....	57

<b>2</b> <b>Ваше первое приложение</b> .....	58
2.1 Краткий обзор приложения ASP.NET Core .....	60
2.2 Создание вашего первого приложения ASP.NET Core.....	62
2.2.1 Использование шаблона .....	63
2.2.2 Сборка приложения.....	66
2.3 Запуск веб-приложения.....	68
2.4 Разбираемся с макетом проекта .....	70

2.5	Файл проекта .csproj: определение зависимостей.....	71
2.6	Класс Program: сборка веб-хоста .....	74
2.7	Класс Startup: настройка вашего приложения.....	77
2.7.1	Добавление и настройка сервисов .....	79
2.7.2	Определяем, как обрабатываются запросы с помощью промежуточного программного обеспечения.....	81
2.8	Создание ответов с помощью Razor Pages.....	86
2.8.1	Создание HTML с помощью страниц Razor .....	87
2.8.2	Логика обработки запросов с помощью PageModels и обработчиков .....	89
	Резюме .....	93

### **3** **Обработка запросов с помощью конвейера промежуточного ПО**..... 95

3.1	Что такое промежуточное ПО? .....	97
3.2	Объединение компонентов в конвейер .....	102
3.2.1	Простой сценарий конвейера 1: страница приветствия .....	102
3.2.2	Простой сценарий конвейера 2: обработка статических файлов ...	106
3.2.3	Простой сценарий конвейера 3: приложение со страницами Razor .....	110
3.3	Обработка ошибок с помощью промежуточного ПО.....	116
3.3.1	Просмотр исключений в окружении разработки: DeveloperExceptionHandler.....	118
3.3.2	Обработка исключений в промышленном окружении: ExceptionHandlerMiddleware .....	119
3.3.3	Обработка других ошибок: StatusCodePagesMiddleware .....	124
3.3.4	Компонент обработки ошибок и веб-API.....	128
	Резюме .....	129

### **4** **Создание веб-сайта с помощью страниц Razor**..... 130

4.1	Введение в Razor Pages .....	132
4.1.1	Изучение типичной страницы Razor .....	132
4.1.2	Паттерн проектирования MVC.....	134
4.1.3	Применение паттерна проектирования MVC к Razor Pages .....	137
4.1.4	Добавление Razor Pages в приложение.....	145
4.2	Сравнение Razor Pages и MVC в ASP.NET Core .....	149
4.2.1	Контроллеры MVC в ASP.NET Core .....	149
4.2.2	Преимущества Razor Pages.....	151
4.2.3	Когда выбирать контроллеры MVC вместо Razor Pages .....	154
4.3	Razor Pages и обработчики страниц .....	155
4.3.1	Прием параметров в обработчиках страниц.....	157
4.3.2	Возврат ответов с помощью ActionResult.....	159
	Резюме .....	163

### **5** **Сопоставление URL-адресов с Razor Pages с использованием маршрутизации**..... 164

5.1	Что такое маршрутизация?.....	165
5.2	Маршрутизация в ASP.NET Core .....	169

5.2.1	Использование маршрутизации конечных точек в ASP.NET Core	169
5.2.2	Маршрутизация на основе соглашений и маршрутизация на основе атрибутов	173
5.2.3	Маршрутизация и страницы Razor	176
5.3	Настройка шаблонов маршрутов для страницы Razor	178
5.3.1	Добавление сегмента в шаблон маршрута	180
5.3.2	Полная замена шаблона маршрута страницы Razor	181
5.4	Изучение синтаксиса шаблона маршрута	182
5.4.1	Использование дополнительных значений и значений по умолчанию	182
5.4.2	Добавление дополнительных ограничений к параметрам маршрута	184
5.4.3	Сопоставление произвольных URL-адресов с помощью универсального параметра	186
5.5	Генерация URL-адресов из параметров маршрута	188
5.5.1	Создание URL-адресов для страницы Razor	189
5.5.2	Создание URL-адресов для контроллера MVC	190
5.5.3	Создание URL-адресов с помощью ActionResult	192
5.5.4	Создание URL-адресов из других частей вашего приложения	193
5.6	Выбор обработчика страницы для вызова	194
5.7	Настройка соглашений с помощью Razor Pages	197
	Резюме	201

<b>6</b>	<b>Модель привязки: получение и проверка пользовательского ввода</b>	203
6.1	Модели в Razor Pages и MVC	204
6.2	От запроса к модели: делаем запрос полезным	208
6.2.1	Связывание простых типов	212
6.2.2	Привязка сложных типов	216
6.2.3	Выбор источника привязки	221
6.3	Обработка пользовательского ввода с помощью валидации модели	223
6.3.1	Необходимость валидации модели	223
6.3.2	Использование атрибутов DataAnnotations для валидации	225
6.3.3	Валидация модели на сервере в целях безопасности	228
6.3.4	Валидация на стороне клиента для улучшения пользовательского интерфейса	232
6.4	Организация моделей привязки в Razor Pages	234
	Резюме	237

<b>7</b>	<b>Визуализация HTML-кода с использованием представлений Razor</b>	239
7.1	Представления: визуализация пользовательского интерфейса	241
7.2	Создание представлений Razor	245
7.2.1	Представления Razor и сопутствующий код	245
7.2.2	Знакомство с шаблонами Razor	247
7.2.3	Передача данных в представления	248
7.3	Создание динамических веб-страниц с помощью Razor	251
7.3.1	Использование C# в шаблонах Razor	252

7.3.2	Добавление циклов и условий в шаблоны Razor .....	253
7.3.3	Визуализация HTML с помощью метода Raw .....	256
7.4	Макеты, частичные представления и <code>_ViewStart</code> .....	259
7.4.1	Использование макетов для общей разметки .....	260
7.4.2	Переопределение родительских макетов с помощью секций .....	262
7.4.3	Использование частичных представлений для инкапсуляции разметки .....	264
7.4.4	Выполнение кода в каждом представлении с помощью <code>_ViewStart</code> и <code>_ViewImports</code> .....	267
7.5	Выбор представления из контроллера MVC .....	270
	Резюме .....	276

<b>8</b>	<b>Создание форм с помощью тег-хелперов</b> .....	278
8.1	Редакторы кода и тег-хелперы .....	280
8.2	Создание форм с помощью тег-хелперов .....	283
8.2.1	Тег-хелпер формы .....	288
8.2.2	Тег-хелпер метки (label) .....	291
8.2.3	Тег-хелперы ввода (input) и области текста (textarea) .....	292
8.2.4	Тег-хелпер раскрывающегося списка .....	296
8.2.5	Тег-хелперы сообщений валидации и сводки сообщений (Validation Summary) .....	302
8.3	Создание ссылок с помощью тег-хелпера якоря (Anchor Tag Helper) .....	305
8.4	Сброс кеша с помощью тег-хелпера добавления версии (Append Version Tag Helper) .....	307
8.5	Использование условной разметки с помощью тег-хелпера окружения .....	308
	Резюме .....	310

<b>9</b>	<b>Создание веб-API для мобильных и клиентских приложений с помощью MVC</b> .....	313
9.1	Что такое веб-API, и когда его следует использовать? .....	314
9.2	Создание первого проекта веб-API .....	318
9.3	Применение паттерна проектирования MVC к веб-API .....	326
9.4	Маршрутизация на основе атрибутов: связывание методов действий с URL-адресами .....	330
9.4.1	Сочетание атрибутов маршрута, чтобы ваши шаблоны маршрутов следовали принципу DRY .....	333
9.4.2	Использование замены маркера для уменьшения дублирования при маршрутизации на основе атрибутов .....	334
9.4.3	Обработка HTTP-методов с помощью маршрутизации на основе атрибутов .....	335
9.5	Использование общепринятых соглашений с атрибутом [ApiController] .....	337
9.6	Генерация ответа от модели .....	341
9.6.1	Настройка форматов по умолчанию: добавляем поддержку XML .....	343
9.6.2	Выбор формата ответа с помощью согласования содержимого ....	345
	Резюме .....	347

## Часть II СОЗДАНИЕ ПОЛНОЦЕННЫХ ПРИЛОЖЕНИЙ ..... 350

<b>10</b>	<b>Конфигурация сервисов с помощью внедрения зависимостей</b> .....	352
10.1	Введение во внедрение зависимостей.....	353
10.1.1	Преимущества внедрения зависимостей .....	354
10.1.2	Создание слабосвязанного кода .....	360
10.1.3	Внедрение зависимостей в ASP.NET Core.....	362
10.2	Использование контейнера внедрения зависимостей.....	364
10.2.1	Добавление сервисов фреймворка ASP.NET Core в контейнер.....	364
10.2.2	Регистрация собственных сервисов в контейнере.....	367
10.2.3	Регистрация сервисов с использованием объектов и лямбда-функций .....	369
10.2.4	Множественная регистрация сервиса в контейнере .....	374
10.2.5	Внедрение сервисов в методы действий, обработчики страниц и представления.....	378
10.3	Жизненный цикл: когда создаются сервисы? .....	382
10.3.1	Transient: все уникально .....	385
10.3.2	Scoped: давайте держаться вместе .....	386
10.3.3	Singleton: может быть только один .....	387
10.3.4	Следите за захваченными зависимостями .....	388
	Резюме .....	392
<b>11</b>	<b>Конфигурирование приложения ASP.NETCore</b> .....	394
11.1	Представляем модель конфигурации ASP.NET Core.....	395
11.2	Конфигурирование приложения с помощью метода CreateDefaultBuilder.....	397
11.3	Создание объекта конфигурации для вашего приложения.....	399
11.3.1	Добавление поставщика конфигурации в файле Program.cs .....	402
11.3.2	Использование нескольких поставщиков для переопределения значений конфигурации .....	405
11.3.3	Безопасное хранение секретов конфигурации .....	407
11.3.4	Перезагрузка значений конфигурации при их изменении .....	412
11.4	Использование строго типизированных настроек с паттерном Options .....	413
11.4.1	Знакомство с интерфейсом IOptions .....	415
11.4.2	Перезагрузка строго типизированных параметров с помощью IOptionsSnapshot .....	417
11.4.3	Разработка классов параметров для автоматической привязки ...	418
11.4.4	Связывание строго типизированных настроек без интерфейса IOptions .....	420
11.5	Настройка приложения для нескольких окружений.....	422
11.5.1	Определение окружения размещения.....	422
11.5.2	Загрузка файлов конфигурации для конкретного окружения .....	424
11.5.3	Задаем окружение размещения .....	426
	Резюме .....	430

<b>12</b>	<b>Сохраняем данные с Entity Framework Core</b> .....	432
12.1	Знакомство с Entity Framework Core .....	434
12.1.1	Что такое EF Core? .....	434
12.1.2	Зачем использовать инструмент объектно-реляционного отображения? .....	436
12.1.3	Когда следует выбирать EF Core? .....	437
12.1.4	Отображение базы данных в код приложения .....	439
12.2	Добавляем EF Core в приложение .....	441
12.2.1	Выбор провайдера базы данных и установка EF Core .....	443
12.2.2	Создание модели данных .....	444
12.2.3	Регистрация контекста данных .....	447
12.3	Управление изменениями с помощью миграций .....	448
12.3.1	Создаем первую миграцию .....	449
12.3.2	Добавляем вторую миграцию .....	452
12.4	Выполнение запроса к базе данных и сохранение в ней данных .....	455
12.4.1	Создание записи .....	455
12.4.2	Загрузка списка записей .....	458
12.4.3	Загрузка одной записи .....	460
12.4.4	Обновление модели .....	462
12.5	Использование EF Core в промышленных приложениях .....	466
	Резюме .....	468
<b>13</b>	<b>Конвейер фильтров MVC и Razor Pages</b> .....	471
13.1	Что такое фильтры, и когда их использовать .....	473
13.1.1	Конвейер фильтров MVC .....	474
13.1.2	Конвейер фильтров Razor Pages .....	476
13.1.3	Фильтры или промежуточное ПО: что выбрать? .....	478
13.1.4	Создание простого фильтра .....	479
13.1.5	Добавляем фильтры к действиям, контроллерам, страницам Razor Pages и глобально .....	482
13.1.6	Порядок выполнения фильтров .....	485
13.2	Создание фильтров для приложения .....	487
13.2.1	Фильтры авторизации: защита API .....	490
13.2.2	Фильтры ресурсов: прерывание выполнения методов действий ....	492
13.2.3	Фильтры действий: настройка привязки модели и результатов действий .....	494
13.2.4	Фильтры исключений: собственная обработка исключений для методов действий .....	499
13.2.5	Фильтры результатов: настройка результатов действий перед их выполнением .....	501
13.2.6	Фильтры страниц: настройка привязки модели для Razor Pages .....	504
13.3	Прерывание выполнения конвейера .....	506
13.4	Использование внедрения зависимостей с атрибутами фильтра .....	508
	Резюме .....	511

<b>14</b>	<b>Аутентификация: добавляем пользователей в приложение с помощью ASP.NET Core Identity</b> .....	513
14.1	Знакомство с аутентификацией и авторизацией .....	515
14.1.1	Пользователи и утверждения в ASP.NET Core .....	515
14.1.2	Аутентификация в ASP.NET Core: сервисы и промежуточное ПО .....	517
14.1.3	Аутентификация для API и распределенных приложений .....	520
14.2	Что такое ASP.NET Core Identity? .....	524
14.3	Создание проекта, в котором используется ASP.NET Core Identity .....	527
14.3.1	Создание проекта из шаблона .....	527
14.3.2	Изучение шаблона в Обзорателе решений .....	529
14.3.3	Модель данных ASP.NET Core Identity .....	533
14.3.4	Взаимодействие с ASP.NET Core Identity .....	535
14.4	Добавляем ASP.NET Core Identity в существующий проект .....	538
14.4.1	Настройка сервисов ASP.NET Core Identity и промежуточного ПО .....	539
14.4.2	Обновление модели данных EF Core для поддержки Identity .....	541
14.4.3	Обновление представлений Razor для связи с пользовательским интерфейсом Identity .....	542
14.5	Настройка страницы в пользовательском интерфейсе ASP.NET Core Identity по умолчанию .....	544
14.6	Управление пользователями: добавление специальных данных для пользователей .....	547
	Резюме .....	550
<b>15</b>	<b>Авторизация: обеспечиваем защиту приложения</b> .....	553
15.1	Знакомство с авторизацией .....	555
15.2	Авторизация в ASP.NET Core .....	558
15.2.1	Предотвращение доступа анонимных пользователей к вашему приложению .....	560
15.2.2	Обработка запросов, не прошедших аутентификацию .....	562
15.3	Использование политик для авторизации на основе утверждений .....	565
15.4	Создание специальных политик авторизации .....	569
15.4.1	Требования и обработчики: строительные блоки политики .....	569
15.4.2	Создание политики со специальным требованием и обработчиком .....	571
15.5	Управление доступом с авторизацией на основе ресурсов .....	577
15.5.1	Ручная авторизация запросов с помощью интерфейса <i>IAuthorizationService</i> .....	579
15.5.2	Создание обработчика <i>AuthorizationHandler</i> на основе ресурсов .....	582
15.6	Скрытие элементов в шаблонах Razor от незарегистрированных пользователей .....	585
	Резюме .....	588



<b>16</b>	<b>Публикация и развертывание приложения</b> .....	590
16.1	Модель хостинга ASP.NET Core .....	592
16.1.1	Запуск и публикация приложения ASP.NET Core .....	594
16.1.2	Выбор метода развертывания для вашего приложения .....	598
16.2	Публикация приложения в IIS .....	600
16.2.1	Конфигурирование IIS для ASP.NET Core .....	600
16.2.2	Подготовка и публикация приложения в IIS.....	603
16.3	Размещение приложения в Linux.....	606
16.3.1	Запуск приложения ASP.NET Core за обратным прокси-сервером в Linux.....	606
16.3.2	Подготовка приложения к развертыванию в Linux .....	609
16.4	Настройка URL-адресов приложения.....	611
16.5	Оптимизация клиентских ресурсов с помощью BundlerMinifier ...	615
16.5.1	Ускорение работы приложения с помощью упаковки и минификации кода.....	618
16.5.2	Добавляем BundlerMinifier в приложение.....	620
16.5.3	Использование минифицированных файлов в промышленном окружении с помощью тег-хелпера окружения .....	623
16.5.4	Обслуживание часто используемых файлов из сети доставки содержимого.....	624
	Резюме .....	625

## Часть III РАСШИРЕНИЕ ПРИЛОЖЕНИЙ .....

628

<b>17</b>	<b>Мониторинг и устранение ошибок с помощью журналирования</b> .....	630
17.1	Эффективное использование журналирования в промышленном приложении .....	632
17.1.1	Выявление проблем с помощью специальных сообщений журнала.....	633
17.1.2	Абстракции журналирования ASP.NET Core.....	635
17.2	Добавление сообщений журнала в приложение .....	636
17.2.1	Уровень сообщения журнала: насколько важно сообщение журнала? .....	639
17.2.2	Категория сообщения журнала: какой компонент создал журнал.....	642
17.2.3	Форматирование сообщений и сбор значений параметров .....	643
17.3	Контроль места записи журналов с помощью поставщиков журналирования.....	645
17.3.1	Добавление нового поставщика журналирования в приложение ....	646
17.3.2	Замена ILoggerFactory по умолчанию на Serilog .....	649
17.4	Изменение избыточности сообщений журналов с помощью фильтрации .....	653
17.5	Структурное журналирование: создание полезных сообщений журналов с возможностью поиска.....	658
17.5.1	Добавление поставщика структурного журналирования в приложение.....	660

17.5.2	Использование областей журналирования для добавления дополнительных свойств в сообщения журнала .....	663
Резюме .....		665

<b>18</b>	<b>Повышаем безопасность приложения .....</b>	<b>667</b>
18.1	Добавляем протокол HTTPS в приложение .....	669
18.1.1	Использование HTTPS-сертификатов для разработки .....	672
18.1.2	Настройка Kestrel для использования сертификата HTTPS в промышленном окружении.....	674
18.1.3	Делаем так, чтобы протокол HTTPS использовался для всего приложения .....	676
18.2	Защита от межсайтового скриптинга .....	681
18.3	Защита от межсайтовой подделки запросов (CSRF).....	685
18.4	Вызов веб-API из других доменов с помощью CORS .....	691
18.4.1	Разбираемся с CORS и тем, как он работает.....	692
18.4.2	Добавление глобальной политики CORS ко всему приложению.....	694
18.4.3	Добавляем CORS к определенным действиям веб-API с помощью атрибута EnableCors .....	697
18.4.4	Настройка политик CORS.....	698
18.5	Изучение других векторов атак.....	699
18.5.1	Обнаружение и предотвращение атак с открытым перенаправлением.....	700
18.5.2	Предотвращение атак с использованием внедрения SQL-кода с помощью EF Core и параметризации .....	702
18.5.3	Предотвращение небезопасных прямых ссылок на объекты .....	704
18.5.4	Защита паролей и данных пользователей .....	705
Резюме .....		707

<b>19</b>	<b>Создание специальных компонентов .....</b>	<b>710</b>
19.1	Настройка конвейера промежуточного ПО .....	711
19.1.1	Создание простых конечных точек с помощью метода расширения Rip .....	713
19.1.2	Ветвление конвейера с помощью метода расширения Map .....	714
19.1.3	Добавление в конвейер с помощью метода расширения Use.....	718
19.1.4	Создание специального компонента промежуточного ПО.....	721
19.2	Создание специальных конечных точек с помощью маршрутизации конечных точек .....	724
19.2.1	Создание специального компонента маршрутизации конечных точек .....	725
19.2.2	Создание простых конечных точек с помощью MapGet и WriteJsonAsync .....	729
19.2.3	Применение авторизации к конечным точкам.....	731
19.3	Работа с требованиями к сложной конфигурации .....	733
19.3.1	Частичное создание конфигурации для настройки дополнительных поставщиков .....	734
19.3.2	Использование сервисов для настройки IOptions с помощью IConfigureOptions.....	736
19.4	Использование стороннего контейнера внедрения зависимостей .....	739
Резюме .....		743

<b>20</b>	<b>Создание специальных компонентов MVC и Razor Pages</b> .....	745
20.1	Создание специального тег-хелпера Razor .....	746
20.1.1	Вывод информации об окружении с помощью специального тег-хелпера .....	747
20.1.2	Создание специального тег-хелпера для условного скрывания элементов .....	751
20.1.3	Создание тег-хелпера для преобразования Markdown в HTML .....	753
20.2	Компоненты представления: добавление логики в частичные представления .....	755
20.3	Создание специального атрибута валидации .....	761
20.4	Замена фреймворка валидации на FluentValidation .....	766
20.4.1	Сравнение FluentValidation и атрибутов DataAnnotations .....	767
20.4.2	Добавляем FluentValidation в приложение .....	771
	Резюме .....	773
<b>21</b>	<b>Вызов удаленных API с помощью HttpClientFactory</b> .....	775
21.1	Вызов API для протокола HTTP: проблема с классом HttpClient .....	776
21.2	Создание экземпляров класса HttpClient с помощью интерфейса HttpClientFactory .....	782
21.2.1	Использование HttpClientFactory для управления жизненным циклом HttpClientHandler .....	783
21.2.2	Настройка именованных клиентов во время регистрации .....	786
21.2.3	Использование типизированных клиентов для инкапсуляции HTTP-вызовов .....	788
21.3	Обработка временных ошибок HTTP с помощью библиотеки Polly .....	791
21.4	Создание специального обработчика HttpResponseMessage .....	794
	Резюме .....	797
<b>22</b>	<b>Создание фоновых задач и сервисов</b> .....	799
22.1	Запуск фоновых задач с помощью интерфейса IHostedService .....	800
22.1.1	Запуск фоновых задач по таймеру .....	801
22.1.2	Использование сервисов с жизненным циклом Scoped в фоновых задачах .....	805
22.2	Создание сервисов рабочей роли без пользовательского интерфейса с использованием IHost .....	807
22.2.1	Создание сервиса рабочей роли из шаблона .....	809
22.2.2	Запуск сервисов рабочей роли в промышленном окружении .....	812
22.3	Координация фоновых задач с помощью Quartz.NET .....	815
22.3.1	Установка Quartz.NET в приложение ASP.NET Core .....	816
22.3.2	Настройка запуска задания по расписанию с помощью Quartz.NET .....	818

22.3.3	Использование кластеризации для добавления избыточности в фоновые задачи.....	821
	Резюме .....	825
<b>23</b>	<b>Тестирование приложения</b> .....	<b>827</b>
23.1	Тестирование в ASP.NET Core.....	829
23.2	Модульное тестирование с xUnit.....	831
23.2.1	Создание первого тестового проекта.....	831
23.2.2	Запуск тестов командой <code>dotnet test</code> .....	833
23.2.3	Ссылка на приложение из тестового проекта .....	835
23.2.4	Добавление модульных тестов с атрибутами <code>Fact</code> и <code>Theory</code> .....	838
23.2.5	Тестирование условий отказа.....	842
23.3	Модульное тестирование специального промежуточного ПО ....	843
23.4	Модульное тестирование API-контроллеров .....	846
23.5	Интеграционное тестирование: тестирование всего приложения в памяти .....	850
23.5.1	Создание <code>TestServer</code> с помощью пакета <code>Test Host</code> .....	851
23.5.2	Тестирование приложения с помощью класса <code>WebApplicationFactory</code> .....	854
23.5.3	Замена зависимостей в классе <code>WebApplicationFactory</code> .....	857
23.5.4	Уменьшение дублирования кода за счет создания специального класса <code>WebApplicationFactory</code> .....	859
23.6	Изоляция базы данных с помощью поставщика EF Core в памяти .....	861
	Резюме .....	866
	Приложение А. Подготовка окружения разработки .....	869
	Приложение В. Экосистема .NET.....	876
	Приложение С. Полезные ссылки.....	895
	Предметный указатель .....	901

# Вступительное слово от сообщества

---

.NET уже много лет является одним из лидирующих фреймворков для разработки веб-приложений. Пройдя длинный путь от ASP.NET до современного ASP.NET Core, он вобрал в себя все лучшие подходы к разработке приложений с отрисовкой на стороне сервера и веб-API-приложений. ASP.NET Core – продукт с открытым исходным кодом, каждый может изучить любой аспект его работы. Однако объем кода велик, и не так-то просто сразу понять, что искать и как разбираться с ним. Microsoft предоставляет отличную документацию по основам серверной веб-разработки и ASP.NET Core на официальном сайте, однако этого может быть недостаточно для выстраивания целостной картины.

Именно поэтому данная книга очень ценна. Автор превосходно знает ASP.NET Core, работал с ним с первых версий и как никто другой понимает, какие аспекты фреймворка наиболее важны для его успешного использования. Разработчику, помимо работы с основной логикой приложения, важно понимать, как работать с настройками, журналированием, авторизацией, как обеспечивать безопасность приложений. Все эти темы тщательно рассмотрены в книге. Автору удалось охватить широту фреймворка, рассмотрев большое количество различных аспектов, и при этом достаточно глубоко разобрать многие из них. Все это позволяет рассматривать эту книгу как отличный способ подробного знакомства с разработкой серверных приложений на .NET.

Систематизированной информации об ASP.NET Core на русском языке мало. Фреймворк быстро развивается, постоянно появляются новые термины, и даже те, что давно используются, не всегда имеют устоявшийся перевод. Мы обсуждали, спорили, думали о том, как читатели будут искать термины в сети интернет, как они звучат в неформальных беседах. Что-то получилось хорошо, что-то не очень, но в целом мы довольны результатом и рады, что такая интересная и полезная книга есть теперь и на русском языке. Отдельная благодарность автору за простые и по-

нятные примеры кода и отличные иллюстрации, наглядно демонстрирующие объясняемые концепции.

Добро пожаловать в мир ASP.NET Core, и приятного чтения!

*Российское сообщество .NET разработчиков DotNet.Ru*

The logo consists of a solid purple square. Inside the square, the text "DOT NET .RU" is written in a white, sans-serif font, stacked vertically in three lines: "DOT" on the top line, "NET" on the middle line, and ".RU" on the bottom line.

Над переводом работали представители сообщества DotNet.Ru:

- Игорь Лабутин;
- Андрей Беленцов;
- Максим Шошин;
- Вадим Мингажев;
- Сергей Бензенко;
- Радмир Тагиров;
- Эмиль Янгиров;
- Анатолий Кулаков.

# Предисловие

---

ASP.NET Core 5.0 появился в 2020 году, более чем через четыре года после выпуска ASP.NET Core 1.0, но ASP.NET также имеет долгую историю, которая послужила основой и стимулом для развития ASP.NET Core.

Microsoft выпустила первую версию ASP.NET в 2002 году как часть платформы .NET Framework 1.0. С тех пор она прошла несколько выпусков, в каждом из которых были добавлены функции и расширяемость. Однако каждый выпуск был построен на основе .NET Framework, поэтому она предустановлена во всех версиях Windows.

Это приносит смешанные преимущества: с одной стороны, сегодня ASP.NET 4.x является надежной, проверенной в боях платформой для создания современных приложений для ОС Windows. С другой стороны, она ограничена этой зависимостью – изменения в базовой платформе .NET Framework имеют далеко идущие последствия, в результате чего наблюдается замедление скорости развертывания, а это оставляет за бортом многих разработчиков, создающих и развертывающих приложения для Linux или macOS.

Когда я впервые начал изучать ASP.NET Core, я был одним из таких разработчиков. Будучи в душе пользователем Windows, я получил от своего работодателя компьютер с macOS и поэтому все время работал на виртуальной машине. ASP.NET Core обещал все это изменить, позволив вести разработку и на компьютере с Windows, и на компьютере с macOS.

Можно сказать, что я опоздал во многих отношениях, проявляя активный интерес только перед выходом релиза-кандидата ASP.NET Core RC2. К тому моменту существовало уже восемь бета-версий, многие из которых содержали существенные критические изменения. Не погружаясь во все это полностью до выхода RC2, я избавился от сырых инструментов и меняющихся API.

То, что я увидел в тот момент, меня очень впечатлило. ASP.NET Core позволяет разработчикам использовать имеющиеся у них знания о платформе .NET и приложениях ASP.NET MVC, в частности используя текущие передовые практики, такие как внедрение зависимостей, строго типизированная конфигурация и журналирование. Кроме того, мож-

но было создавать и развертывать кросс-платформенные приложения. Я не устоял.

Эта книга появилась во многом благодаря моему подходу к изучению ASP.NET Core. Вместо того чтобы просто читать документацию и статьи в блогах, я решил попробовать что-то новое и начать писать о том, что я узнал. Каждую неделю я посвящал время изучению нового аспекта ASP.NET Core и писал об этом сообщение в блоге. Когда появилась возможность написать книгу, я ухватился за этот шанс – это еще один повод подробно изучить фреймворк!

С тех пор, как я начал писать эту книгу, многое изменилось как в отношении самой книги, так и ASP.NET Core. Первый крупный выпуск фреймворка в июне 2016 года по-прежнему имел много шероховатостей, в частности что касалось работы с инструментами. С выпуском .NET 5.0 в ноябре 2020 года ASP.NET Core действительно стал самостоятельным: API и инструменты достигли зрелого уровня. Данная книга нацелена на версию .NET 5.0 для ASP.NET Core, но если вы используете хотя бы версию .NET Core 3.1, то сможете без проблем работать с этим изданием.

В этой книге рассказывается обо всем, что вам нужно для начала работы с ASP.NET Core, независимо от того, новичок ли вы в веб-разработке или уже являетесь разработчиком ASP.NET. В ней очень много внимания уделяется самому фреймворку, поэтому я не буду вдаваться в подробности, касающиеся клиентских фреймворков, таких как Angular и React, или таких технологий, как Docker. Я также не описываю все новые функции .NET 5.0, такие как Blazor и gRPC. Вместо этого я даю ссылки, по которым вы можете найти дополнительную информацию.

Мы сосредоточимся на создании приложений с отрисовкой на стороне сервера, используя страницы Razor и веб-API, применяя контроллеры MVC. Вы познакомитесь с основами ASP.NET Core, такими как промежуточное ПО, внедрение зависимостей и конфигурация, а также узнаете, как настроить их в соответствии со своими требованиями. Вы узнаете, как добавить аутентификацию и авторизацию в свои приложения, как повысить их безопасность, а также как развертывать их и осуществлять мониторинг. Наконец, вы узнаете, как тестировать приложения, используя модульные и интеграционные тесты.

Лично мне приятно работать с приложениями ASP.NET Core по сравнению с приложениями, использующими предыдущую версию ASP.NET, и надеюсь, что эта страсть проявится в данной книге!



# Благодарности

---

Хотя на обложке этой книги только одно имя, множество людей внесли свой вклад как в ее написание, так и в публикацию. В этом разделе я хотел бы поблагодарить всех, кто поддерживал меня, оказывал содействие и терпел меня в течение прошлого года.

Прежде всего я хочу поблагодарить свою девушку Бекки. Твоя постоянная поддержка и воодушевление – все для меня. Они помогли мне пережить этот напряженный период. Ты приняла на себя всю тяжесть этого стресса и давления, и я бесконечно благодарен тебе. Безмерно люблю тебя.

Я также хотел бы поблагодарить всю свою семью за их поддержку. В частности, моих родителей, Жен и Боба, за то, что терпели мои разглагольствования, и свою сестру, Аманду, за все ее веселые беседы.

На профессиональном уровне я хотел бы поблагодарить издательство Manning за предоставленную мне возможность. Брайан Соьер «нашел» меня в первом издании этой книги и побудил меня заняться вторым изданием. Марина Майклс стала моим редактором-консультантом по аудитории второй раз подряд и снова была то дотошной, то критически настроенной, то обнадеживающей и восторженной. Книга, несомненно, стала лучше благодаря вашему участию. Я также благодарю редактора проекта Дейдру Хиям, редактора Энди Кэрролла, своего корректора Джейсона Эверетта и редактора-рецензента Михаэла Батиника.

Я благодарен техническому редактору Марку Элстону и корректору Тане Уилке. Марк оказал неоценимую поддержку, подчеркнув мои неверные предположения и технические предубеждения, касающиеся работы с хорошо знакомым мне фреймворком. Таня Уилке подтвердила, что написанный мной код действительно работает и что он не лишен смысла.

Я сердечно благодарю всех сотрудников Manning, которые помогли издать эту книгу и вывести ее на рынок. Я также хотел бы поблагодарить всех рецензентов MEAP за их комментарии, которые помогли улучшить книгу.

Я бы никогда не смог написать ее, если бы не отличный контент, созданный сообществом .NET и теми пользователями, на которых я подпи-

сан в Twitter. В частности, спасибо Джону Гэллоуэю за регулярное размещение моего блога на форуме сообщества ASP.NET.

Наконец, спасибо всем друзьям, которые воодушевляли и поддерживали меня и в целом проявляли интерес. Возможно, нам не удалось встречаться настолько часто, насколько нам хотелось бы, но я с нетерпением жду возможности как можно скорее собраться вместе и выпить.

Благодарю всех рецензентов: Эла Пезевски (Al Pezewski), Бена Макнамару (Ben McNamara), Даниэля Васкеса (Daniel Vásquez), Филипа Войчешина (Filip Wojcieszyn), Фостера Хейнса (Foster Haines), Густаво Филипе Рамоса Гомеса (Gustavo Filipe Ramos Gomes), Жана-Франсуа Морена (Jean-François Morin), Джоэля Котарски (Joel Kotarski), Джона Гатри (John Guthrie), Хуана Луиса Барреду (Juan Luis Barreda), Луиса Му (Luis Moux), Майка Эриксона (Mike Erickson), Раушана Джа (Raushan Jha), Роба Ройтча (Rob Ruetsch), Рона Лиза (Ron Lease), Рубена Вандегинсте (Ruben Vandeginste), Сау Фай Фонг (Sau Fai Fong), Стива Лава (Steve Love), Таню Уилке (Tanya Wilke), Винсента Делкойна (Vincent Delcoigne) и Уиллиса Г. Хэмптона (Willis G. Hampton) – ваши предложения помогли сделать эту книгу лучше.

# Об этой книге

---

Данная книга посвящена фреймворку ASP.NET Core: в ней рассказывается о том, что это такое и как использовать его для создания веб-приложений. Хотя часть этой информации уже доступна в интернете, она разбросана по сети в виде разрозненных документов и сообщений в блогах. Эта книга показывает, как создать свое первое приложение, наращивая сложность по мере того, как вы будете закреплять предыдущие концепции.

Я представляю каждую тему на относительно небольших примерах, вместо того чтобы создавать одно-единственное приложение на протяжении всей книги. У обоих подходов есть свои достоинства, но я хотел убедиться, что основное внимание уделяется конкретным изучаемым темам, без умственных затрат на навигацию по растущему проекту.

К концу книги вы должны иметь твердое представление о том, как создавать приложения с помощью ASP.NET Core, знать сильные и слабые стороны фреймворка и как использовать его функции для безопасного создания приложений. Хотя я не трачу много времени на архитектуру приложений, я непременно привожу передовые практики, особенно там, где лишь поверхностно рассказываю об архитектуре для краткости.

## *Кому адресована эта книга*

Данная книга рассчитана на разработчиков на языке C#, которые заинтересованы в изучении кросс-платформенного веб-фреймворка. Она не предполагает, что у вас есть какой-либо опыт создания веб-приложений, например вы можете разрабатывать приложения для мобильных устройств или ПК, хотя предыдущий опыт работы с ASP.NET или другим веб-фреймворком, несомненно, полезен.

Помимо практических знаний C# и .NET, предполагается наличие знания общих объектно-ориентированных практик и базового понимания реляционных баз данных в общем. Я предполагаю, что вы немного знакомы с HTML и CSS, а также с тем, что JavaScript является языком сценариев на стороне клиента. Вам не нужно знать JavaScript- или CSS-фреймворки

для работы с этой книгой, хотя ASP.NET Core хорошо работает с ними, если это ваша сильная сторона.

Веб-фреймворки естественным образом затрагивают широкий круг тем, начиная с базы данных и сети и заканчивая визуальным дизайном и написанием скриптов на стороне клиента. Я предоставляю как можно больше контекста и включаю ссылки на сайты и книги, где можно получить более подробную информацию.

## **Как организована эта книга: дорожная карта**

Эта книга состоит из трех частей, 23 глав и трех приложений. В идеале вы должны прочитать ее от корки до корки, а затем использовать ее в качестве справочника, но я понимаю, что такой вариант подойдет не всем. Хотя я использую небольшие примеры приложений для демонстрации той или иной темы, некоторые главы основаны на предыдущих, поэтому содержание книги будет иметь больше смысла, если вы будете читать главы последовательно.

Я настоятельно рекомендую читать главы первой части последовательно, поскольку каждая глава основывается на темах, представленных в предыдущих главах. Вторую часть также лучше читать последовательно, хотя большинство глав независимы, если вы хотите перескакивать от одной темы к другой. Главы в третьей части можно читать в произвольном порядке, хотя я рекомендую делать это только после того, вы прошли первую и вторую части.

Первая часть представляет собой общее введение в ASP.NET Core и дает общую архитектуру типичного веб-приложения. Изучив основы, мы переходим к фреймворку Razor Pages, который составляет основную часть веб-страниц, веб-приложений ASP.NET Core с отрисовкой на стороне сервера и базовой архитектуры *Модель–представление–контроллер* (MVC):

- глава 1 знакомит вас с ASP.NET Core и его местом в среде веб-разработки. В ней обсуждается, когда следует и когда не следует использовать ASP.NET Core, основы веб-запросов в ASP.NET Core и варианты, доступные для окружения разработки;
- в главе 2 рассматриваются все компоненты базового приложения ASP.NET Core, обсуждаются их роли и то, как они сочетаются для генерации ответа на веб-запрос;
- в главе 3 описывается конвейер промежуточного ПО, который является основным конвейером приложения в ASP.NET Core. Он определяет, как обрабатываются входящие запросы и как должен генерироваться ответ;
- в главе 4 показано, как использовать Razor Pages для создания страничных веб-сайтов. Razor Pages – это рекомендуемый способ создания приложений с отрисовкой на стороне сервера в ASP.NET Core, предназначенный для страничных приложений;
- в главе 5 описана система маршрутизации Razor Pages. Маршрутизация – это процесс сопоставления URL-адресов входящих запросов

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

[e-Univers.ru](http://e-Univers.ru)