

Краткое оглавление

1	■	<i>Начало работы с ASP.NET Core</i>	34
ЧАСТЬ I. Начало работы с минимальными API			45
2	■	<i>Что такое ASP.NET Core</i>	47
3	■	<i>Наше первое приложение</i>	67
4	■	<i>Обработка ошибок с помощью конвейера промежуточного ПО</i>	94
5	■	<i>Создание JSON API с помощью минимальных API</i>	126
6	■	<i>Сопоставление URL-адресов с конечными точками с помощью маршрутизации</i>	163
7	■	<i>Привязка модели и валидация в минимальных API</i>	187
ЧАСТЬ II. Создание полноценных приложений			221
8	■	<i>Введение во внедрение зависимостей</i>	223
9	■	<i>Регистрация сервисов с помощью внедрения зависимостей</i>	242
10	■	<i>Конфигурирование приложения ASP. Net Core</i>	269
11	■	<i>Документирование API с помощью OpenAPI</i>	306
12	■	<i>Сохранение данных с Entity Framework Core</i>	340
ЧАСТЬ III. Генерация HTML-кода с помощью Razor Pages и MVC			377
13	■	<i>Создание сайта с помощью страниц Razor</i>	379
14	■	<i>Сопоставление URL-адресов с Razor Pages с использованием маршрутизации</i>	405
15	■	<i>Создание ответов с помощью обработчиков страниц в Razor Pages</i>	427
16	■	<i>Привязка и валидация запросов с помощью Razor Pages</i>	445
17	■	<i>Отрисовка HTML-кода с использованием представлений Razor</i>	476
18	■	<i>Создание форм с помощью тег-хелперов</i>	508
19	■	<i>Создание сайта с использованием контроллеров MVC</i>	541

20	■	<i>Создание HTTP API с использованием контроллеров веб-API</i>	562
21	■	<i>Конвейер фильтров MVC и Razor Pages.....</i>	595
22	■	<i>Создание собственных фильтров MVC и страниц Razor</i>	612
ЧАСТЬ IV. Защита и развертывание приложений.....			639
23	■	<i>Аутентификация: добавление пользователей в приложение с помощью ASP.NET Core Identity</i>	641
24	■	<i>Авторизация: обеспечиваем защиту приложения.....</i>	677
25	■	<i>Аутентификация и авторизация для API.....</i>	713
26	■	<i>Мониторинг и устранение ошибок с помощью журналирования</i>	747
27	■	<i>Публикация и развертывание приложения</i>	780
28	■	<i>Добавляем протокол HTTPS в приложение.....</i>	805
29	■	<i>Повышаем безопасность приложения</i>	826
ЧАСТЬ V. Дальнейшая работа с ASP.NET Core			855
30	■	<i>Создание приложений ASP.NET Core с помощью универсального узла и класса Startup</i>	857
31	■	<i>Расширенная настройка ASP.NET Core</i>	877
32	■	<i>Создание специальных компонентов MVC и Razor Pages</i>	907
33	■	<i>Вызов удаленных API с помощью IHttpClientFactory</i>	937
34	■	<i>Создание фоновых задач и сервисов</i>	960
35	■	<i>Тестирование приложений с xUnit</i>	989
36	■	<i>Тестирование приложений ASP.NET Core</i>	1006

Оглавление

Предисловие от издательства	21
Вступительное слово от сообщества DotNet.Ru.....	22
Предисловие	24
Благодарности	26
Об этой книге	28
Об авторе	33
Об иллюстрации на обложке.....	33
1 Начало работы с ASP.NET Core	34
1.1 Что такое ASP.NET Core?.....	34
1.2 Какие типы приложений можно создавать?	35
1.3 Выбор ASP.NET Core	36
1.4 Как работает ASP.NET Core?	38
1.4.1 Как работает веб-запрос по протоколу HTTP?	38
1.4.2 Как ASP.NET Core обрабатывает запрос?	41
1.5 Что вы узнаете из этой книги	42
Резюме	43
ЧАСТЬ I. Начало работы с минимальными API	45
2 Что такое ASP.NET Core.....	47
2.1 Использование фреймворка для создания веб-приложений.....	47
2.2 Для чего был создан ASP.NET Core.....	48
2.3 Парадигмы ASP.NET Core	53
2.4 Когда следует выбрать ASP.NET Core.....	56
2.4.1 Если вы новичок в разработке .NET	57
2.4.2 Если вы .NET-разработчик, создающий новое приложение.....	59
2.4.3 Перенос существующего ASP.NET-приложения на ASP.NET Core.....	64
Резюме	66
3 Наше первое приложение.....	67
3.1 Краткий обзор приложения ASP.NET Core.....	68
3.2 Создаем наше первое приложение ASP.NET Core	70
3.2.1 Использование шаблона	72
3.2.2 Сборка приложения.....	75
3.3 Запуск веб-приложения	76
3.4 Разбираемся с шаблоном проекта	77
3.5 Файл проекта .csproj: объявление зависимостей.....	79
3.6 Файл Program.cs: определение приложения	81
3.7 Добавляем функциональность в приложение	84
3.7.1 Добавление и настройка сервисов	87
3.7.2 Определение способа обработки запросов с помощью промежуточного ПО и конечных точек.....	89
Резюме	92

4	<i>Обработка ошибок с помощью конвейера промежуточного ПО ...</i>	94
4.1	Что такое промежуточное ПО	96
4.2	Объединение компонентов в конвейер	100
4.2.1	Простой сценарий конвейера 1: страница приветствия	101
4.2.2	Простой сценарий конвейера 2: обработка статических файлов	104
4.2.3	Простой сценарий конвейера 3: приложение с минимальным API	108
4.3	Обработка ошибок с помощью промежуточного ПО	115
4.3.1	Просмотр исключений в окружении разработки: DeveloperExceptionPage	117
4.3.2	Обработка исключений в промышленном окружении: ExceptionHandlerMiddleware	119
	Резюме	124
5	<i>Создание JSON API с помощью минимальных API</i>	126
5.1	Что такое HTTP API и когда его следует использовать?	127
5.2	Определение конечных точек минимальных API	131
5.2.1	Извлечение значений из URL-адреса с помощью маршрутизации	132
5.2.2	Сопоставление HTTP-методов с конечными точками	133
5.2.3	Определение обработчиков маршрутов с помощью функций ...	135
5.3	Генерация ответов с помощью IActionResult	138
5.3.1	Возврат кодов состояния с помощью Results и TypedResults	139
5.3.2	Возврат полезных данных об ошибках с помощью Problem Details	141
5.3.3	Преобразование всех ответов в Problem Details	143
	Преобразование исключений в Problem Details	144
	Преобразование кодов состояния ошибок в формат Problem Details	146
5.4	Запуск общего кода с фильтрами конечных точек	148
5.4.1	Добавление нескольких фильтров к конечной точке	151
5.4.2	Фильтры или промежуточное ПО: что выбрать?	153
5.4.3	Обобщенные фильтры конечных точек	154
5.4.4	Реализация интерфейса IEndpointFilter	157
5.5	Организация API с помощью групп маршрутов	158
	Резюме	161
6	<i>Сопоставление URL-адресов с конечными точками с помощью маршрутизации</i>	163
6.1	Что такое маршрутизация?	164
6.2	Маршрутизация конечных точек в ASP.NET Core	168
6.3	Изучение синтаксиса шаблона маршрута	172
6.3.1	Работа с параметрами и литеральными сегментами	172
6.3.2	Использование необязательных значений и значений по умолчанию	174
6.3.3	Добавление дополнительных ограничений к параметрам маршрута	175

6.3.4 Сопоставление произвольных URL-адресов с помощью универсального параметра	178
6.4 Генерация URL-адресов из параметров маршрута	179
6.4.1 Генерация URL-адресов для конечной точки минимального API с помощью класса LinkGenerator	180
6.4.2 Генерация URL-адресов с помощью IResults	182
6.4.3 Управление созданными URL-адресами с помощью RouteOptions	183
Резюме	185
7 Привязка модели и валидация в минимальных API	187
7.1 Извлечение значений из запроса с привязкой модели	188
7.2 Привязка простых типов к запросу	190
7.3 Привязка сложных типов к телу запроса в формате JSON	195
7.4 Массивы: простые типы или сложные?	197
7.5 Делаем параметры необязательными с помощью типов, допускающих значение NULL	200
7.6 Привязка сервисов и специальных типов	203
7.6.1 Внедрение хорошо известных типов	203
7.6.2 Внедрение сервисов	204
7.6.3 Привязка загрузки файлов с помощью IFormFile и IFormFileCollection	205
7.7 Собственная привязка с помощью BindAsync	207
7.8 Выбор источника привязки	208
7.9 Упрощение обработчиков с помощью AsParameters	209
7.10 Обработка пользовательского ввода с помощью валидации модели	211
7.10.1 Необходимость валидации модели	211
7.10.2 Использование атрибутов DataAnnotations для валидации	212
7.10.3 Добавление фильтра валидации в минимальные API	215
Резюме	218
ЧАСТЬ II. Создание полноценных приложений	221
8 Введение во внедрение зависимостей	223
8.1 Преимущества внедрения зависимостей	224
8.2 Создание слабосвязанного кода	231
8.3 Использование внедрения зависимостей в ASP.NET Core	233
8.4 Добавление сервисов ASP.NET Core в контейнер	235
8.5 Использование сервисов из контейнера внедрения зависимостей	238
Резюме	240
9 Регистрация сервисов с помощью внедрения зависимостей	242
9.1 Регистрация собственных сервисов в контейнере внедрения зависимостей	243
9.2 Регистрация сервисов с использованием объектов и лямбда-функций	247
9.3 Многократная регистрация сервиса в контейнере	251
9.3.1 Внедрение нескольких реализаций интерфейса	252

9.3.2 Внедрение одной реализации при регистрации нескольких сервисов	254
9.3.3 Условная регистрация сервисов с помощью TryAdd	254
9.4 Жизненный цикл: когда создаются сервисы?	255
9.4.1 Transient: уникален каждый	258
9.4.2 Scoped: держимся вместе	259
9.4.3 Singleton: может быть только один	260
9.4.4 Следите за захваченными зависимостями	262
9.5 Разрешение сервисов с жизненным циклом scoped за пределами запроса	265
Резюме	267

10 *Конфигурирование приложения ASP. Net Core* 269

10.1 Представляем модель конфигурации ASP.NET Core	270
10.2 Создание объекта конфигурации для приложения	272
10.2.1 Добавление поставщика конфигурации в файле Program.cs	275
10.2.2 Использование нескольких поставщиков для переопределения значений конфигурации	278
10.2.3 Безопасное хранение секретов конфигурации	280
Сохранение секретов в переменных окружения в промышленном окружении	281
Хранение секретов с помощью менеджера User Secrets в окружении разработки	282
10.2.4 Перегрузка значений конфигурации при их изменении	284
10.3 Использование строго типизированных настроек с паттерном «Параметры»	286
10.3.1 Знакомство с интерфейсом IOptions	288
10.3.2 Перегрузка строго типизированных параметров с помощью IOptionsSnapshot	289
10.3.3 Разработка классов параметров для автоматической привязки	291
10.3.4 Привязка строго типизированных параметров без интерфейса IOptions	293
10.4 Настройка приложения для нескольких окружений	295
10.4.1 Определение окружения размещения	296
10.4.2 Загрузка файлов конфигурации для конкретного окружения ...	297
10.4.3 Задаем окружение размещения	299
Резюме	303

11 *Документирование API с помощью OpenAPI* 306

11.1 Добавление описания OpenAPI в приложение	307
11.2 Тестирование API с помощью Swagger UI	310
11.3 Добавление метаданных в минимальные API	312
11.4 Создание строго типизированных клиентов с помощью NSwag	315
11.4.1 Генерация клиента с помощью Visual Studio	316
11.4.2 Создание клиента с помощью инструмента .NET Global	320
11.4.3 Использование сгенерированного клиента для вызова API	322
11.4.4 Настройка сгенерированного кода	323
11.4.5 Обновление описания OpenAPI	326

11.5 Добавление описаний и сводок в конечные точки	327
11.5.1 Использование текущих методов для добавления описаний.....	328
11.5.2 Использование атрибутов для добавления метаданных.....	330
11.6 Ограничения OpenAPI.....	334
11.6.1 Не все API можно описать, используя OpenAPI	334
11.6.2 Сгенерированный код чрезмерно категоричен	335
11.6.3 Инструменты часто отстают от спецификации.....	336
Резюме	337

12 Сохранение данных с Entity Framework Core 340

12.1 Знакомство с Entity Framework Core	342
12.1.1 Что такое EF Core?	342
12.1.2 Зачем использовать инструмент объектно-реляционного отображения?.....	344
12.1.3 Когда следует выбирать EF Core?	345
12.1.4 Отображение базы данных в код приложения	346
12.2 Добавляем EF Core в приложение	348
12.2.1 Выбор провайдера базы данных и установка EF Core.....	350
12.2.2 Создание модели данных	351
12.2.3 Регистрация контекста данных	354
12.3 Управление изменениями с помощью миграций.....	355
12.3.1 Создаем первую миграцию	356
12.3.2 Добавляем вторую миграцию	359
12.4 Выполнение запроса к базе данных и сохранение в ней данных	362
12.4.1 Создание записи	363
12.4.2 Загрузка списка записей.....	365
12.4.3 Загрузка отдельной записи	367
12.4.4 Обновление модели с изменениями	369
12.5 Использование EF Core в промышленных приложениях	373
Резюме	375

ЧАСТЬ III. Генерация HTML-кода с помощью Razor Pages и MVC377

13 Создание сайта с помощью Razor Pages 379

13.1 Наше первое приложение Razor Pages	380
13.1.1 Использование шаблона «Веб-приложение»	380
13.1.2 Добавление и настройка сервисов	384
13.1.3 Создание HTML с помощью Razor Pages.....	386
13.1.4 Логика обработки запросов с помощью моделей страницы и обработчиков.....	388
13.2 Изучение типичной страницы Razor	390
13.3 Паттерн проектирования MVC.....	393
13.4 Применение паттерна проектирования MVC к Razor Pages.....	395
13.4.1 Направление запроса на страницу Razor и создание модели привязки	397
13.4.2 Выполнение обработчика с использованием модели приложения	399
13.4.3 Генерация HTML-кода с использованием модели представления	400
13.4.4 Собираем все вместе: полный запрос страницы Razor.....	401
Резюме	403

14	<i>Сопоставление URL-адресов с Razor Pages с использованием маршрутизации</i>	405
14.1	Маршрутизация в ASP.NET Core	406
14.2	Маршрутизация на основе соглашений и явная маршрутизация	407
14.3	Маршрутизация запросов в Razor Pages	410
14.4	Настройка шаблонов маршрутов страницы Razor	412
14.4.1	Добавление сегмента в шаблон маршрута	413
14.4.2	Полная замена шаблона маршрута	414
14.5	Генерация URL-адресов для страниц Razor	415
14.5.1	Генерация URL-адресов для страницы Razor	415
14.5.2	Генерация URL-адресов для контроллера MVC	416
14.5.3	Генерация URL-адресов с помощью класса LinkGenerator	418
14.6	Настройка соглашений с помощью Razor Pages	419
	Резюме	425
15	<i>Создание ответов с помощью обработчиков страниц в Razor Pages</i>	427
15.1	Razor Pages и обработчики страниц	428
15.2	Выбор обработчика страницы для вызова	429
15.3	Прием параметров в обработчиках страниц	432
15.4	Возврат ответов IActionResult	434
15.4.1	PageResult и RedirectToPageResult	436
15.4.2	NotFoundResult и StatusCodeResult	436
15.5	Обработка кодов состояния с помощью StatusCodePagesMiddleware	438
	Резюме	443
16	<i>Привязка и валидация запросов с помощью Razor Pages</i>	445
16.1	Модели в Razor Pages и MVC	446
16.2	От запроса к модели: делаем запрос полезным	449
16.2.1	Связывание простых типов	454
16.2.2	Связывание сложных типов	457
	Упрощение параметров метода привязкой к сложным объектам	458
	Привязка коллекций и словарей	459
	Привязка загрузки файлов с помощью IFormFile	461
16.2.3	Выбор источника привязки	461
16.3	Валидация моделей привязки	464
16.3.1	Валидация в Razor Pages	464
16.3.2	Валидация на сервере в целях безопасности	466
16.3.3	Валидация на стороне клиента для улучшения пользовательского опыта	470
16.4	Организация моделей привязки в Razor Pages	471
	Резюме	474
17	<i>Отрисовка HTML-кода с использованием представлений Razor</i>	476
17.1	Представления: отрисовка пользовательского интерфейса	478

17.2	Создание представлений Razor.....	482
17.2.1	Представления Razor и сопутствующий код	482
17.2.2	Знакомство с шаблонами Razor.....	483
17.2.3	Передача данных в представления	485
17.3	Создание динамических веб-страниц с помощью Razor	488
17.3.1	Использование кода C# в шаблонах Razor	489
17.3.2	Добавление циклов и условий в шаблоны Razor	490
17.3.3	Отрисовка HTML-кода с помощью метода Raw	493
17.4	Макеты, частичные представления и _ViewStart	495
17.4.1	Использование макетов для общей разметки	496
17.4.2	Переопределение родительских макетов с помощью секций....	499
17.4.3	Использование частичных представлений для инкапсуляции разметки	501
17.4.4	Выполнение кода в каждом представлении с помощью _ViewStart и _ViewImports.....	504
	Импорт общих директив с помощью _ViewImports.....	504
	Выполнение кода для каждого представления с помощью _ViewStart	505
	Резюме	506

18 Создание форм с помощью тег-хелперов..... 508

18.1	Редакторы кода и тег-хелперы	510
18.2	Создание форм с помощью тег-хелперов	513
18.2.1	Тег-хелпер формы	518
18.2.2	Тег-хелпер метки	519
18.2.3	Тег-хелперы текстового ввода.....	521
18.2.4	Тег-хелпер раскрывающегося списка.....	525
18.2.5	Тег-хелперы сообщений валидации и сводок валидации	531
18.3	Создание ссылок с помощью тег-хелпера привязки ссылки	534
18.4	Сброс кеша с помощью тег-хелпера добавления версии	535
18.5	Использование условной разметки с помощью тег-хелпера окружения.....	537
	Резюме	538

19 Создание сайта с использованием контроллеров MVC 541

19.1	Razor Pages и MVC в ASP.NET Core	542
19.2	Наше первое веб-приложение MVC	543
19.3	Сравнение контроллера MVC с PageModel из Razor Page	547
19.4	Выбор представления из контроллера MVC	549
19.5	Выбор между Razor Pages и контроллерами MVC.....	556
19.5.1	Преимущества Razor Page.....	556
19.5.2	Когда выбирать контроллеры MVC вместо Razor Pages	559
	Резюме	561

20 Создание HTTP API с использованием контроллеров веб-API 562

20.1	Создание первого проекта веб-API.....	563
20.2	Применение паттерна проектирования MVC к веб-API.....	570
20.3	Маршрутизация на основе атрибутов: связывание методов действий с URL-адресами	574

20.3.1 Сочетание атрибутов маршрута для следования принципу DRY	576
20.3.2 Использование замены маркера для уменьшения дублирования при маршрутизации на основе атрибутов.....	578
20.3.3 Обработка HTTP-методов с помощью маршрутизации на основе атрибутов.....	579
20.4 Использование общепринятых соглашений с атрибутом [ApiController]	580
20.5 Генерация ответа от модели	583
20.5.1 Настройка форматов по умолчанию: добавляем поддержку XML.....	586
20.5.2 Выбор формата ответа с помощью согласования типа содержимого	587
20.6 Контроллеры веб-API и минимальные API: что выбрать?.....	589
Резюме	592

21 Конвейер фильтров MVC и Razor Pages 595

21.1 Что такое конвейер фильтров MVC.....	596
21.2 Конвейер фильтров Razor Pages	599
21.3 Фильтры или промежуточное ПО: что выбрать?	601
21.4 Создание простого фильтра	602
21.5 Добавляем фильтры к действиям и страницам Razor Pages	605
21.6 Порядок выполнения фильтров	608
21.6.1 Порядок выполнения фильтров по умолчанию.....	608
21.6.2 Переопределение порядка выполнения фильтров по умолчанию с помощью интерфейса IOrderedFilter	609
Резюме	610

22 Создание собственных фильтров MVC и страниц Razor 612

22.1 Создание собственных фильтров для приложения.....	613
22.1.1 Фильтры авторизации: защита API	616
22.1.2 Фильтры ресурсов: прерывание выполнения методов действий.....	617
22.1.3 Фильтры действий: настройка привязки модели и результатов действий	619
22.1.4 Фильтры исключений: собственная обработка исключений для методов действий	625
22.1.5 Фильтры результатов: настройка результатов действий перед их выполнением	627
22.1.6 Фильтры страниц: настройка привязки модели для Razor Pages.....	629
22.2 Прерывание выполнения конвейера.....	631
22.3 Использование внедрения зависимостей с атрибутами фильтра	634
Резюме	637

ЧАСТЬ IV. Защита и развертывание приложений639

23 Аутентификация: добавление пользователей в приложение с помощью ASP.NET Core Identity..... 641

23.1 Знакомство с аутентификацией и авторизацией.....	643
23.1.1 Пользователи и утверждения в ASP.NET Core.....	643
23.1.2 Аутентификация в ASP.NET Core: сервисы и промежуточное ПО	644
Вход в приложение ASP.NET Core	645
Аутентификация пользователей для последующих запросов	646
23.2 Что такое ASP.NET Core Identity?	649
23.3 Создание проекта, в котором используется ASP.NET Core Identity....	652
23.3.1 Создание проекта из шаблона.....	652
23.3.2 Изучение шаблона в Обзорщике решений	654
23.3.3 Модель данных ASP.NET Core Identity	658
23.3.4 Взаимодействие с ASP.NET Core Identity	660
23.4 Добавляем ASP.NET Core Identity в существующий проект.....	663
23.4.1 Настройка сервисов ASP.NET Core Identity и промежуточного ПО	664
23.4.2 Обновление модели данных EF Core для поддержки Identity ...	665
23.4.3 Обновление представлений Razor для связи с пользовательским интерфейсом Identity	666
23.5 Настройка страницы по умолчанию в пользовательском интерфейсе ASP.NET Core Identity.....	668
23.6 Управление пользователями: добавление специальных данных для пользователей	672
Резюме	674

24 Авторизация: обеспечиваем защиту приложения 677

24.1 Знакомство с авторизацией.....	679
24.2 Авторизация в ASP.NET Core	682
24.2.1 Предотвращение доступа анонимных пользователей к приложению.....	683
24.2.2 Обработка запросов, не прошедших аутентификацию	685
24.3 Использование политик для авторизации на основе утверждений... 689	
24.4 Создание пользовательских политик авторизации.....	692
24.4.1 Требования и обработчики: строительные блоки политики	693
24.4.2 Создание политики со специальным требованием и обработчиком.....	694
Создание IAuthorizationRequirement для представления требования	694
Создание политики с несколькими требованиями	695
Создаем обработчики авторизации, чтобы отвечать требованиям.....	696
24.5 Управление доступом с авторизацией на основе ресурсов	700
24.5.1 Ручная авторизация запросов с помощью IAuthorizationService	702
24.5.2 Создание обработчика AuthorizationHandler на основе ресурсов.....	705
24.6 Скрытие HTML-элементов от неавторизованных пользователей	708
Резюме	711

25 Аутентификация и авторизация для API 713

25.1 Аутентификация для API и распределенных приложений	714
25.1.1 Распространение аутентификации на несколько приложений	714

25.1.2 Централизация аутентификации в поставщике идентификационной информации	716
25.1.3 OpenID Connect и OAuth 2.0	719
25.2 Аутентификация с токеном на предъявителя	722
25.3 Добавление аутентификации на основе JWT-токенов на предъявителя в минимальные API	729
25.4 Использование инструмента user-jwts для локального тестирования JWT-токенов	731
25.4.1 Создание JWT-токенов с помощью инструмента user-jwts.....	732
25.4.2 Настройка JWT-токенов.....	734
25.4.3 Управление локальными JWT	735
25.5 Описание требований к аутентификации для OpenAPI	736
25.6 Применение политик авторизации к конечным точкам минимальных API	740
Резюме	744

26 *Мониторинг и устранение ошибок с помощью журналирования* 747

26.1 Эффективное использование журналирования в промышленном приложении.....	749
26.1.1 Выявление проблем с помощью специальных сообщений журнала	750
26.1.2 Абстракции журналирования ASP.NET Core	752
26.2 Добавление сообщений журнала в ваше приложение	753
26.2.1 Уровень сообщения журнала: насколько важно сообщение журнала?	757
26.2.2 Категория журнала: какой компонент создал журнал?	759
26.2.3 Форматирование сообщений и сбор значений параметров... ..	760
26.3 Контроль места записи журналов с помощью поставщиков журналирования	762
26.4 Изменение избыточности сообщений журналов с помощью фильтрации	766
26.5 Структурное журналирование: создание полезных сообщений журналов с возможностью поиска	771
26.5.1 Добавление поставщика структурного журналирования в приложение	773
26.5.2 Использование областей журналирования для добавления дополнительных свойств в сообщения журнала.....	776
Резюме	778

27 *Публикация и развертывание приложения* 780

27.1 Что такое модель хостинга ASP.NET Core.....	781
27.1.1 Запуск и публикация приложения ASP.NET Core.....	784
27.1.2 Выбор метода развертывания для приложения.....	788
27.2 Публикация приложения в IIS	790
27.2.1 Конфигурирование IIS для ASP.NET Core	790
27.2.2 Подготовка и публикация приложения в IIS	792
27.3 Размещение приложения в Linux	795

27.3.1 Запуск приложения ASP.NET Core за обратным прокси-сервером в Linux	795
27.3.2 Подготовка приложения к развертыванию в Linux	798
27.4 Настройка URL-адресов приложения	800
Резюме	804

28 Добавляем протокол HTTPS в приложение 805

28.1 Для чего нужен протокол HTTPS?	806
28.2 Использование HTTPS-сертификатов для разработки	810
28.3 Настройка Kestrel для использования сертификата HTTPS в промышленном окружении	813
28.4 Делаем так, чтобы протокол HTTPS использовался для всего приложения	815
28.4.1 Принудительное использование протокола HTTPS с помощью заголовков HTTP Strict Transport Security	816
28.4.2 Переадресация с HTTP на HTTPS с помощью компонента HttpsRedirectionMiddleware	820
28.4.3 Отклонение HTTP-запросов в приложениях API	823
Резюме	824

29 Повышаем безопасность приложения 826

29.1 Защита от межсайтового скриптинга	827
29.2 Защита от межсайтовой подделки запросов (CSRF)	831
29.3 Вызов веб-API из других доменов с помощью CORS	837
29.3.1 Что такое CORS и как он работает	838
29.3.2 Добавление глобальной политики CORS ко всему приложению	840
29.3.3 Добавление CORS к определенным конечным точкам с помощью метаданных EnableCors	842
29.3.4 Настройка политик CORS	844
29.4 Изучение других векторов атак	846
29.4.1 Обнаружение и предотвращение атак с открытым перенаправлением	846
29.4.2 Предотвращение атак с использованием внедрения SQL-кода с помощью EF Core и параметризации	848
29.4.3 Предотвращение небезопасных прямых ссылок на объекты	850
29.4.4 Защита паролей и данных пользователей	851
Резюме	853

ЧАСТЬ V. Дальнейшая работа с ASP.NET Core855

30 Создание приложений ASP.NET Core с помощью обобщенного хоста и класса Startup 857

30.1 Разделение ответственностей между двумя файлами	858
30.2 Класс Program: сборка веб-хоста	859
30.3 Класс Startup: настройка приложения	861
30.4 Создание собственного экземпляра IHostBuilder	866
30.5 Сложность обобщенного хоста	869

30.6 Выбор между обобщенным хостом и минимальным хостингом	872
Резюме	874

31 *Расширенная настройка ASP.NET Core* 877

31.1 Настройка конвейера промежуточного ПО	878
31.1.1 Создание простых приложений с помощью метода расширения Run	879
31.1.2 Ветвление конвейера с помощью метода расширения Map	881
31.1.3 Добавление в конвейер с помощью метода расширения Use....	885
31.1.4 Создание пользовательского компонента промежуточного программного обеспечения	888
31.1.5 Преобразование промежуточного ПО в конечные точки	892
31.2 Использование внедрения зависимостей с OptionsBuilder и IConfigurationOptions	896
31.3 Использование стороннего контейнера внедрения зависимостей ...	902
Резюме	905

32 *Создание собственных компонентов MVC и Razor Pages*..... 907

32.1 Создание собственного тег-хелпера Razor	908
32.1.1 Вывод информации об окружении с помощью собственного тег-хелпера	909
32.1.2 Создание специального тег-хелпера для условного скрывания элементов	913
32.1.3 Создание тег-хелпера для преобразования Markdown в HTML.....	915
32.2 Компоненты представления: добавление логики в частичные представления	917
32.3 Создание собственного атрибута валидации.....	923
32.4 Замена фреймворка валидации на FluentValidation	928
32.4.1 Сравнение FluentValidation и атрибутов DataAnnotations	929
32.4.2 Добавляем FluentValidation в приложение	933
Резюме	934

33 *Вызов удаленных API с помощью IHttpClientFactory* 937

33.1 Вызов API по протоколу HTTP: проблема с классом HttpClient	938
33.2 Создание экземпляров класса HttpClient с помощью интерфейса IHttpClientFactory	944
33.2.1 Использование IHttpClientFactory для управления жизненным циклом HttpClientHandler	944
33.2.2 Настройка именованных клиентов во время регистрации	948
33.2.3 Использование типизированных клиентов для инкапсуляции HTTP-вызовов.....	949
33.3 Обработка временных ошибок HTTP с помощью библиотеки Polly....	952
33.4 Создание специального обработчика HttpResponseMessage.....	955
Резюме	958

34 *Создание фоновых задач и сервисов*..... 960

34.1 Запуск фоновых задач с помощью IHostedService	961
34.1.1 Запуск фоновых задач по таймеру	962

34.1.2	Использование сервисов с жизненным циклом Scored в фоновых задачах	967
34.2	Создание сервисов рабочей роли без пользовательского интерфейса с использованием IHost	969
34.2.1	Создание сервиса рабочей роли из шаблона.....	971
34.2.2	Запуск сервисов рабочей роли в промышленном окружении	974
34.3	Координация фоновых задач с помощью Quartz.NET.....	977
34.3.1	Установка Quartz.NET в приложение ASP.NET Core	978
34.3.2	Настройка задания для запуска по расписанию с помощью Quartz.NET	980
34.3.3	Использование кластеризации для повышения избыточности фоновых задач.....	983
	Резюме	987
35	Тестирование приложений с помощью xUnit.....	989
35.1	Тестирование в ASP.NET Core	990
35.2	Создание первого тестового проекта с xUnit.....	992
35.3	Запуск тестов командой dotnet test.....	995
35.4	Ссылка на ваше приложение из тестового проекта	996
35.5	Добавление модульных тестов с атрибутами Fact и Theory	999
35.6	Тестирование условий сбоя	1003
	Резюме	1004
36	Тестирование приложений ASP.NET Core	1006
36.1	Модульное тестирование пользовательского промежуточного ПО.....	1007
36.2	Модульное тестирование контроллеров API и конечных точек минимальных API.....	1010
36.3	Интеграционное тестирование: тестирование всего приложения в памяти.....	1015
36.3.1	Создание TestServer с помощью пакета Test Host	1016
36.3.2	Тестирование приложения с помощью WebApplicationFactory	1019
36.3.3	Замена зависимостей в классе WebApplicationFactory	1022
36.3.4	Уменьшение дублирования кода за счет создания своего класса WebApplicationFactory	1024
36.4	Изоляция базы данных с помощью поставщика EF Core в памяти.....	1026
	Резюме	1031
	Приложение А. Подготовка окружения разработки	1033
A.1	Установка .NET SDK	1034
A.2	Выбор интегрированной среды разработки или редактора кода	1035
A.2.1	Visual Studio (Windows)	1036
A.2.2	Rider от компании JetBrains (Windows, Linux, macOS)	1037
A.2.3	Visual Studio для Mac (macOS)	1038
A.2.4	Visual Studio Code (Windows, Linux, macOS)	1038

<i>Приложение Б. Полезные ссылки</i>	1040
Б.1 Список литературы	1040
Б.2 Анонсы в блогах	1041
Б.3 Документация Microsoft	1042
Б.4 Ссылки по темам, связанным с безопасностью	1042
Б.5 Репозитории ASP.NET Core на GitHub.....	1043
Б.6 Инструменты и сервисы	1043
Б.7 Блоги ASP.NET Core.....	1043
Б.8 Ссылки на видео.....	1044

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Вступительное слово от сообщества DotNet.Ru

.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, а значит, каждый может изучить любой аспект его работы. Однако объем кода велик, и не так просто сразу понять, что именно искать и как с этим разбираться. Microsoft предоставляет отличную документацию по ASP.NET Core и основам серверной веб-разработки на официальном сайте, но этого может быть недостаточно для выстраивания целостной картины.

Много полезной информации можно почерпнуть из книг, и по ASP.NET Core их написано уже немало. Одной из таких книг стала «ASP.NET Core в действии» (2-е издание) от Эндрю Лока, вышедшая несколько лет назад. Но .NET не стоит на месте, появляются новые концепции, изменяются распространенные сценарии использования, и чтобы быть в курсе современных стилей и подходов к программированию на .NET с использованием ASP.NET Core, мы представляем вам третье издание этой прекрасной книги.

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. Третье издание пополнилось новыми главами про минимальные API, а также существенно расширился материал о безопасной разработке. Автору удалось охватить широту фреймворка, рассмотрев большое количество различных аспектов и при этом достаточно глубоко разобрав 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 стремительно развивается, постоянно появляются новые термины, и даже те, что давно используются, не всегда имеют устоявшийся в профессиональной среде перевод. Мы обсуждали, спорили, думали о том, как читатели будут искать термины в сети

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru