

Рикардо Террелл

Конкурентность и параллелизм на платформе .NET

Паттерны эффективного
проектирования



 MANNING



Concurrency in .NET

Modern patterns of concurrent and parallel programming

RICCARDO TERRELL



MANNING
SHELTER ISLAND

Рикардо Террелл

**Конкурентность
и параллелизм
на платформе
.NET**

Паттерны
эффективного
проектирования



Санкт-Петербург • Москва • Екатеринбург • Воронеж
Нижний Новгород • Ростов-на-Дону • Самара • Минск

2019

ББК 32.973.2-018-02
УДК 004.4
Т35

Террелл Рикардо

Т35 Конкурентность и параллелизм на платформе .NET. Паттерны эффективного проектирования. — СПб.: Питер, 2019. — 624 с.: ил. — (Серия «Для профессионалов»). ISBN 978-5-4461-1072-8

Рикардо Террелл научит вас писать идеальный код, с которым любые приложения будут просто летать. Книга содержит примеры на языках C# и F#, описывает паттерны проектирования конкурентных и параллельных программ как в теории, так и на практике.

Вы начнете с теоретических основ параллелизма, после чего перейдете к примерам и проверенным решениям, помогающим создавать и оптимизировать код для современных многопроцессорных систем.

В этой книге автор раскрыл важнейшие конкурентные абстракции, реализацию потоковой обработки событий в реальном времени и наилучшие конкурентные паттерны и практики, применимые на любых платформах.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.2-018-02
УДК 004.4

Права на издание получены по соглашению с Apress. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги. Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

ISBN 978-1617292996 англ.
ISBN 978-5-4461-1072-8

© 2018 by Manning Publications Co. All rights reserved
© Перевод на русский язык ООО Издательство «Питер», 2019
© Издание на русском языке, оформление ООО Издательство «Питер», 2019
© Серия «Для профессионалов», 2019

Краткое содержание

Предисловие.....	18
Благодарности	22
Об этой книге	24
Об авторе	28
Об иллюстрации на обложке	29

Часть I. Преимущества функционального программирования в применении к конкурентным программам

Глава 1. Основы функциональной конкурентности	33
Глава 2. Технологии функционального программирования для конкурентных систем... 66	
Глава 3. Функциональные структуры данных и неизменяемость.....	99

Часть II. Конкурентная программа: разные части, разные подходы

Глава 4. Основы обработки больших данных: распараллеливание данных, часть 1... 145	
Глава 5. PLINQ и MapReduce: распараллеливание данных, часть 2	169
Глава 6. Поток событий реального времени: функциональное реактивное программирование	204
Глава 7. Функциональный параллелизм на основе задач.....	243

Глава 8. Асинхронность задач — путь к победе.....	279
Глава 9. Асинхронное функциональное программирование на F#	318
Глава 10. Функциональные комбинаторы для быстрого конкурентного программирования	350
Глава 11. Реактивное программирование с использованием агентов.....	411
Глава 12. Параллельный рабочий процесс и агентное программирование с помощью TPL Dataflow	453

Часть III. Современные шаблоны конкурентного программирования

Глава 13. Рецепты и шаблоны для успешного конкурентного программирования.....	489
Глава 14. Построение масштабируемого мобильного приложения методом конкурентного функционального программирования	547

Приложения

Приложение А. Функциональное программирование	588
Приложение Б. Обзор F#	603
Приложение В. Совместимость асинхронного рабочего процесса F# и задач .NET ...	620

Оглавление

Предисловие.....	18
Благодарности.....	22
Об этой книге	24
Кому следует прочесть эту книгу.....	24
Структура издания: дорожная карта.....	25
О коде.....	27
От издательства	27
Об авторе	28
Об иллюстрации на обложке	29

Часть I. Преимущества функционального программирования в применении к конкурентным программам

Глава 1. Основы функциональной конкурентности	33
1.1. Что вы узнаете из этой книги	35
1.2. Начнем с терминологии.....	36
1.2.1. При последовательном программировании задачи выполняются одна за другой.....	37
1.2.2. При конкурентном программировании выполняется несколько задач в одно и то же время	38
1.2.3. При параллельном программировании выполняется несколько задач одновременно	39
1.2.4. При многозадачности выполняется несколько задач одновременно.....	41
1.2.5. Многопоточность как средство повышения производительности.....	42
1.3. Зачем нужна конкурентность	44

1.4.	Ловушки параллельного программирования.....	47
1.4.1.	Риски конкурентности.....	48
1.4.2.	Эволюция разделяемых состояний.....	51
1.4.3.	Простой пример из практики: параллельная быстрая сортировка	52
1.4.4.	Бенчмаркинг в F#	56
1.5.	Почему для конкурентности выбирают функциональное программирование	57
1.6.	Область применения функциональной парадигмы.....	61
1.7.	Зачем использовать F# и C# для функционального конкурентного программирования	62
	Резюме	65
Глава 2. Технологии функционального программирования для конкурентных систем...		66
2.1.	Использование компоновки функций для решения сложных задач.....	67
2.1.1.	Функциональная компоновка в C#	68
2.1.2.	Функциональная компоновка в F#	70
2.2.	Использование замыканий для упрощения функционального мышления.....	71
2.2.1.	Захваченные переменные в замыканиях с лямбда-выражениями	72
2.2.2.	Замыкания в многопоточной среде	75
2.3.	Технология мемоизации и кэширования для ускорения программы.....	77
2.4.	Применение мемоизации для создания быстрого веб-робота	81
2.5.	«Ленивая» мемоизация для повышения производительности.....	85
2.6.	Эффективная конкурентная упреждающая обработка для уменьшения издержек на затратные вычисления	87
2.6.1.	Предварительные вычисления с естественной поддержкой функционального программирования	89
2.6.2.	Пусть победит быстрее!	91
2.7.	Лень — это хорошо	92
2.7.1.	Использование строгих языков программирования для лучшего понимания конкурентного поведения	92
2.7.2.	«Ленивое» кэширование и потокобезопасный шаблон «Одиночка»	94
2.7.3.	Поддержка «ленивых» вычислений в F#.....	96
2.7.4.	Мощное сочетание Lazy и Task.....	96
	Резюме	98
Глава 3. Функциональные структуры данных и неизменяемость.....		99
3.1.	Практический пример: охота на потокобезопасный объект.....	100
3.1.1.	Неизменяемые коллекции .NET: надежное решение.....	104
3.1.2.	Конкурентные коллекции .NET: более быстрое решение	109
3.1.3.	Шаблон агента передачи сообщений: более быстрое и верное решение.....	112

3.2.	Безопасное совместное использование потоками функциональных структур данных.....	114
3.3.	Наконец-то неизменяемость!.....	115
3.3.1.	Функциональная структура данных для обеспечения их параллелизма.....	119
3.3.2.	Влияние неизменяемости на производительность.....	119
3.3.3.	Неизменяемость в C#	120
3.3.4.	Неизменяемость в F#	123
3.3.5.	Функциональные списки: объединение ячеек в цепочки	124
3.3.6.	Построение персистентной структуры данных: неизменяемое бинарное дерево	131
3.4.	Рекурсивные функции: естественный способ итерирования	134
3.4.1.	Хвост правильной рекурсивной функции: оптимизация хвостового вызова	135
3.4.2.	Оптимизация рекурсивной функции в стиле передачи продолжений	137
	Резюме	142

Часть II. Конкурентная программа: разные части, разные подходы

Глава 4.	Основы обработки больших данных: распараллеливание данных, часть 1	145
4.1.	Что такое распараллеливание данных.....	146
4.1.1.	Распараллеливание данных и задач.....	147
4.1.2.	Концепция «естественной параллельности»	148
4.1.3.	Поддержка распараллеливания данных в .NET	149
4.2.	Шаблон Fork/Join: параллельный алгоритм Мандельброта	150
4.2.1.	Когда узким местом является сборка мусора: структуры и объекты класса	156
4.2.2.	Оборотная сторона параллельных циклов	159
4.3.	Измерение производительности	160
4.3.1.	Определение предела повышения эффективности по закону Амдала.....	161
4.3.2.	Закон Густафсона: еще один шаг вперед в измерении повышения производительности	162
4.3.3.	Ограничения параллельных циклов: сумма простых чисел	162
4.3.4.	Что может пойти не так в простом цикле.....	164
4.3.5.	Модель декларативного параллельного программирования	166
	Резюме	168

Глава 5. PLINQ и MapReduce: распараллеливание данных, часть 2	169
5.1. Краткое введение в PLINQ.....	170
5.1.1. Почему язык PLINQ более функциональный.....	171
5.1.2. PLINQ и чистые функции: параллельный счетчик слов	172
5.1.3. Исключение побочных эффектов посредством чистых функций.....	174
5.1.4. Изоляция и контроль побочных эффектов: рефакторинг параллельного счетчика слов	176
5.2. Агрегирование и сокращение данных в параллельных программах	177
5.2.1. Усечение: одно из многих преимуществ свертки.....	180
5.2.2. Свертки в PLINQ: функции агрегирования	182
5.2.3. Реализация параллельной функции Reduce на PLINQ	188
5.2.4. Параллельное списковое включение в F#: PSeq	191
5.2.5. Параллельные массивы в F#.....	192
5.3. Параллельный шаблон MapReduce	193
5.3.1. Функции Map и Reduce.....	195
5.3.2. Использование MapReduce совместно с галереей пакетов NuGet	196
Резюме	203
Глава 6. Потоки событий реального времени: функциональное реактивное программирование	204
6.1. Реактивное программирование: обработка больших событий.....	206
6.2. Инструментарий .NET для реактивного программирования.....	209
6.2.1. Улучшенное решение: комбинаторы событий	210
6.2.2. Совместимость .NET с комбинаторами F#	211
6.3. Реактивное программирование в .NET: реактивные расширения (Rx).....	214
6.3.1. От LINQ/PLINQ к Rx	217
6.3.2. IObservable и IEnumerable: дуальные интерфейсы	218
6.3.3. Реактивные расширения в действии	219
6.3.4. Поточковая передача в реальном времени с помощью Rx	220
6.3.5. От событий к наблюдаемым объектам F#	221
6.4. Укрощение потока событий: анализ эмоций в Twitter посредством Rx-программирования	222
6.5. Шаблон «издатель — подписчик» в Rx.....	232
6.5.1. Использование типа Subject для создания мощного концентратора в шаблоне «издатель — подписчик»	233
6.5.2. Rx и конкурентность	234
6.5.3. Реализация на Rx многоразового шаблона «издатель — подписчик»	235
6.5.4. Анализ эмоций твитов с помощью Rx-класса Pub-Sub.....	237
6.5.5. Наблюдатели в действии	240
6.5.6. Удобное выражение объектов в F#	240
Резюме	242

Глава 7. Функциональный параллелизм на основе задач.....	243
7.1. Краткое введение в параллелизм.....	244
7.1.1. Зачем нужны параллелизм задач и функциональное программирование.....	245
7.1.2. Поддержка параллелизма задач в .NET.....	246
7.2. Библиотека параллельных задач в .NET.....	248
7.2.1. Выполнение параллельных операций с помощью Parallel.Invoke из библиотеки TPL.....	250
7.3. Проблема void в C#.....	253
7.4. Стил ь продолжений: функциональный поток управления.....	256
7.4.1. Зачем нужен CPS.....	256
7.4.2. Ожидание завершения задачи: модель продолжения.....	258
7.5. Стратегии компоновки операций для выполнения задач.....	263
7.5.1. Использование математических шаблонов для оптимальной компоновки.....	265
7.5.2. Рекомендации по использованию задач.....	271
7.6. Параллельный функциональный шаблон конвейера.....	271
Резюме.....	278
Глава 8. Асинхронность задач — путь к победе.....	279
8.1. Модель асинхронного программирования (APM).....	280
8.1.1. В чем ценность асинхронного программирования.....	281
8.1.2. Асинхронное программирование и масштабируемость.....	284
8.1.3. Операции с ограничениями процессора и ввода-вывода.....	285
8.2. Неограниченный параллелизм при асинхронном программировании.....	286
8.3. Поддержка асинхронности в .NET.....	287
8.3.1. Асинхронное программирование нарушает структуру кода.....	290
8.3.2. Асинхронное программирование на основе событий.....	291
8.4. Асинхронное программирование на основе задач в C#.....	291
8.4.1. Анонимные асинхронные лямбда-функции.....	295
8.4.2. Монадический контейнер Task<T>.....	296
8.5. Асинхронное программирование на основе задач: практический пример.....	299
8.5.1. Отмена асинхронных операций.....	304
8.5.2. Асинхронная компоновка на основе задач с монадическим оператором Bind.....	307
8.5.3. Отсрочка асинхронного вычисления, обеспечивающая компоновку.....	309
8.5.4. Повторная попытка в случае, если что-то пошло не так.....	310
8.5.5. Обработка ошибок в асинхронных операциях.....	312
8.5.6. Асинхронная параллельная обработка изменений фондового рынка.....	314
8.5.7. Асинхронная параллельная обработка данных фондового рынка после завершения выполнения задач.....	315
Резюме.....	317

Глава 9. Асинхронное функциональное программирование на F#	318
9.1. Аспекты асинхронного функционального программирования	319
9.2. Что такое асинхронный рабочий процесс F#	319
9.2.1. Стиль прохождения продолжений в вычислительных выражениях.....	320
9.2.2. Асинхронный рабочий процесс в действии: параллельные операции сохранения данных из Azure Blob	322
9.3. Асинхронные вычислительные выражения	328
9.3.1. Различия между вычислительными выражениями и монадами.....	330
9.3.2. AsyncRetry: построение собственных вычислительных выражений.....	331
9.3.3. Расширение асинхронного рабочего процесса	334
9.3.4. Отображение асинхронных операций: функтор Async.map.....	335
9.3.5. Распараллеливание асинхронных рабочих процессов: Async.Parallel.....	337
9.3.6. Поддержка отмены асинхронного рабочего процесса	343
9.3.7. Управление параллельными асинхронными операциями.....	345
Резюме	349
Глава 10. Функциональные комбинаторы для быстрого конкурентного программирования	350
10.1. Поток выполнения не всегда проходит по «счастливому пути»: обработка ошибок.....	351
10.2. Комбинаторы ошибок: Retry, Otherwise и Task.Catch в C#	355
10.2.1. Обработка ошибок в функциональном программировании: исключения при управлении потоком выполнения.....	358
10.2.2. Обработка ошибок с помощью Task<Option<T>> в C#	360
10.2.3. Тип AsyncOption в F#: сочетание Async и Option	361
10.2.4. Функциональная асинхронная обработка ошибок, свойственная F#	362
10.2.5. Сохранение семантики исключений с помощью типа Result	364
10.3. Обработка исключений при асинхронных операциях.....	368
10.3.1. Моделирование обработки ошибок в F# с помощью Async и Result.....	373
10.3.2. Расширение типа F# AsyncResult посредством операторов монадического связывания	374
10.4. Абстрагирование операций с функциональными комбинаторами.....	379
10.5. Коротко о функциональных комбинаторах	380
10.5.1. Встроенные асинхронные комбинаторы TPL.....	381
10.5.2. Использование комбинатора Task.WhenAny для избыточности и чередования	382
10.5.3. Использование комбинатора Task.WhenAll в асинхронном цикле for-each.....	384
10.5.4. Обзор математических шаблонов: что мы уже знаем?	385

10.6. Окончательный вариант параллельной компоновки аппликативного функтора.....	389
10.6.1. Расширение асинхронного рабочего процесса F# с помощью операторов аппликативных функторов	396
10.6.2. Семантика аппликативных функторов и инфиксных операторов в F#	398
10.6.3. Использование аппликативных функторов в гетерогенных параллельных вычислениях	399
10.6.4. Компоновка и выполнение гетерогенных параллельных вычислений	401
10.6.5. Управление потоком с помощью условных асинхронных комбинаторов	404
10.6.6. Как работают асинхронные комбинаторы	408
Резюме	410
Глава 11. Реактивное программирование с использованием агентов.....	411
11.1. Что такое реактивное программирование и чем оно полезно	413
11.2. Программная модель асинхронной передачи сообщений.....	415
11.2.1. Передача сообщений и неизменяемость	417
11.2.2. Естественная изоляция	417
11.3. Что такое агент.....	418
11.3.1. Компоненты агента.....	419
11.3.2. Что может делать агент	420
11.3.3. Подход без разделения ресурсов для конкурентного программирования без блокировок.....	420
11.3.4. Как функционирует агентное программирование	422
11.3.5. Агент является объектно-ориентированным.....	422
11.4. Агенты в F#: MailboxProcessor	423
11.5. Как избежать узких мест при обращении к базе данных с помощью F#-типа MailboxProcessor	426
11.5.1. Тип сообщения MailboxProcessor: размеченные объединения.....	429
11.5.2. Двусторонний обмен данными посредством MailboxProcessor	430
11.5.3. Использование AgentSQL из C#	431
11.5.4. Распараллеливание рабочего процесса и координация групп агентов	433
11.5.5. Как обрабатывать ошибки на F# с помощью MailboxProcessor.....	435
11.5.6. Остановка агентов MailboxProcessor — CancellationToken.....	436
11.5.7. Распределение работы с помощью MailboxProcessor.....	437
11.5.8. Операции кэширования в агентном программировании	439
11.5.9. Получение результатов от MailboxProcessor	443
11.5.10. Использование пула потоков для сообщения о событиях MailboxProcessor.....	446
11.6. F# MailboxProcessor: 10 000 агентов для Game of Life	446
Резюме	452

Глава 12. Параллельный рабочий процесс и агентное программирование с помощью TPL Dataflow	453
12.1. Преимущества TPL Dataflow.....	454
12.2. Блоки TPL Dataflow: созданы для компоновки	455
12.2.1. Использование BufferBlock<TInput> в качестве буфера FIFO.....	457
12.2.2. Преобразование данных с помощью TransformBlock<TInput, TOutput>	458
12.2.3. Законченный пример с ActionBlock<TInput>	459
12.2.4. Связывание блоков в потоке данных	461
12.3. Реализация сложных шаблонов «поставщик — потребитель» с помощью TDF	461
12.3.1. Реализация шаблона «несколько поставщиков — один потребитель» с помощью TDF.....	461
12.3.2. Шаблон «один поставщик — несколько потребителей»	463
12.4. Использование агентной модели в C# с помощью TPL Dataflow.....	464
12.4.1. Свертка состояний и сообщений агента: Aggregate	468
12.4.2. Агентное взаимодействие: параллельный счетчик слов	468
12.5. Параллельный рабочий процесс для сжатия и шифрования больших потоков.....	474
12.5.1. Контекст: проблема обработки большого потока данных.....	474
12.5.2. Сохранение порядка в потоке сообщений	480
12.5.3. Связывание, распространение и завершение	481
12.5.4. Правила построения рабочего процесса TDF	483
12.5.5. Реактивные расширения генерации сетки (Rx) и TDF.....	484
Резюме	486

Часть III. Современные шаблоны конкурентного программирования

Глава 13. Рецепты и шаблоны для успешного конкурентного программирования.....	489
13.1. Освобождение памяти, занимаемой объектами, для сокращения потребления памяти.....	490
13.2. Нестандартный параллельный оператор Fork/Join	494
13.3. Распараллеливание задач с зависимостями: разработка кода для оптимизации производительности	497
13.4. Шлюз для координации конкурентных операций ввода-вывода с разделяемыми ресурсами: одна операция записи, несколько операций чтения	502
13.5. Потокбезопасный генератор случайных чисел.....	508
13.6. Полиморфный агрегатор событий	510

13.7. Нестандартный Rx-планировщик для управления степенью параллелизма.....	513
13.8. Конкурентное реактивное масштабируемое приложение «клиент-сервер»	516
13.9. Многоразовый специальный высокопроизводительный параллельный оператор фильтрации-отображения	527
13.10. Неблокирующая синхронная модель передачи сообщений	532
13.11. Координирование конкурентных заданий с использованием агентной модели программирования	537
13.12. Компоновка монадических функций	542
Резюме	546
Глава 14. Построение масштабируемого мобильного приложения методом конкурентного функционального программирования	547
14.1. Практическое применение функционального программирования для серверной части приложения	548
14.2. Как разработать высокопроизводительное приложение	550
14.2.1. Ноу-хау: ACD	551
14.2.2. Другой асинхронный шаблон: постановка в очередь для отложенного выполнения	552
14.3. Правильный выбор конкурентной модели программирования	553
14.4. Торговля акциями в реальном времени: архитектура высокого уровня для фондового рынка	557
14.5. Главные элементы приложения для фондового рынка	562
14.6. Пишем код приложения для торговли на фондовом рынке	563
Резюме	586

Приложения

Приложение А. Функциональное программирование	588
Что такое функциональное программирование	588
Преимущества функционального программирования	589
Основные принципы функционального программирования	590
Противостояние программных парадигм: от императивного к объектно-ориентированному и функциональному программированию	590
Применение функций высшего порядка для увеличения абстракции	592
Применение функций высшего порядка и лямбда-выражений для создания многократно используемого кода	593
Лямбда-выражения и анонимные функции	593
Каррирование	595
Частично примененные функции	599
Преимущества частичного применения и каррирования функций в C#	601

Приложение Б. Обзор F#	603
let-привязки	603
Сигнатуры функций в F#	604
Создание изменяемых типов: mutable и ref	604
Функции как типы первого класса	605
Компоновка: операторы конвейера и компоновки	605
Делегаты	606
Комментарии	606
Оператор open	606
Основные типы данных	607
Специальное определение строки	607
Кортежи	607
Записи	608
Размеченные объединения	609
Сопоставление с образцом	610
Активные образцы	611
Коллекции	612
Массивы	612
Последовательности (seq)	613
Списки	613
Множества	614
Словари	614
Циклы	614
Классы и наследование	615
Абстрактные классы и наследование	615
Интерфейсы	616
Объектные выражения	617
Приведение типов	617
Единицы измерения	618
Краткая справка по API модуля событий	618
Приложение В. Совместимость асинхронного рабочего процесса F# и задач .NET ...	620

Я посвящаю эту книгу своей жене Бриони — замечательной, всегда готовой прийти на помощь. Твоя поддержка, любовь, забота и постоянное воодушевление, пока я писал книгу, — только они позволили мне превратить настоящий проект в жизнь. Я люблю тебя и преклоняюсь перед тобой за все твои усилия и терпение, пока я был занят данной работой. Спасибо, что всегда верила в меня.

Я также посвящаю эту книгу Буггине и Стеллине, двум моим верным мопсам, которые были постоянно со мной при ее написании, беззаветно и с энтузиазмом поддерживая меня. Вы лучшие друзья человека и самые горячие мохнатые фанаты автора.

Предисловие

Эту книгу, «Конкурентность в .NET», вы читаете, возможно, потому, что хотите научиться создавать невероятно быстрые приложения или узнать, как резко повысить производительность уже существующего приложения. Вас заботит производительность, потому что вы посвятили себя созданию более быстрых программ и потому что вас восхищает, когда всего несколько изменений в коде делают приложение более быстрым и отзывчивым. Параллельное программирование предоставляет бесконечные возможности разработчикам, которые влюблены в свое дело и хотят внедрять новые технологии. В отношении производительности преимущества применения параллелизма в программировании невозможно переоценить. Но при использовании императивного и объектно-ориентированного стиля программирования (ООП) написание конкурентного кода может оказаться запутанным и сложным делом. Именно поэтому конкурентное программирование не стало распространенной практикой и крупным ведущим программистам приходилось искать другие варианты.

В колледже я прослушал курс функционального программирования. В то время я изучал Haskell и, несмотря на большие учебные нагрузки, наслаждался каждым занятием. Помню, как, увидев первые примеры, был поражен элегантностью и простотой решений. Пятнадцать лет спустя, когда я стал искать варианты улучшения своих программ с использованием конкурентности, я снова вспомнил те уроки. На этот раз я в полной мере оценил, насколько мощным и полезным было бы применение функционального программирования в моих текущих задачах. У функционального стиля программирования есть несколько преимуществ; каждое из них мы обсудим в данной книге.

Знания, полученные во время учебы, пригодились мне в профессиональной деятельности, когда пришлось создавать программную систему для сферы здравоохранения. Этот проект требовал разработки приложения для анализа медицинских

рентгеновских снимков. Обработка подобных изображений состояла из нескольких этапов, таких как уменьшение шума, обработка по алгоритму Гаусса, интерполяция и фильтрация, в результате чего черно-белые изображения становились цветными. Приложение было разработано на Java и сначала функционировало так, как и ожидалось. Но однажды, как это часто бывает, нагрузку увеличили, и начались проблемы. В программе не было никаких неполадок или ошибок, но с увеличением количества анализируемых изображений она стала работать медленнее.

Естественно, первым предложением для решения данной проблемы было приобрести более мощный сервер. На тот момент это было правильное решение, но сегодня, купив новую машину с целью получить более высокую вычислительную скорость процессора, вы будете разочарованы. Дело в том, что у современного процессора несколько ядер и скорость каждого из них не выше, чем у ядра, приобретенного в 2007 г. Лучшим и более надежным вариантом (вместо того чтобы покупать новый сервер или компьютер) было использовать параллелизм, чтобы реализовать преимущества многоядерного оборудования, задействовав все его ресурсы, что в итоге ускорило бы обработку изображений.

Теоретически это была простая задача, но на практике не все было так тривиально. Мне пришлось научиться применять потоки и устанавливать блокировки, и, к сожалению, я на собственном опыте узнал, что такое взаимная блокировка.

Эта взаимная блокировка побудила меня внести существенные изменения в код приложения. Их было так много, что я допустил ошибки, даже не связанные с первоначальной целью изменений. Я был расстроен: код стал неподдерживаемым и нестабильным и в процессе постоянно возникали новые ошибки. Мне пришлось временно отложить решение исходной проблемы и посмотреть на задачу с другой точки зрения. Должен был существовать способ получше.

Инструменты, которые мы используем, оказывают глубокое (и коварное!) влияние на наши мыслительные привычки и, следовательно, на наши интеллектуальные способности.

Эдсгер Дейкстра (Edsger Dijkstra)

Проведя несколько дней в поисках выхода из этого многопоточного кошмара, я нашел ответ. Все, что я исследовал и читал, указывало на функциональную парадигму. Принципы, изученные мною в колледже много лет назад, стали тем механизмом, который позволил мне продвинуться вперед. Я переписал ядро приложения для обработки изображений с учетом параллельной работы, используя функциональный язык программирования. Сначала переход от императивного к функциональному стилю вызывал трудности. Я забыл почти все, что выучил в колледже, и не горжусь тем, что написал тогда на функциональном языке код, выглядевший очень объектно-ориентированным; но в целом это было успешное решение. После компиляции новая программа заработала без ошибок, резко повысилась производительность, аппаратные ресурсы были использованы полностью. Но самым неожиданным было то, что функциональное программирование привело

к впечатляющему сокращению числа строк кода: их стало почти на 50 % меньше, чем в исходной реализации на объектно-ориентированном языке.

Этот опыт заставил меня пересмотреть отношение к ООП как к решению всех проблем программирования. Я понял, что перспективы данной модели программирования и ее подхода к решению проблем ограничены. Мое путешествие в мир функционального программирования началось с потребности в хорошей модели конкурентного программирования.

С тех пор я проявляю большой интерес к применению функционального программирования для многопоточности и конкурентности. Там, где другие усматривали сложную проблему и источник трудностей, я видел решение в функциональном программировании как в мощном инструменте, позволяющем ускорить работу, используя имеющееся оборудование. Я пришел к пониманию того, как данная дисциплина приводит к последовательной, удобной и красивой форме написания конкурентных программ.

Впервые идея создания этой книги возникла у меня в июле 2010 г., после того как Microsoft представила F# как часть Visual Studio 2010. Уже в то время было ясно, что все больше основных языков программирования — включая C#, C++, Java и Python — поддерживают функциональную парадигму. В 2007 г. в C# 3.0 появились функции первого класса, а также новые конструкции, такие как лямбда-выражения и вывод типов, что позволило программистам использовать концепции функционального программирования. Вскоре после этого появился Language Integrated Query (LINQ), допускающий декларативное программирование.

Платформа .NET особенно сильно погрузилась в мир функционального программирования. После F# у Microsoft появились полнофункциональные языки, поддерживающие как объектно-ориентированную, так и функциональную парадигму. Кроме того, объектно-ориентированные языки, такие как C#, становятся все более гибридными и устраняют разрыв между различными парадигмами, что позволяет применять оба стиля программирования.

Более того, мы вступаем в эпоху многоядерности, когда мощность процессоров измеряется не числом тактов в секунду, а количеством доступных ядер. При такой тенденции однопоточные приложения не позволят повысить скорость на многоядерных системах, если не интегрировать в них параллелизм и не использовать алгоритмы для распределения работы между несколькими ядрами.

Мне стало ясно, что многопоточность востребована, и я загорелся идеей поделиться с вами таким подходом к программированию. В этой книге показано, как, применяя возможности параллельного программирования и функциональной парадигмы, писать легко читаемый, лучше разбитый на модули, поддерживаемый код на языках C# и F#. Благодаря данным технологиям ваш код будет работать с максимальной скоростью при меньшем количестве строк, что приведет к повышению производительности и отказоустойчивости программ.

Настали великолепные времена — можно начинать разработку многопоточного кода. Сегодня компании — разработчики программного обеспечения создают больше, чем когда-либо, инструментов и средств, позволяющих выбирать стиль программирования без каких-либо компромиссов. Первоначальные проблемы, связанные

с освоением параллельного программирования, быстро уйдут, а награда за вашу настойчивость будет неизмеримой. Независимо от области знаний, будь вы разработчик серверной или клиентской части приложений, создатель облачных продуктов или видеоигр, вам все равно придется применять параллелизм для повышения производительности и создания масштабируемых приложений.

Эта книга основана на моем опыте функционального программирования для написания конкурентных программ в .NET с использованием C# и F#. Я уверен, что функциональное программирование фактически становится способом написания конкурентного кода для координации асинхронных и параллельных программ в .NET. Я также считаю, что данная книга даст вам все необходимое для освоения этого захватывающего мира многоядерных компьютеров.

Благодарности

Написать книгу — всегда подвиг. Сделать это на неродном языке бесконечно сложнее и страшнее. Что касается меня, то мне бы такое и не приснилось, не будь у меня группы поддержки размером с небольшой город. Я хотел бы поблагодарить всех, кто поддерживал меня и участвовал в создании данной книги.

Мои приключения с F# начались в 2013 г., когда я посетил FastTrack по F# в Нью-Йорке. Там я встретил Томаса Петричека (Tomas Petricek), который вдохновил меня нырнуть с головой в мир F#. Томас пригласил меня в сообщество и с тех пор является моим наставником и поверенным моих тайн.

Я неизмеримо благодарен фантастическому персоналу Manning Publications. Пятнадцать месяцев назад я начал тяжелый труд над данной книгой с редактором — консультантом по аудитории Дэном Магарри (Dan Maharry), затем продолжил его с Мариной Майклз (Marina Michaels). Оба они были моими терпеливыми и мудрыми руководителями в этой удивительной работе.

Спасибо многим рецензентам, особенно научному редактору-консультанту Майклу Лунду (Michael Lund) и техническому корректору Вайорель Моисей (Viorel Moisei). Ваш критический анализ позволил мне убедиться: я правильно передал на бумаге все, что было у меня в голове, хотя большая часть из этого рисковала потеряться при «переводе». Спасибо также тем, кто участвовал в программе Manning MEAP и оказал поддержку в качестве рецензентов: Энди Киршу (Andy Kirsch), Антону Херцог (Anton Herzog), Крису Бойярду (Chris Bolyard), Крейгу Фуллертону (Craig Fullerton), Гарету ван дер Бергу (Gareth van der Berg), Джереми Ланге (Jeremy Lange), Джиму Вельху (Jim Velch), Джоэлу Котарски (Joel Kotarski), Кевину Орру (Kevin Orr), Люку Бирлу (Luke Bearn), Павлу Климчику (Pawel Klimczyk), Рикардо Пересу (Ricardo Peres), Рохиту Шарме (Rohit Sharma), Стефано Дриусси (Stefano Driussi) и Субхазису Гошу (Subhasis Ghosh).

Я получил огромную поддержку от членов сообщества F#, сплотившихся вокруг меня на этом пути, особенно от Сергея Тихона (Sergey Tihon), который оказался прекрасным слушателем.

И спасибо моей семье и друзьям, которые подбадривали меня и терпеливо ждали, когда я снова вернусь в мир общих уик-эндов, обедов и отдыха.

Прежде всего я бы хотел выразить признательность моей жене, которая поддерживала все мои усилия и никогда не позволяла мне уходить от проблем.

Я также признателен моим преданным и верным мопсам, Буггине и Стеллине, которые всегда были на моей стороне — или на моих коленях, — когда я писал эту книгу глубокой ночью. А во время наших долгих вечерних прогулок я имел возможность освежать голову и находить лучшие идеи для книги.

Об этой книге

Книга «Конкурентность и параллелизм на платформе .NET» дает представление о рекомендуемых методах создания конкурентных и масштабируемых программ в .NET, освещая преимущества функциональной парадигмы и предоставляя соответствующие инструменты и принципы, позволяющие легко и правильно поддерживать конкурентность. В итоге, вооружившись новыми навыками, вы получите знания, необходимые для того, чтобы стать экспертом в предоставлении успешных высокопроизводительных решений.

Кому следует прочесть эту книгу

Если вы пишете многопоточный код на .NET, то эта книга может вам помочь. Если вы заинтересованы в использовании функциональной парадигмы для упрощения конкурентного программирования и максимального повышения производительности приложений, то данная книга станет для вас важным руководством. Она принесет пользу любым разработчикам на .NET, желающим писать конкурентные, реактивные и асинхронные приложения, которые масштабируются и автоматически адаптируются к имеющимся аппаратным ресурсам везде, где бы ни работали такие программы.

Данная книга также подойдет тем разработчикам, которых интересует применение функционального программирования для реализации конкурентных методов. Предварительные знания или опыт работы с функциональной парадигмой для этого не требуются, а основные понятия функциональной парадигмы рассматриваются в приложении А.

В примерах кода используются языки C# и F#. Читатели, знакомые с C#, сразу почувствуют себя комфортно. Знакомство с языком F# не является обязательным,

его основы рассмотрены в приложении Б. Опыт и знания в области функционального программирования не требуются: в книгу включены все необходимые понятия.

Предполагается хорошее знание .NET. Вы должны иметь некоторый опыт работы с коллекциями .NET и обладать знаниями .NET Framework как минимум версии .NET 3.5 (LINQ-делегаты `Action<>` и `Func<>`). Наконец, эта книга подходит для любой платформы, поддерживаемой .NET (включая .NET Core).

Структура издания: дорожная карта

Четырнадцать глав этой книги разделены на три части. В части I представлены функциональные концепции конкурентного программирования и описаны навыки, необходимые для понимания функциональных аспектов написания многопоточных программ.

- ❑ В главе 1 описаны основные понятия и цели конкурентного программирования, а также причины применения функционального программирования для написания многопоточных приложений.
- ❑ В главе 2 исследуется ряд технологий функционального программирования для повышения производительности многопоточных приложений. Цель этой главы — предоставить читателю концепции, используемые в остальной книге, и познакомить с мощными идеями, возникающими из функциональной парадигмы.
- ❑ В главе 3 дается обзор функциональной концепции неизменяемости. Здесь объясняется, как неизменяемость применяется для написания предсказуемых и корректных конкурентных программ и для реализации функциональных структур данных, которые являются потокобезопасными по своей сути.

В части II углубленно рассматриваются различные модели конкурентного программирования в функциональной парадигме. Мы изучим такие темы, как библиотека Task Parallel Library (TPL), и реализуем параллельные шаблоны, такие как Fork/Join, «разделяй и властвуй» и MapReduce. В этой части также обсуждаются декларативная компоновка, высокоуровневые абстракции в асинхронных операциях, агентное программирование и семантика передачи сообщений.

- ❑ В главе 4 изложены основы параллельной обработки большого количества данных, включая такие шаблоны, как Fork/Join.
- ❑ В главе 5 представлены более сложные методы параллельной обработки больших объемов информации, такие как параллельное агрегирование, сокращение данных и реализация параллельного шаблона MapReduce.
- ❑ В главе 6 представлена подробная информация о функциональных методах обработки потоков событий (данных) в реальном времени с применением функциональных операторов высокого порядка в .NET Reactive Extensions для формирования асинхронных комбинаторов событий. Изученные методы затем будут использованы для реализации рассчитанного на конкурентность реактивного шаблона «издатель — подписчик».

- ❑ В главе 7 дается объяснение модели программирования на основе задач в применении к функциональному программированию для реализации конкурентных операций с использованием шаблона `Monadic`. Затем этот метод применяется для построения конкурентного конвейера на основе функциональной парадигмы программирования.
- ❑ Глава 8 посвящена реализации неограниченных параллельных вычислений с помощью модели асинхронного программирования на `C#`. В этой главе также рассматриваются методы обработки ошибок и методы построения асинхронных операций.
- ❑ В главе 9 описывается асинхронный рабочий процесс на `F#`. В ней показано, как отсроченная и явная оценка в этой модели позволяет получить более высокую композиционную семантику. Затем мы научимся реализовывать пользовательские вычислительные выражения для повышения уровня абстракции до декларативного программирования.
- ❑ В главе 10 показано, как на основе знаний, полученных в предыдущих главах, можно реализовать комбинаторы и шаблоны, такие как `Functor`, `Monad` и `Applicative`, для составления и запуска нескольких асинхронных операций и обработки ошибок без побочных эффектов.
- ❑ В главе 11 анализируется реактивное программирование с использованием программной модели передачи сообщений. В ней раскрывается концепция естественной изоляции как технологии, дополняющей неизменяемость и позволяющей создавать конкурентные программы. В этой главе основное внимание уделяется классу `MailboxProcessor`, используемому в `F#` для распределения параллельной работы с применением агентного программирования и подхода без разделения ресурсов.
- ❑ В главе 12 описывается агентное программирование с использованием библиотеки `TPL Dataflow` из `.NET` с примерами на `C#`. Здесь показано, как реализовать на `C#` агенты без сохранения состояния и с сохранением состояния, а также как параллельно выполнить несколько вычислений, которые обмениваются данными между собой, применяя (передавая) сообщения в стиле конвейера.

В части III показано, как реализовать на практике все функциональные методы конкурентного программирования, изученные в предыдущих главах.

- ❑ В главе 13 представлен набор полезных рецептов для решения сложных проблем конкурентности, взятых из реальной практики. В этих рецептах используются все функциональные шаблоны, описанные в данной книге.
- ❑ В главе 14 описано полноценное приложение, разработанное и внедренное с применением функциональных конкурентных шаблонов и методов, изученных в этой книге. Вы создадите хорошо масштабируемое, отзывчивое серверное приложение и реактивную клиентскую программу. В книге представлены две версии: одна для `iOS` (`iPad`), созданная с помощью `Xamarin Visual Studio`, и вторая — созданная с помощью `Windows Presentation Foundation (WPF)`. Для обеспечения максимальной масштабируемости в серверном приложении использована комбинация различных моделей программирования, таких как асинхронная, агентская и реактивная.

В книге также содержится три приложения.

- ❑ В приложении А кратко описаны основные понятия функционального программирования, а также представлена базовая теория функциональных методов, использованных в данной книге.
- ❑ В приложении Б раскрываются основные понятия языка F#. Это базовый обзор F#, который позволит вам ближе познакомиться с данным языком и комфортно почувствовать себя в процессе чтения книги.
- ❑ В приложении В наглядно демонстрируется несколько методов, упрощающих взаимодействие между асинхронным рабочим процессом на F# и задачей .NET на C#.

О коде

В этой книге содержится много примеров исходного кода: в виде многочисленных листингов и прямо в тексте. В обоих случаях исходный код выделен **вот таким моноширинным шрифтом**, что позволяет отличать его от обычного текста. Иногда код также выделен жирным шрифтом, чтобы подчеркнуть обсуждаемую тему.

В некоторых местах первоначальный исходный код переформатирован; мы добавили разрывы строк и изменили отступы так, чтобы наилучшим образом разместить код на странице. Но иногда даже этого было недостаточно, и тогда в листинги были вставлены маркеры переноса строки (↵). Кроме того, комментарии в исходном коде часто удалены из листингов, так как описание кода содержится в тексте главы. Многие листинги снабжены аннотациями к коду, в которых описываются важные понятия.

Исходный код листингов, представленных в этой книге, можно загрузить с сайта издательства (www.manning.com/books/concurrency-in-dotnet) и с GitHub (<https://github.com/rikace/fConcBook>). Большая часть кода представлена в двух версиях: на C# и F#. Инструкции по использованию кода содержатся в файле **README**, который находится в корневом каталоге репозитория.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

Об авторе



Рикардо Террелл (Riccardo Terrell) — опытный инженер по программному обеспечению, обладатель статуса Microsoft MVP, увлеченный идеей функционального программирования. Имеет более чем 20-летний опыт разработки экономически эффективных технологических решений в конкурентной экономической среде.

В 1998 г. Рикардо основал в Италии собственный бизнес по разработке программного обеспечения. Он специализировался на индивидуальной разработке медицинского программного обеспечения по требованиям клиентов. В 2007 г. Рикардо переехал в Соединенные Штаты и с тех пор является старшим разработчиком программного обеспечения .NET и старшим архитектором программного обеспечения по предоставлению экономически эффективных технологических решений для бизнеса. Рикардо специализируется на интеграции передовых технологических инструментов для повышения внутренней эффективности, повышения производительности и сокращения издержек.

Он хорошо известен как активный участник сообщества функционального программирования, включая встречи .NET и международные конференции. Рикардо верит в мультипарадигматическое программирование как механизм повышения эффективности кода. Вы можете присоединиться к Рикардо в его исследованиях кода, которые он ведет в своем блоге www.rickyterrell.com.

Об иллюстрации на обложке

На обложке этой книги — человек из деревни в Абиссинии, стране, которую сегодня называют Эфиопией. Иллюстрация взята из испанского альбома с изображениями костюмов разных народов, впервые опубликованного в Мадриде в 1799 г., с гравюрами Мануэля Альбуерне (Manuel Albuerne) (1764–1815). На титульной странице книги написано:

Coleccion general de los Trages que usan actualmente todas las Naciones del Mundo desubierto, dibujados y grabados con la mayor exactitud por R.M.V.A.R. Obra muy util y en special para los que tienen la del viajero universal.

Если перевести это как можно точнее, то получится следующее:

Общая коллекция костюмов, используемых в настоящее время в странах известного мира, смоделированная и напечатанная с большой точностью R.M.V.A.R. Эта работа особенно полезна для тех, кто считает себя универсальным путешественником.

О граверах, дизайнерах и рабочих, которые вручную раскрасили эту иллюстрацию, известно очень мало. Но точность исполнения данного рисунка очевидна. Абиссинец — всего лишь одна из многих фигур, представленных в этой красочной коллекции. Многообразие костюмов ярко свидетельствует об уникальности и индивидуальности городов и стран мира всего 200 лет назад. То было время, когда по одежде можно было однозначно определить принадлежность человека к тому или иному региону, даже если расстояние между этими регионами не превышало нескольких десятков миль. Данный альбом говорит об изолированности и дистанцированности между людьми в то время и в любой другой исторический период, кроме того, в котором живем мы, нашего гиперактивного настоящего.

С тех пор традиции одежды изменились, а ее различия в зависимости от страны, некогда столь значительные, стерлись. Сегодня зачастую трудно отличить жителя одного континента от обитателя другого. Возможно, с оптимистической точки зрения мы отдали культурное и визуальное многообразие в обмен на более богатую личную жизнь — или более разностороннюю и интересную интеллектуальную и техническую жизнь.

Издательство Manning приветствует изобретательность, инициативу и получение удовольствия от компьютерного бизнеса, поэтому снабжает книги обложками с иллюстрациями из данного альбома, свидетельствующими о широком многообразии жизни в разных странах два столетия назад.