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

ПИТЕР®
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



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

# *Краткое содержание*

## Часть III. Современные шаблоны конкурентного программирования



## Приложения

# *Оглавление*

## Часть II. Конкурентная программа: разные части, разные подходы

## Часть III. Современные шаблоны конкурентного программирования

## Приложения

*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 претворить настоящий проект в жизнь. Я люблю тебя и преклоняюсь пред 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 ограниченны. 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 Херцогу (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 Bearl), Павлу Климчику (Pawel Klimczyk), Рикардо Пересу (Ricardo Peres), Рохиту Шарме (Rohit Sharma), Стефано Дрюсси (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 чувствовать 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 Nacionas 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.