Реактивное программирование на Java

Устойчивая архитектура на основе Quarkus

Клеман Эскофье
Кен Финниган

УДК 004.4
ББК 32.973.26-018.2

Э85

Reactive Systems in Java: Resilient, Event-Driven Architecture with Quarkus
Clement Escoffier, Ken Finnigan

Эскофье, Клеман.

Э85 Реактивное программирование на Java. Устойчивая архитектура на основе Quarkus / Клеман Эскофье, Кен Финниган : [перевод с английского М. Райтмана]. — Алматы : Астана иностранная пресса, 2026. — 336 с. — (O’Reilly. Книги по программированию).

ISBN 978-601-12-6021-3

В книге вас ждет передовой подход в разработке приложений: реактивное программирование, реактивные системы, реактивные потоки и др.

Вы узнаете, как Reactive помогает решать задачи современных приложений и интегрироваться с облачными архитектурами.

Для Java-разработчиков, которые хотят научиться создавать надежные распределенные системы, сократить задержки и увеличить пропускную способность приложений, понять основы Quarkus и создавать облачные приложения на базе Kubernetes.

УДК 004.4
ББК 32.973.26-018.2

ISBN 978-601-12-6021-3

Посвящается Деборе, Флор и Нео — тем,
кто покинул нас слишком рано.

Посвящается Эрин, Лоркану и Дайру.

Оглавление

Предисловие 11
Кому будет полезна эта книга 12
А как насчет Quarkus? 12
Структура книги 13
Подготовка к работе 14
Принятые условные обозначения 17
Благодарности 18

Часть I
Введение в Reactive и Quarkus

Глава 1. Пара слов о Reactive 21
Что мы подразумеваем под Reactive? 21
Реактивное ПО — ничего нового 22
Реактивная экосистема 23
Почему реактивная архитектура идеальна для облачных приложений? 27
Reactive — не универсальное решение 27

Глава 2. Введение в Quarkus 29
Облачные Java-технологии 29
Подход Quarkus 40
Создание приложения на Quarkus 43
Kubernetes и Quarkus за 10 минут 50
Сборка в нативный формат 58
Итоги 62

Часть II
Реактивные и событийно-ориентированные приложения

Глава 3. По ту сторону распределенных систем 67
Что такое распределенная система? 67

Новички на сцене: облачные и Kubernetes-ориентированные приложения ... 71
Темная сторона распределенных систем ... 76
Заблуждения распределенных вычислений в мире Kubernetes ... 77
Вопрос времени: недостаток синхронного взаимодействия ... 80
Итоги ... 87

Глава 4. Принципы проектирования реактивных систем ... 88
Основы реактивных систем ... 88
Команды и события ... 91
Точки назначения и пространственное разделение ... 96
Временно́е разделение ... 99
Роль неблокирующего ввода-вывода ... 100
Структура реактивных приложений ... 109
Итоги ... 114

Глава 5. Реактивное программирование: как приручить асинхронность ... 115
Асинхронный код и паттерны ... 115
Future-объекты ... 122
Проект Loom: виртуальные потоки и потоки-носители ... 124
Реактивное программирование ... 127
Стандарт Reactive Streams и необходимость управления потоком ... 135
Итоги ... 142

Часть III
Создание реактивных приложений и систем с помощью Quarkus

Глава 6. Quarkus: реактивный движок ... 147
Императивная модель ... 147
Реактивная модель ... 151
Объединение реактивного и императивного подходов ... 152
Реактивный движок ... 157
Модель реактивного программирования ... 157
Событийно-ориентированная архитектура и Quarkus ... 158
Итоги ... 159

Глава 7. Mutiny: событийно-ориентированный API реактивного программирования ... 161
Зачем нам еще одна библиотека реактивного программирования? ... 161
Чем Mutiny отличается от других? ... 163

Использование Mutiny в Quarkus ... 163
Классы Uni и Multi ... 164
Mutiny и управление потоком данных ... 167
Наблюдение за событиями ... 168
Преобразование событий ... 169
Встраивание асинхронных действий ... 170
Обработка ошибок ... 173
Объединение и комбинирование элементов ... 174
Выбор элементов ... 176
Сбор элементов ... 177
Итоги ... 178

Глава 8. HTTP с учетом реактивности ... 179
Маршрут HTTP-запроса ... 180
Знакомство с RESTEasy Reactive ... 181
В чем же преимущество? ... 184
Асинхронные конечные точки, возвращающие объекты Uni ... 187
Обработка сбоев и настройка ответа ... 190
Потоковая передача данных ... 193
Показатель реактивности ... 200
Итоги ... 201

Глава 9. Реактивная работа с данными ... 203
Проблема доступа к данным ... 203
Неблокирующее взаимодействие с реляционными базами данных ... 205
Использование реактивного ORM: Hibernate Reactive ... 206
А как насчет NoSQL? ... 211
Работа с Redis ... 211
События, связанные с данными, и захват изменений данных ... 215
Платформа Debezium для захвата изменений ... 217
Итоги ... 221

Часть IV
Объединение компонентов

Глава 10. Реактивная передача сообщений: связующее звено ... 225
От реактивных приложений к реактивным системам ... 225
Построение приложений с асинхронной передачей сообщений ... 237
Объединение всех компонентов ... 245
Итоги ... 250

Глава 11. Шина событий: основа системы ... 251
Kafka или AMQP: выбираем подходящий инструмент ... 251
Построение реактивных систем с использованием Kafka ... 252
Создание реактивных систем с использованием AMQP ... 268
Итоги ... 275

Глава 12. Реактивный REST-клиент: подключение к HTTP-ресурсам ... 277
Взаимодействие с HTTP-ресурсами ... 277
Реактивный REST-клиент ... 280
Блокирующий и неблокирующий режимы ... 285
Обработка сбоев ... 286
Создание API-шлюзов с помощью RESTEasy Reactive ... 293
Использование REST-клиента в приложениях обмена сообщениями ... 299
Итоги ... 304

Глава 13. Наблюдаемость в реактивной и событийно-ориентированной архитектуре ... 305
Почему наблюдаемость так важна? ... 305
Проверки состояния в приложениях обмена сообщениями ... 307
Метрики в приложениях обмена сообщениями ... 314
Распределенное трассирование в приложениях обмена сообщениями ... 318
Итоги ... 322

Заключение ... 324
Подведем итоги ... 324
И это все? ... 325
Будущее реактивных систем ... 326
Все еще впереди ... 326

Об авторах ... 327

Послесловие ... 328

Предметный указатель ... 329

Предисловие

В мире информационных технологий границы сегодняшнего дня становятся отправной точкой для завтрашних возможностей. За последние 50 лет ИТ-сфера непрерывно развивалась, без устали расширяя свои пределы. Это происходило не только благодаря техническому прогрессу, но и под влиянием нас — пользователей. Мы продолжаем ожидать все большего от программ, с которыми взаимодействуем каждый день. Более того, изменилась сама форма этого взаимодействия. Сегодня трудно представить жизнь без мобильных приложений и устройств, и мы спокойно воспринимаем уведомления, поступающие в течение всего дня. Интернет вещей (IoT) — это быстрорастущий рынок, сулящий новые волны инноваций, а вместе с ними — рост числа событий и объема обрабатываемых данных, причем непрерывно. Облачные технологии и Kubernetes не только изменили то, как мы используем приложения, но и кардинально повлияли на подход к проектированию, разработке, развертыванию и поддержке программных систем.

Но не стоит обманываться: у всех этих революционных изменений есть цена. С развитием приложений и технологий существенно возросла их сложность. Сегодня большинство программных систем — распределенные. А проектировать, строить и обслуживать распределенные системы сложно, особенно в тех масштабах, которые требуются для современных приложений. Нужно учитывать сбои, асинхронные взаимодействия, постоянно меняющуюся топологию, динамическую доступность ресурсов и многое другое. Хотя облачные технологии открывают доступ практически к неограниченным ресурсам, финансовые ограничения по-прежнему сохраняются. Поэтому повышение плотности развертывания — то есть запуск большего числа приложений на меньшем объеме ресурсов — становится особенно актуальной задачей.

И вот тут возникает вопрос: что такое Reactive? Это не просто библиотека кода и не волшебный фреймворк. Reactive — это совокупность принципов, инструментов, методик и фреймворков, которые помогают строить более эффективные распределенные системы. Насколько более эффективные? Это зависит от конкретной системы, построенные по принципам реактивного программирования, принимают вызовы распределенных сред и фокусируются на эластичности, устойчивости и отзывчивости, как это описано в манифесте Reactive Manifesto (*https://oreil.ly/fO6n0*).

В этой книге мы намеренно используем слово *Reactive* с прописной буквы, чтобы охватить все аспекты реактивного подхода: реактивное программирование,

реактивные системы, реактивные потоки и многое другое. Вы узнаете, как Reactive помогает справляться с вызовами, с которыми сталкиваются современные приложения, и как этот подход органично вписывается в облачные архитектуры. По мере чтения книги вы научитесь создавать реактивные системы — устойчивые, адаптивные, событийно-ориентированные распределенные приложения.

Кому будет полезна эта книга

Эта книга рассчитана на Java-разработчиков среднего и продвинутого уровня. Желательно, чтобы вы чувствовали себя уверенно при работе с Java, однако предварительное знание реактивного программирования или вообще понятия Reactive не требуется. Многие темы книги связаны с распределенными системами, но их знание тоже не обязательно.

Реактивные системы часто опираются на брокеры сообщений, такие как Apache Kafka или AMQP (Advanced Message Queuing Protocol). В книге рассматриваются базовые принципы работы с такими брокерами, чтобы вы поняли, как они используются при проектировании и реализации реактивных систем.

Книга рассчитана на три основные категории читателей:

— разработчиков, создающих облачные приложения и/или распределенные системы;
— архитекторов, которые хотят понять преимущества реактивных и событийно-ориентированных архитектур;
— любознательных разработчиков, которые слышали о Reactive и хотят разобраться, что это такое.

Эта книга поможет вам понять, как проектировать, создавать и внедрять реактивные архитектуры. Вы узнаете не только о том, как с их помощью конструировать более качественные распределенные системы и облачные приложения, но и о том, как можно применять реактивные паттерны для улучшения уже существующих систем.

А как насчет Quarkus?

Внимательный читатель уже заметил упоминание термина Quarkus в подзаголовке книги. Однако до этого момента мы о нем почти не говорили. *Quarkus* — это Java-стек, специально адаптированный для облачных сред. Благодаря подготовке на этапе сборки удается сократить потребление памяти и ускорить запуск приложения.

Но Quarkus — это еще и реактивный стек. В его основе лежит реактивный движок, позволяющий создавать конкурентные и устойчивые приложения. Кроме того, Quarkus предоставляет все необходимые возможности для создания распределенных систем, которые могут адаптироваться к изменяющейся нагрузке и неизбежным сбоям.

На протяжении всей книги мы будем использовать Quarkus, чтобы показать преимущества реактивного подхода и познакомить вас с различными паттернами и лучшими практиками. Если у вас нет предыдущего опыта работы с Quarkus — не переживайте. На каждом этапе вы будете получать понятные и четкие пояснения.

В книге основное внимание уделяется созданию реактивных приложений и систем, использующих возможности Quarkus. Здесь вы найдете всю необходимую информацию для разработки таких систем. Полный обзор всей экосистемы Quarkus мы приводить не будем — сосредоточимся на тех компонентах, которые помогают создавать реактивные системы.

Структура книги

Если вы только начинаете знакомство с реактивным подходом и хотите разобраться, что это такое, чтение книги от начала до конца поможет получить общее представление о Reactive и понять, как этот подход может пригодиться при разработке приложений. А если вы уже опытный разработчик, знакомый с реактивным программированием, и вас в первую очередь интересует Quarkus и его реактивные возможности, можете сразу перейти к тем главам, которые кажутся вам наиболее полезными.

Часть I — краткое введение, формирующее общий контекст.

— В главе 1 приведен краткий обзор реактивной экосистемы, включая ее преимущества и ограничения.
— Глава 2 знакомит с Quarkus и его подходом к сборке, направленным на сокращение времени запуска и уменьшение потребления памяти.

Часть II посвящена реактивному подходу в целом.

- Глава 3 объясняет сложности распределенных систем и типичные заблуждения — именно они являются причинами для перехода к реактивности.
- Глава 4 описывает характеристики реактивных систем.

— Глава 5 рассматривает различные модели асинхронной разработки с акцентом на реактивное программирование.

Часть III повествует о создании реактивных приложений с использованием Quarkus.

— Глава 6 посвящена реактивному движку и объясняет, как совместить императивный и реактивный подходы в программировании.
— Глава 7 — углубленный разбор SmallRye Mutiny, реактивной библиотеки программирования, используемой в Quarkus.
— Глава 8 объясняет особенности обработки HTTP-запросов и показывает, как сделать работу с HTTP реактивной.
— Глава 9 рассказывает, как с помощью Quarkus создавать высококонкурентные и эффективные приложения, работающие с базой данных.

И наконец, часть IV объединяет все элементы и освещает, как конструировать реактивные системы на основе Quarkus.

— В главе 10 подробно рассматривается интеграция приложений Quarkus с системами обмена сообщениями — ключевым элементом реактивных систем.
— Глава 11 посвящена интеграции с Apache Kafka и AMQP и тому, как строить с их помощью реактивные системы.
— Глава 12 посвящена различным способам обращения к HTTP-ресурсам из приложений на Quarkus и объясняет, как обеспечить устойчивость и отзывчивость.
— Глава 13 охватывает аспекты наблюдаемости в реактивных системах, такие как самовосстановление, трассировка и мониторинг.

Подготовка к работе

На протяжении всей книги вы увидите множество примеров кода. Они иллюстрируют рассматриваемые концепции. Некоторые примеры простые и запускаются прямо в среде разработки, другие требуют предварительной подготовки.

Каждый пример будет подробно разобран по мере продвижения по главам. Но если вам не по душе ожидание или, что более вероятно, вы уже устали от наших вступительных рассуждений и просто хотите посмотреть, как все это

работает, переходите по адресу *https://github.com/cescoffier/reactive-systems-in-java* и смело пробуйте примеры в действии.

Репозиторий можно клонировать с помощью Git-команды `git clone https://github.com/cescoffier/reactive-systems-in-java.git`.

Либо скачать ZIP-архив по ссылке *https://oreil.ly/Ey74z* и распаковать его.

Структура кода организована по главам. Например, код, относящийся ко второй главе, размещен в папке *chapter-2* (см. таблицу П-1). В зависимости от главы код может быть разбит на несколько модулей. Для примеров, размещенных в репозитории, в книге в заголовке фрагмента кода указано, где именно находится соответствующий файл.

Таблица П-1. *Ссылки на примеры кода по главам*

Глава	Название	Путь
Глава 2	Введение в Quarkus	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-2*
Глава 3	Темная сторона распределенных систем	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-3*
Глава 4	Принципы проектирования реактивных систем	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-4*
Глава 5	Реактивное программирование: как укротить асинхронность	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-5*
Глава 7	Mutiny: событийно-ориентированный API реактивного программирования	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-7*
Глава 8	HTTP с учетом реактивности	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-8*
Глава 9	Работа с данными в реактивном стиле	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-9*
Глава 10	Реактивная передача сообщений: связующее звено	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-10*
Глава 11	Шина событий: основа системы	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-11*
Глава 12	Реактивный REST-клиент: подключение к HTTP-ресурсам	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-12*
Глава 13	Наблюдаемость в реактивной и событийно-ориентированной архитектуре	*https://github.com/cescoffier/reactive-systems-in-java/tree/master/chapter-13*

Примеры из репозитория кода написаны на языке Java 11, поэтому обязательно установите подходящий комплект для разработки на Java (JDK) на свой компьютер. Для сборки используется инструмент Apache Maven. Устанавливать его вручную необязательно, так как в репозитории используется компонент

Maven Wrapper (*https://oreil.ly/0oKc9*), который автоматически загружает Maven при первой сборке. Если хотите установить Maven вручную, скачайте его с сайта Apache Maven Project (*https://oreil.ly/XgiCr*) и следуйте инструкциям на странице Installing Apache Maven (*https://oreil.ly/nwJV9*).

Чтобы собрать проект, выполните в корневой папке проекта команду `mvn verify`. Инструмент Maven скачает все необходимые зависимости (требуется подключение к интернету).

Книга посвящена работе с Quarkus, стеком Java, ориентированным на Kubernetes. Для использования Quarkus ничего дополнительно устанавливать не нужно — достаточно Java и Maven. Остальное Quarkus скачает автоматически.

Вам понадобится инструмент Docker. Он используется для создания контейнеров с нашими приложениями. Установите Docker, следуя инструкциям на странице Get Docker (*https://oreil.ly/DjBnj*).

Кроме того, в нескольких главах книги описывается, как развертывать реактивные приложения в Kubernetes. Для этого потребуется `kubectl` — командная утилита для взаимодействия с Kubernetes. Ее можно установить согласно инструкции на странице Kubernetes Install Tools (*https://oreil.ly/4SA4J*). Если у вас нет собственного кластера Kubernetes, рекомендуется установить инструмент `minikube`, чтобы настроить локальную среду Kubernetes. Инструкции по установке ищите на сайте *https://oreil.ly/vuCs1*.

Зачем нужны все эти инструменты? Как вы увидите в книге, переход к реактивному подходу накладывает определенные требования не только на само приложение, но и на инфраструктуру. Kubernetes предоставляет основные механизмы для развертывания приложений, создания реплик и поддержания стабильности системы. В свою очередь, Quarkus предоставляет все необходимые средства для реализации реактивных приложений, включая неблокирующий ввод-вывод, реактивное программирование, реактивные API и возможности обработки сообщений. Кроме того, Quarkus интегрируется с Kubernetes, облегчая развертывание и настройку приложений.

В таблице П-2 перечислены инструменты, которые мы будем использовать в этой книге.

Таблица П-2. *Инструменты, описываемые в этой книге*

Инструмент	Сайт	Описание
Java 11	*https://adoptopenjdk.net*	Виртуальная машина Java (JVM) и комплект разработчика (JDK)
Apache Maven	*https://maven.apache.org/download.cgi*	Инструмент автоматизации сборки на основе Project Object Model (POM)
Quarkus	*https://quarkus.io*	Java-стек, оптимизированный для контейнеров и ориентированный на Kubernetes
Docker	*https://www.docker.com/get-started*	Создание и запуск контейнеров
Kubernetes	*https://kubernetes.io*	Платформа оркестрации контейнеров, также известная как K8s
minikube	*https://minikube.sigs.k8s.io/docs/start*	Локальное дистрибутивное решение Kubernetes
GraalVM	*https://www.graalvm.org*	В том числе содержит компилятор для создания нативных исполняемых файлов из Java-кода
Node.js	*https://nodejs.org/en*	Движок выполнения JavaScript

Принятые условные обозначения

В этой книге используются следующие типографские условные обозначения.

Курсив
: Используется для обозначения новых терминов, URL-адресов, адресов электронной почты, имен файлов и расширений файлов.

`Моноширинный шрифт`
: Применяется для оформления листингов программ, а также внутри абзацев обычного текста для обозначения программных элементов, таких как имена переменных, функций и баз данных, типы данных, переменные окружения, операторы и ключевые слова.

`Моноширинный полужирный шрифт`
: Используется для оформления команд и других строк, которые нужно ввести дословно.

`Моноширинный курсивный шрифт`
: Таким образом оформляется текст, который следует заменить на пользовательские значения или значения, определяемые по контексту.