

В память о моем отце, Иване.

– Игорь Лозинский



Март – это цифровая онлайн-библиотека, которая откроет вам полный доступ к более чем 5 тыс. книг и видеороликов, а также к передовым инструментам, которые помогут вам планировать свое личное развитие и продвигаться по карьерной лестнице. За более подробной информацией обращайтесь на наш веб-сайт www.mart.io.

Что дает подписка?

- Электронные книги и видеоуроки практической направленности, созданные более чем 4 тыс. профессионалами в своем деле, помогут вам меньше времени тратить на обучение и больше на программирование.
- Воспользовавшись инструментом Skill Plans, вы сможете составить индивидуальный план своего обучения.
- Станете получать доступ к новым электронным книгам и видеоурокам каждый месяц.
- У вас будет возможность полнотекстового поиска в библиотеке Март.
- Вы сможете копировать и распечатывать содержимое книг, а также оставлять закладки.

www.PacktPub.com

Знаете ли вы, что издательство Packt предлагает электронные версии всех выпускаемых им книг в форматах PDF и ePub? Приобрести электронные версии можно на сайте www.PacktPub.com, а при покупке печатной книги вы получите скидку на ее электронную копию. За дополнительной информацией обращайтесь по адресу: customercare@packtpub.com.

На сайте www.PacktPub.com можно также найти множество технических статей, подписаться на бесплатные новостные письма и получить исключительные скидки на печатные и электронные книги издательства Packt.

Предисловие

Наконец реактивное программирование получило заслуженную поддержку в таких известных в мире Java продуктах, как Spring Boot и Spring Framework. Как бы вы охарактеризовали решения Spring? Многие пользователи обычно отвечают на этот вопрос коротко: прагматичность. Предлагаемая поддержка реактивного программирования не является исключением, и команда решила продолжать оказывать содействие как реактивным, так и неактивным стекам. С выбором приходит ответственность, поэтому важно понимать, когда лучше использовать парадигму «реактивного программирования» и какие передовые методы можно применить при создании своей следующей промышленной системы.

Проект Spring позиционируется как поставщик лучших инструментов для разработки любых видов микросервисов. Поддержка реактивного стека Spring помогает разработчикам создавать невероятно эффективные, высокодоступные и надежные конечные точки. Кроме того, реактивные микросервисы на основе Spring терпимо относятся к задержкам в Сети и в меньшей степени подвержены разрушительному влиянию отказов. Только представьте – это решение можно использовать и для Edge API, и для серверной части мобильного приложения, и для тесно взаимосвязанных микросервисов! Не знали? Реактивные микросервисы изолируют медленные транзакции и вознаграждают быстрые.

Как только вы определитесь с потребностями, Project Reactor станет для вас реактивной основой, естественным образом сочетающейся с вашим реактивным проектом на основе Spring. В последних выпусках 3.x этой библиотеки было реализовано большинство реактивных расширений, описанных компанией Microsoft в 2011 году. Наряду со стандартным словарем библиотека Reactor добавляет великолепную поддержку механизма Reactive Streams управления потоком на каждом функциональном шаге и уникальные возможности, такие как передача контекста.

В своей книге Олег и Игорь на примерах пусть и искусственных, но не упрощенных, описывают фантастическое путешествие по миру реактивного программирования и реактивных систем. После краткого введения в библиотеку Project Reactor и историю ее развития вы быстро погрузитесь в исследование практических примеров на основе Spring Boot 2. Кроме того, авторы не упустили случая всерьез охватить тему тестирования и четко и ясно показать, как создается качественный реактивный код.

Авторы предлагают читателям исчерпывающее введение в шаблоны реактивного проектирования для эффективного масштабирования, отвечающего настоящим и будущим потребностям. Они не просто описывают приемы реактивного программирования, но также дают массу практических советов по использованию Spring Boot и Spring Framework. В главе, посвященной перспективам реактивного программирования, авторы разжигают любопытство читателя, знакомя его с некоторыми подробностями о реактивных взаимодействиях с использованием

RSocket – многообещающей технологии, призванной привнести преимущества реактивного программирования в транспортный уровень.

Надеюсь, вы получите от чтения книги столько же удовольствия, сколько и я, и продолжите исследовать новые способы разработки приложений.

Стефан Мальдини (Stéphane Maldini)
Ведущий разработчик, Project Reactor

Об авторах

Олег Докука (Oleh Dokuka) – опытный инженер-программист, обладатель награды Pivotal Champion и один из основных вкладчиков в развитие Project Reactor и Spring Framework. Он хорошо знает, как устроены оба фреймворка, и ежедневно популяризирует идеи реактивного программирования с использованием Project Reactor. Наряду с этим Олег использует Spring Framework и Project Reactor в разработке программного обеспечения, поэтому он не понаслышке знает, как создавать реактивные системы с применением этих технологий.

Игорь Лозинский (Igor Lozynskiy) – старший Java-разработчик, в основном создающий надежные, масштабируемые и невероятно быстрые системы. Имеет за плечами более чем семилетний опыт работы с платформой Java. Увлекается интересными и динамичными проектами как в своей жизни, так и в разработке программного обеспечения.

Об обозревателях

Николай Алименко (Mikalai Alimenkou) – старший менеджер по работе с корпоративными клиентами, технический лидер Java и опытный преподаватель. Эксперт в разработке на Java, в области масштабируемых архитектур, а также методов гибкой разработки, процессов обеспечения качества и управления проектами. Имеет более чем 14-летний опыт разработки программного обеспечения, специализируется на создании сложных, распределенных и масштабируемых систем и на глубоком преобразовании компаний. Активный участник многих международных конференций. Основатель, тренер и независимый консультант тренинг-центра XP Injection. Организатор и идеолог международных конференций Selenium Camp, JEEConf и XP Days Ukraine, а также клуба анонимных разработчиков Anonymous Developers Club (UADEVCLUB).

Назарий Черкас (Nazarii Cherkas) трудится архитектором решений в Hazelcast – в компании, разрабатывающей проекты с открытым исходным кодом, такие как Hazelcast IMDG и Hazelcast Jet. Имеет многолетний опыт деятельности на разных должностях – от инженера-программиста на Java до руководителя группы разработчиков. Принимал участие в различных проектах для множества отраслей – от телекоммуникации и здравоохранения до критических систем, обслуживающих инфраструктуру одной из крупнейших в мире авиакомпаний. Имеет степень магистра информатики Черновицкого национального университета имени Юрия Федьковича.

Томаш Нуркевич (Tomasz Nurkiewicz) – обладатель титула Java Champion. Потратил на программирование половину жизни. Трудится в сфере электронной коммерции. Занимается разработкой свободного программного обеспечения, является самым ценным блогером DZone и очень активно участвует в работе сайта Stack Overflow. Автор, тренер, докладчик, технический обозреватель и легкоатлет. Считает, что не бывает кода, который не тестируется автоматически. Написал книгу о RxJava.

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 прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма. Если есть тема, в которой вы квалифицированы, и вы 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 удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии 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. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции. Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы. Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Оглавление

Предисловие	7
Глава 1. Причины выбора Spring	22
Основные преимущества реактивности	22
Взаимодействия на основе обмена сообщениями	25
Примеры использования реактивности	30
Причины добавления поддержки реактивности в Spring	33
Реактивность на уровне служб	34
Заключение	42
Глава 2. Реактивное программирование в Spring. Основные понятия	44
Первые реактивные решения в Spring	44
Шаблон «Наблюдатель»	45
Примеры использования шаблона «Наблюдатель»	49
Шаблон «Публикация/Подписка» с использованием @EventListener	52
Создание приложений с @EventListener	54
Создание приложения на основе Spring	54
Реализация бизнес-логики	55
Асинхронные взаимодействия по HTTP с помощью Spring Web MVC	57
Публикация конечной точки SSE	57
Настройка поддержки асинхронного выполнения	59
Создание пользовательского интерфейса с поддержкой SSE	60
Проверка приложения	61
Критический обзор решения	61
RxJava как реактивный фреймворк	62
«Наблюдатель» плюс «Итератор» равно «реактивный поток»	63
Производство и потребление потоков	65
Генерация последовательности асинхронных событий	68
Преобразование потоков и диаграммы Marble	69
Оператор map	69
Оператор filter	70
Оператор count	71
Оператор zip	71
Требования и преимущества RxJava	72
Переделка приложения с RxJava	75
Реализация бизнес-логики	75
Нестандартный SseEmitter	77
Публикация конечной точки SSE	78
Конфигурация приложения	79
Краткая история развития реактивных библиотек	80
Реактивный ландшафт	82
Заключение	84
Глава 3. Reactive Streams – новый стандарт потоков	85
Реактивность для всех	85
Проблема несовместимости API	86

Модели обмена PULL и PUSH	89
Проблема управления потоком данных	95
Медленный производитель и быстрый потребитель	95
Быстрый производитель и медленный потребитель	96
Неограниченная очередь	96
Ограниченная очередь со сбросом избыточных элементов	97
Ограниченная очередь с блокировкой	97
Решение	99
Основные положения стандарта Reactive Streams	99
Требования Reactive Streams в действии	106
Введение в понятие обработчика Processor	109
Проверка совместимости с Reactive Streams	113
Проверка издателя Publisher	115
Проверка подписчика Subscriber	117
JDK 9	121
Асинхронный и параллельный API в Reactive Streams	123
Преобразование реактивного ландшафта	125
Изменения в RxJava	125
Изменения в Vert.x	129
Усовершенствования в Ratpack	130
Драйвер MongoDB с поддержкой Reactive Streams	131
Комбинирование реактивных технологий на практике	132
Заключение	135
Глава 4. Project Reactor – основа реактивных приложений	137
Краткая история Project Reactor	137
Project Reactor 1.x	138
Project Reactor 2.x	141
Основы Project Reactor	142
Добавление библиотеки Reactor в проект	144
Реактивные типы: Flux и Mono	145
Flux	145
Mono	147
Реактивные типы из RxJava 2	148
Observable	148
Flowable	148
Single	148
Maybe	149
Completable	149
Создание последовательностей Flux и Mono	149
Подписка на реактивный поток	151
Реализация своих подписчиков	154
Преобразование реактивных последовательностей с помощью операторов	156
Отображение элементов реактивных последовательностей	157
Фильтрация реактивных последовательностей	158
Сбор данных из реактивных последовательностей	160

Сокращение элементов потока	161
Комбинирование реактивных потоков	164
Пакетная обработка элементов потока	164
Операторы flatMap, concatMap и flatMapSequential	168
Извлечение выборки элементов	170
Преобразование реактивных последовательностей в блокирующие структуры	170
Просмотр элементов при обработке последовательности	171
Материализация и дематериализация сигналов	172
Поиск подходящего оператора	173
Создание потоков данных программным способом	173
Фабричные методы push и create	173
Фабричный метод generate	174
Передача одноразовых ресурсов в реактивные потоки	175
Обертывание транзакций с помощью фабричного метода usingWhen	178
Обработка ошибок	180
Управление обратным давлением	183
Горячие и холодные потоки данных	184
Широковещательная рассылка элементов потока данных	185
Кеширование элементов потока	186
Совместное использование элементов из потока	187
Работа со временем	188
Компоновка и преобразование реактивных потоков	188
Процессоры	190
Тестирование и отладка Project Reactor	191
Дополнения к Reactor	192
Продвинутое средства в Project Reactor	193
Жизненный цикл реактивных потоков данных	193
Этап сборки	193
Этап подписки	195
Выполнение	196
Модель планирования потоков выполнения в Reactor	199
Оператор publishOn	199
Параллельная обработка с помощью publishOn	201
Оператор subscribeOn	202
Оператор parallel	204
Планировщик	204
Контекст	205
Особенности внутренней реализации Project Reactor	209
Макрослияние	210
Микрослияние	211
Заключение	214
Глава 5. Добавление реактивности с помощью Spring Boot 2	216
Быстрый старт как ключ к успеху	217
Использование Spring Roo для ускорения разработки приложений	219
Spring Boot как ключ к созданию быстро растущих приложений	219

Реактивность в Spring Boot 2.0	220
Реактивность в Spring Core	221
Поддержка преобразования реактивных типов	221
Реактивный ввод/вывод	222
Реактивность в Web	224
Реактивность в Spring Data	226
Реактивность в Spring Session	227
Реактивность в Spring Security	228
Реактивность в Spring Cloud	228
Реактивность в Spring Test	229
Реактивность в мониторинге	229
Заключение	230
Глава 6. Неблокирующие и асинхронные взаимодействия с WebFlux	231
WebFlux как основа реактивного сервера	231
Реактивное веб-ядро	234
Реактивные фреймворки Web и MVC	238
Чисто функциональные приемы в WebFlux	242
Неблокирующие взаимодействия между службами с WebClient	246
Реактивный WebSocket API	249
Серверный WebSocket API	250
Клиентский WebSocket API	251
Сравнение WebFlux WebSocket и Spring WebSocket	252
Реактивный поток SSE и легковесная замена WebSocket	253
Реактивные механизмы шаблонов	255
Реактивная безопасность	258
Реактивный доступ к SecurityContext	258
Использование реактивной безопасности	261
Взаимодействия с другими реактивными библиотеками	262
Сравнение WebFlux и Web MVC	263
Законы сравнения фреймворков	264
Закон Литтла	264
Закон Амдала	265
Универсальный закон масштабируемости	269
Анализ и сравнение	272
Модели обработки в WebFlux и Web MVC	272
Влияние моделей обработки на пропускную способность и задержку	274
Проблемы модели обработки в WebFlux	282
Потребление памяти разными моделями обработки	285
Влияние модели обработки на удобство	291
Практическое применение WebFlux	292
Системы на основе микросервисов	292
Системы, обслуживающие клиентов с медленными соединениями	294
Потоковые системы или системы реального времени	294
WebFlux в действии	295
Заключение	299

Глава 7. Реактивный доступ к базам данных	301
Модели обработки данных в современном мире	302
Предметно-ориентированное проектирование	302
Хранение данных в эпоху микросервисов	303
Использование хранилищ разного типа	306
База данных как услуга	307
Разделение данных между микросервисами	309
Распределенные транзакции	310
Событийно-ориентированные архитектуры	310
Согласованность в конечном счете	311
Шаблон SAGA	312
Регистрация событий	312
Разделение ответственности на команды и запросы	313
Бесконфликтно реплицируемые типы данных	314
Система обмена сообщениями как хранилище данных	315
Синхронная модель извлечения данных	316
Протокол связи для доступа к базе данных	316
Драйвер базы данных	318
JDBC	319
Управление соединениями	320
Реактивный доступ к базе данных	321
Spring JDBC	322
Spring Data JDBC	323
Добавление реактивности в Spring Data JDBC	326
JPA	326
Добавление реактивности в JPA	327
Spring Data JPA	327
Добавление реактивности в Spring Data JPA	328
Spring Data NoSQL	329
Ограничения синхронной модели	332
Достоинства синхронной модели	333
Реактивный доступ к данным с использованием Spring Data	334
Реактивное хранилище на основе MongoDB	336
Объединение операций с хранилищем	339
Как работают реактивные хранилища	344
Поддержка разбиения на страницы	345
Детали реализации ReactiveMongoRepository	345
Использование ReactiveMongoTemplate	346
Использование реактивных драйверов (MongoDB)	348
Использование асинхронных драйверов (Cassandra)	350
Реактивные транзакции	352
Реактивные транзакции в MongoDB 4	352
Распределенные транзакции с шаблоном SAGA	361
Реактивные коннекторы в Spring Data	361
Реактивный коннектор MongoDB	361
Реактивный коннектор Cassandra	362

Реактивный коннектор Couchbase	362
Реактивный коннектор Redis	363
Ограничения и ожидаемые улучшения	364
Асинхронный доступ к базам данных	365
Реактивное соединение с реляционной базой данных	367
Использование R2DBC вместе с Spring Data R2DBC	369
Преобразование синхронного хранилища в реактивное	370
С помощью библиотеки gxjava2-jdbc	371
Обертывание синхронного CrudRepository	373
Реактивный Spring Data в действии	378
Заключение	382
Глава 8. Масштабирование с Cloud Streams	383
Брокеры сообщений как основа систем, управляемых сообщениями	384
Балансировка нагрузки на стороне сервера	384
Балансировка нагрузки на стороне клиента с Spring Cloud и Ribbon	386
Брокеры сообщений как эластичный и надежный слой для передачи сообщений	392
Рынок брокеров сообщений	396
Spring Cloud Streams как мост в экосистему Spring	397
Реактивное программирование в облаке	406
Spring Cloud Data Flow	407
Модульная организация приложений с Spring Cloud Function	409
Spring Cloud – функция как часть конвейера обработки данных	416
RSocket для реактивной передачи сообщений с низкой задержкой	420
RSocket и Reactor-Netty	421
RSocket в Java	425
RSocket и gRPC	428
RSocket в Spring Framework	430
RSocket в других фреймворках	432
Проект ScaleCube	432
Проект Proteus	433
В заключение о RSocket	433
Заключение	433
Глава 9. Тестирование реактивных приложений	435
Почему реактивные потоки данных сложно тестировать?	435
Тестирование реактивных потоков с помощью StepVerifier	436
Основы StepVerifier	436
Продвинутые приемы тестирования с использованием StepVerifier	440
Виртуальное время	442
Проверка реактивного контекста	445
Тестирование WebFlux	445
Тестирование контроллеров с помощью WebTestClient	446
Тестирование WebSocket	451

Заключение	455
Глава 10. И наконец, выпуск!	456
Важность поддержки идеологии DevOps в приложениях	456
Мониторинг реактивных Spring-приложений	460
Spring Boot Actuator	460
Добавление механизма мониторинга в проект	460
Конечная точка для получения информации о службе	461
Конечная точка для получения информации о работоспособности	463
Конечная точка для получения информации о параметрах работы	466
Конечная точка управления журналированием	467
Другие важные конечные точки	468
Реализация своей конечной точки для Actuator	469
Безопасность конечных точек	470
Micrometer	472
Параметры по умолчанию в Spring Boot	473
Мониторинг реактивных потоков данных	474
Мониторинг потоков в Reactor	474
Мониторинг планировщиков в Reactor	475
Реализация своих параметров Micrometer	477
Распределенная трассировка с Spring Boot Sleuth	478
Пользовательский интерфейс Spring Boot Admin 2.x	480
Развертывание в облаке	482
Развертывание в Amazon Web Services	485
Развертывание в Google Kubernetes Engine	486
Развертывание в Pivotal Cloud Foundry	487
Обнаружение RabbitMQ в PCF	488
Обнаружение MongoDB в PCF	489
Развертывание в PCF без конфигурации с помощью Spring Cloud Data Flow	491
Knative для FaaS на основе Kubernetes и Istio	491
Советы по успешному развертыванию приложений	492
Заключение	493
Указатель	495

Вступление

Реактивные системы всегда откликаются, что и требуется большинству предприятий. Разработка таких систем – сложная задача, требующая глубокого понимания предмета. К счастью, разработчики Spring Framework создали новую, реактивную версию проекта.

В книге «Практика реактивного программирования в Spring 5» вы познакомитесь с увлекательным процессом разработки реактивных систем на основе фреймворка Spring Framework 5.

Книга начинается со знакомства с основами реактивного программирования в Spring. Вы получите представление о возможностях фреймворка и узнаете об основах реактивного программирования. Далее вашему вниманию будут представлены методы реактивного программирования, способы их использования для организации взаимодействия баз данных и серверов. Все эти задачи продемонстрированы на примере реального проекта, что позволит вам попрактиковаться в применении полученных навыков.

А теперь просим всех на борт реактивной революции в Spring 5!

Кому адресована эта книга

Книга адресована разработчикам на Java, использующим фреймворк Spring для своих приложений и желающим получить возможность создавать надежные и реактивные приложения, которые можно масштабировать в облаке. Предполагается базовое знание распределенных систем и асинхронного программирования.

Содержание книги

Глава 1 «*Причины выбора Spring*» описывает случаи, для которых подходит реактивность. Здесь вы узнаете, чем реактивное решение лучше проактивного, познакомитесь с несколькими примерами кода, демонстрирующими разные способы связи между серверами, а также с потребностями и требованиями бизнеса к современному Spring Framework.

Глава 2 «*Реактивное программирование в Spring – основные понятия*» раскрывает потенциал реактивного программирования, знакомит с основными понятиями на примерах кода. Она демонстрирует мощь реактивного, асинхронного, неблокирующего программирования в Spring Framework на примерах кода и как оно применяется на практике. Здесь вы познакомитесь с моделью «издатель-подписчик», с реактивными событиями Flow и особенностями применения этих методов на практике.

Глава 3 «*Reactive Streams – новый стандарт потоков*» описывает проблемы реактивных расширений Reactive Extensions. На примерах кода раскрывается природа проблем, исследуются разные подходы к их решению. Глава также знакомит со спецификацией Reactive Streams, которая вводит новые компоненты в хорошо известную модель «издатель-подписчик».

Глава 4 «*Project Reactor – основа реактивных приложений*» рассматривает реализацию реактивной библиотеки, полностью осуществляющей спецификацию Reactive Streams. Сначала описываются преимущества Reactor, а затем рассматриваются причины, побудившие разработчиков Spring заняться созданием нового решения. Кроме того, глава знакомит с основами использования данной впечатляющей библиотеки – здесь вы получите представление о Mono и Flux, а также об особенностях применения реактивных типов.

Глава 5 «*Добавление реактивности с помощью Spring Boot 2*» знакомит с реактивными модулями, имеющимися в Spring, которые пригодятся при разработке реактивных приложений. Здесь вы узнаете, как начать использовать модули и как Spring Boot 2 помогает разработчикам быстро настраивать приложения.

Глава 6 «*Неблокирующие и асинхронные взаимодействия с WebFlux*» описывает один из основных модулей – Spring WebFlux, являющийся важным инструментом для организации асинхронного и неблокирующего взаимодействия пользователя и внешних служб. В ней дается обзор преимуществ этого модуля и сравнение его с Spring MVC.

Глава 7 «*Реактивный доступ к базам данных*» описывает модель реактивного программирования доступа к данным на основе Spring 5. Основное внимание уделяется поддержке реактивности в модулях Spring Data, исследуются средства, входящие в состав Spring 5, Reactive Streams и Project Reactor. Здесь представлен код, демонстрирующий реактивный подход к взаимодействию разных баз данных, таких как SQL и NoSQL.

Глава 8 «*Масштабирование с Cloud Streams*» познакомит с реактивными возможностями Spring Cloud Streams. Для начала вашему вниманию будет предложен обзор проблем и недостатков, с которыми можно столкнуться при организации масштабирования на разных серверах. Глава раскроет возможности решения Spring Cloud и продемонстрирует его реализацию на примерах кода с соответствующей конфигурацией Spring Boot 2.

Глава 9 «*Тестирование реактивных приложений*» описывает основы тестирования реактивных приложений. Вы познакомитесь с модулями Spring 5 Test и Project Reactor Test, узнаете, как управлять частотой следования событий, перемещать временные интервалы, расширять пулы потоков выполнения, проверять результаты и переданные сообщения.

Глава 10 «И наконец, выпуск!» содержит подробное пошаговое руководство по развертыванию и мониторингу решения. Наглядно показано, как осуществляется мониторинг реактивных микросервисов с использованием модулей Spring 5. Также в этой главе рассматриваются инструменты, которые пригодятся для сбора данных мониторинга и их отображения.

Что потребуется для работы с книгой

Разработка реактивных систем – сложная задача, требующая глубокого понимания предметной области, поэтому вам потребуется познакомиться с распределенными системами и асинхронным программированием.

Загрузка исходного кода примеров

Загрузить файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.dmk.ru в разделе «Читателям – Файлы к книгам».

Кроме того, примеры кода к книге доступны на сайте GitHub: <https://github.com/PacktPublishing/Hands-On-Reactive-Programming-in-Spring-5>.

Загрузка цветных иллюстраций

Мы также подготовили документ PDF, содержащий цветные иллюстрации со скриншотами и диаграммами, используемыми в книге. Его можно взять здесь: https://www.packtpub.com/sites/default/files/downloads/9781787284951_ColorImages.pdf.

Типографские соглашения

В книге используется несколько разных стилей оформления текста с целью обеспечить визуальное отличие информации разных типов. Ниже приводится несколько примеров стилей оформления и краткое описание их назначения.

Программный код в тексте, имена таблиц баз данных, имена папок, имена файлов, расширения файлов, пути в файловой системе, адреса URL, ввод пользователя и ссылки в Twitter оформляются так, как показано в следующем предложении: «При первом обращении будет вызван обработчик `onSubscribe()`, который сохранит подписку `Subscription` локально и известит издателя `Publisher` о готовности подписчика принимать события через метод `request()`».

Блоки программного кода оформляются следующим образом:

```
@Override
public long maxElementsFromPublisher() {
    return 1;
}
```

Ввод или вывод в командной строке оформляются так:

```
./gradlew clean build
```

Новые термины и важные определения выделяются в тексте полужирным.



Так оформляются предупреждения и важные примечания.



Так оформляются советы и рекомендации.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите, 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 полезными.

Вы можете написать отзыв прямо 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 есть тема, в которой у вас высокая квалификация, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте 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 удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг (в тексте или в коде), мы будем благодарны, если вы сообщите нам о ней. Этим вы поможете улучшить последующие версии книги.

Если найдете ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу: dmkpress@gmail.com, и мы исправим их в следующих изданиях.

Нарушение авторских прав

Пиратство в интернете по-прежнему насущная проблема. Издательства «ДМК Пресс» и Раскт очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли принять меры.

Пожалуйста, свяжитесь с нами по адресу dmkpress@gmail.com и пришлите ссылки на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, способствующую предоставлению качественных материалов.

Глава 1

Причины выбора Spring

В этой главе мы объясним понятие **реактивности** и расскажем, почему реактивные подходы лучше традиционных, для чего рассмотрим примеры, когда традиционные подходы терпят неудачу. Затем исследуем фундаментальные принципы построения надежных систем, которые в большинстве своем являются **реактивными системами**. Узнаем, каковы основные причины, объясняющие необходимость использования механизмов рассылки сообщений для организации взаимодействий между распределенными серверами. Мы покажем, в какие случаи реактивность вписывается как нельзя лучше, расскажем о приемах **реактивного программирования** для создания модульной реактивной системы. Кроме того, обсудим, почему команда разработчиков Spring Framework решила включить реактивный подход в ядро фреймворка **Spring Framework 5**. Прочитав главу, вы поймете важность реактивности и то, почему стоит перенести свои проекты в реактивный мир.

Здесь рассматриваются следующие темы:

- основные преимущества реактивности;
- основные принципы создания реактивных систем;
- случаи, когда реактивный дизайн подходит лучше всего;
- приемы программирования реактивных систем;
- причины включения поддержки реактивности в Spring Framework.

Основные преимущества реактивности

В наши дни стало модным использовать слово **реактивный** – оно такое волнующее и непонятное. Однако стоит ли продолжать популяризовать реактивность, даже после того как слово заняло почетное место на разнообразных международных конференциях? Если забьем в поиск слово «реактивный», то обнаружим, что чаще всего оно встречается в паре со словом «программирование», и вместе они обозначают модель программирования. Однако это не единственный смысл реак-

тивности. За данным словом стоят фундаментальные принципы проектирования, направленные на создание надежных систем. Чтобы понять ценность реактивности как важнейшего принципа проектирования, представим, что мы развиваем малое предприятие.

Допустим, наше малое предприятие – это интернет-магазин, продающий современные товары по привлекательным ценам. Как большинство владельцев подобных магазинов, мы также наняли разработчиков программного обеспечения, которые помогут нам справиться с проблемами. Мы выбрали традиционный подход к разработке программного обеспечения и создали магазин.

Каждый час наш онлайн-магазин обычно посещает 1 тыс. пользователей. Чтобы справиться с потоком покупателей, мы купили современный компьютер и запустили на нем веб-сервер Tomcat, настроив пул с 500 предварительно созданными потоками выполнения. Среднее время отклика на большинство запросов – около 250 мс. Простейшие расчеты показывают, что такая конфигурация позволит нам обслуживать до 2 тыс. запросов в секунду. Согласно статистике, вышеупомянутое число пользователей в среднем производит около 1 тыс. запросов в секунду. Следовательно, текущей производительности системы вполне достаточно для обслуживания средней нагрузки.

Итак, мы настроили приложение с неплохим запасом производительности. Более того, наш интернет-магазин работал вполне стабильно до... последней пятницы ноября, то есть до Черной пятницы.

Черная пятница – важный день и для покупателей, и для продавцов. Покупатели получают возможность купить товар со скидкой, а продавцы – получить дополнительную прибыль. Однако этот день характеризуется необычным наплывом клиентов, что может стать причиной сбоев в работе интернет-магазина.

И конечно же, мы потерпели сокрушительное фиаско! В какой-то момент нагрузка превысила все наши ожидания. В пуле не оказалось свободных потоков выполнения для обработки запросов. Сервер резервного копирования не справился с наплывом покупателей, время отклика возросло, периодически наблюдались сбои. Мы начали терять некоторые запросы, в результате наши клиенты, неудовлетворенные долгим обслуживанием, переметнулись к конкурентам.

В итоге мы потеряли большое число клиентов, остались без дополнительной прибыли, а рейтинг магазина рухнул. Все это стало результатом увеличения времени отклика в условиях возросшей нагрузки.

Но не волнуйтесь, это далеко не новая проблема. Даже такие гиганты, как Amazon и Walmart, сталкивались с ней и давно нашли способы ее решения. Но не будем спешить и пройдем тот же путь, каким следовали наши предшественники, чтобы понять основные принципы проектирования надежных систем и дать им общее определение.



Узнать больше о проблемах магазинов-гигантов можно на следующих сайтах:

- Amazon.com. Проблема с отключениями (<https://www.cnet.com/news/amazon-com-hit-with-outages/>);
- Amazon.com. Как отказы приводили к потерям до 66 240 долларов в минуту (<https://www.forbes.com/sites/kellyclay/2013/08/19/amazon-com-goes-down-loses-66240-per-minute/#3fd8db37495c>);
- Walmart. Провал в Черную пятницу: веб-сайт не справился с нагрузкой (<https://techcrunch.com/2011/11/25/walmart-black-friday/>).

Теперь главный вопрос: что с этим делать? Из примера выше следует, что приложение должно как-то реагировать на изменение нагрузки и доступности внешних служб. Иначе говоря, оно должно активно реагировать на любые изменения, которые могут ухудшить доступность системы и ее способность откликаться на запросы пользователей.

Один из путей к главной цели – увеличение **эластичности**. Под этим термином понимается способность сохранять отзывчивость при различной рабочей нагрузке, то есть пропускная способность системы должна автоматически увеличиваться с ростом числа пользователей и уменьшаться – со снижением спроса. Эта особенность улучшает отзывчивость системы, потому что в любой момент пропускная способность системы может вырасти и обеспечить приемлемое среднее время задержки.



Время задержки – важная характеристика отзывчивости. При отсутствии должной эластичности из-за роста нагрузки увеличится время задержки, которое напрямую влияет на отзывчивость системы.

Например, увеличить пропускную способность системы можно, расширяя вычислительные мощности или запуская дополнительные экземпляры. В результате возрастет отзывчивость системы. С другой стороны, если поток пользователей уменьшился, система в ответ должна снизить потребление ресурсов, сократив тем самым накладные расходы. Добиться желаемой эластичности можно путем масштабирования – горизонтального или вертикального. Однако масштабирование распределенной системы – сложная задача. Обычно ограничиваются узкими местами или точками синхронизации в системе. С теоретической и практической точек зрения эти проблемы объясняются законом Амдала и универсальной моделью масштабирования Гюнтера Нейла. Мы обсудим их позже – в главе 6 «*Неблокирующие и асинхронные взаимодействия с WebFlux*».



Здесь под накладными расходами понимается стоимость развертывания новых экземпляров в облаке или дополнительное потребление электроэнергии в случае установки добавочных компьютеров.

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru