УДК 004.65; 004.738.5; 519.6

ПРИМЕНЕНИЕ РЕАКТИВНОГО ПРОГРАММИРОВАНИЯ В МИКРОСЕРВИСНОЙ АРХИТЕКТУРЕ ДЛЯ ВЕБ-ПРИЛОЖЕНИЙ С ЭКОСИСТЕМОЙ SPRING FRAMEWORK

И. М. Примова

ИООО «ЭПАМ Системз», Беларусь, Минск, mmf.inna@gmail.com

Рассмотрена проблема применения реактивного программирования в микросервисной архитектуре для веб-приложений с экосистемой Spring Framework. В связи с необходимостью создания высокопроизводительных и масштабируемых систем, способных обеспечивать стабильность работы при повышенной нагрузке, применение реактивного подхода становится все более актуальным. В данном контексте иллюстрации применения реактивного программирования для решения конкретных задач, а также выявление основных принципов и преимуществ этого подхода в микросервисной архитектуре могут служить важными дидактическими факторами для более глубокого понимания и принятия этого подхода.

Ключевые слова: микросервисная архитектура; реактивное программирование; Spring Framework; RxJava; Project Reactor; Spring WebFlux; Apache Kafka; EDA.

APPLICATION OF REACTIVE PROGRAMMING IN MICROSERVICE ARCHITECTURE FOR SPRING FRAMEWORK ECOSYSTEM WEB APPLICATIONS

I. M. Primova

EPAM Systems, Belarus, Minsk, mmf.inna@gmail.com

The problem of applying reactive programming in microservice architecture for Spring Framework ecosystem web applications has been considered. The application of the reactive approach is becoming increasingly relevant due to the need to create high-performance and scalable systems capable of maintaining stable operation under increased load. In this context, illustrations of the application of reactive programming to solve specific problems, as well as identifying the main principles and advantages of this approach in microservice architecture, can serve as important factors for deeper understanding and adoption of this approach.

Keywords: microservice architecture; reactive programming; Spring Framework; RxJava; Project Reactor; Spring WebFlux; Apache Kafka; EDA.

Введение

Разработка приложений в экосистеме Spring Framework на основе микросервисной архитектуры является популярным подходом для создания распределенных систем. Однако, при таком подходе возникают проблемы с высокой производительностью, устойчивостью и масштабируемостью системы, особенно при возрастающей нагрузке. В этом контексте применение реактивного программирования становится важным инструментом для решения данных проблем и обеспечения эффективной работы микросервисной архитектуры.

Актуальность проблемы заключается в том, что все больше организаций переходят на микросервисную архитектуру для создания высокопроизводительных и масштабируемых систем, которые могут обеспечивать стабильность работы при повышенной нагрузке. Понимание принципов реактивного программирования в этом контексте является важным фактором для успешной разработки и сопровождения микросервисов. Также они упрощают работу с потоками данных, повышают гибкость и адаптивность системы к изменяющимся условиям.

Определение понятия реактивного программирования

Для реализации микросервисов часто используют **событийно-ориентированную архитектуру (Event-Driven Architecture, EDA)**. EDA — это паттерн, который позволяет создавать слабосвязанные и масштабируемые микросервисы. Основная идея его заключается в том, что вместо того, чтобы полагаться на синхронную связь между сервисами, каждый сервис публикует события в брокере сообщений или очереди сообщений, на которые могут подписаться другие сервисы. В качестве брокера сообщений может выступать, к примеру Apache Kafka.

Соответственно для повышения общей производительности и масштабируемости микросервисов в сочетании с Kafka применяют реактивное программирование. Реактивное программирование — это парадигма программирования, которая использует асинхронное, неблокирующее взаимодействие между компонентами и методы функционального программирования, для создания систем, способных обрабатывать большое количество запросов и отвечать на них в режиме реального времени.

В основе реактивного программирования лежит **шаблон «Публикация/Подписка»**, который является разновидностью шаблона «Наблюдатель». Шаблон «Наблюдатель» состоит из субъекта (Subject), который хранит список своих наблюдателей (Observer). Субъект уведомляет

наблюдателей обо всех изменениях, которое произошли в его состоянии, вызывая один из их методов. Отличием шаблона «Публикация/Подписка» от «Наблюдатель» являются то, что в первом издатели и подписчики ничего не знают друг о друге (рисунок 1).

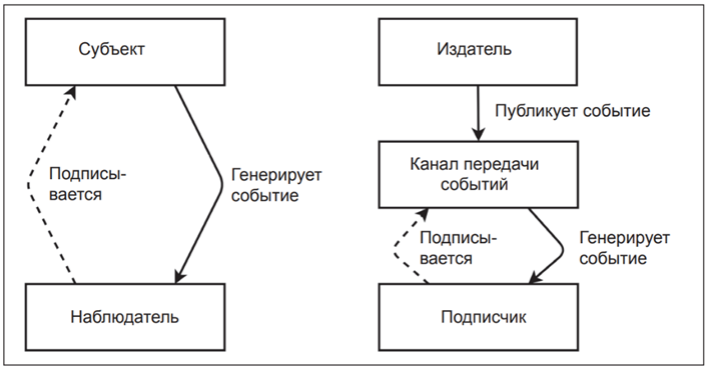

Рис. 1. Шаблон «Наблюдатель» (слева) и шаблон «Публикация/Подписка» (справа)

Реализация концепции реактивного программирования в экосистеме Spring Framework

В мире JVM некоторое время назад появилась библиотека поддержки реактивного программирования — RxJava, которая представляет собой реализацию реактивных расширений (Reactive Extensions/ReactiveX) в Java. ReactiveX — это набор инструментов, которые позволяют императивным языкам работать с потоками данных независимо от того, какая у них природа: асинхронная или синхронная. Reactive Extensions часто определяется как комбинация шаблонов «Наблюдатель», «Итератор» и функционального программирования. В RxJava интерфейс Observer является комбинацией Observer из «Наблюдателя» и «Итератора». Аналогом Subject из «Наблюдателя» выступает Observable, который играет роль источника событий и содержит много методов преобразования потока данных.

Однако кроме RxJava существует много других реализаций реактивных библиотек. В результате чего появился многообразный реактивный ландшафт. Это многообразие обусловило разработку реактивного **стандарта Reactive Streams**, чья идея заключается в стандартизации шаблонов реактивного программирования. Стандарт использует гибридную модель PUSH-PULL потоков, которая поддерживает возможность управления обратным давлением. **Обратное давление** — это обратная связь, которая согласовывает скорость работы производителя со скоростью работы потребителя: потребитель запрашивает данные, когда готов

их обработать, то есть он контролирует, сколько данных должно быть извлечено или когда подписка должна быть отменена.

В экосистеме Spring Framework широкое распространение получила **библиотека Project Reactor**, которая является реализаций Reactive Streams. Она избавляет разработчиков от глубоко вложенного кода при создании асинхронных пайплайнов и ада обратных вызовов. Её цель — повышение удобочитаемости кода и обеспечение возможности компоновать рабочие процессы. Одной из центральных задач библиотеки является управление обратным давлением. Данные передаются от издателя к подписчику. В обратном направлении (от подписчика к издателю) распространяются сигналы управления (рисунок 2).

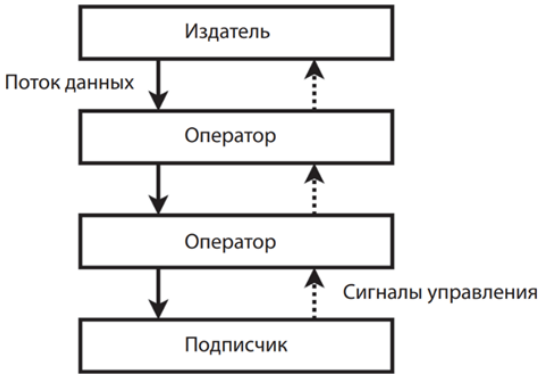

Рис. 2. Передача данных и сигналов управления в реактивном потоке

Библиотека включает в себя концепции реактивного программирования, такие как Flux и Mono. Flux используется для обработки нескольких потоков данных и может вернуть 0, 1 или несколько элементов и даже бесконечное число элементов. Mono предназначен для обработки одного потока данных, который может произвести не более одного элемента. Вызов методов (операторов) Flux или Mono не вызывает немедленного поведения. Вместо этого возвращается новый экземпляр Flux или Mono, на основе которого можно продолжать составлять дальнейшие операторы. Таким образом, создается цепочка операторов (или ациклический граф операторов), которая представляет собой конвейер асинхронной обработки.

В дополнении библиотека Project Reactor является основой для **модуля Spring WebFlux**, который поддерживает асинхронную, неблокирующею коммуникацию в веб-приложениях. Альтернативным подходом является Spring Web MVC, который реализует концепцию Servlet API и контейнер сервлетов. Spring WebFlux поддерживает аннотированные контроллеры, функциональные конечные точки, WebClient — неблоки-

рующий, реактивный HTTP-клиент, который предоставляет упрощенный API для выполнения HTTP-запросов к внешним сервисам реактивным способом. Далее в таблице представлено сравнение реактивного стека и стека сервлетов.

Сравнение реактивного стека и стека сервлетов

Параметр	Реактивный стек	Стек сервлетов
Модули Spring	Spring WebFlux, Spring Security Reactive	Spring MVC, Spring Security
Поддерживаемые репозитории	Spring Data Reactive Repositories Mongo, Cassandra, Reds, Couchbase	Spring Data Repositories JDBC, JPA, NoSQL
Модель программирования	Асинхронная, неблокирующая, блокирующая	Синхронная, блокирующая
API, поддерживаемые среды	Reactive Streams Adapters, Netty или Undertow	Servlet API, Java Servlet Containers, Java EE Application Servers

Заключение

Таким образом, применение в веб-приложениях на базе Spring Framework реактивного программирования по средствам реализации Project Reactor, Spring WebFlux, обеспечивает высокую производительность, устойчивость и масштабируемость системы, особенно при высокой нагрузке. Это достигается благодаря использованию асинхронных и неблокирующих операций в обработке запросов и работы с данными. Кроме того, позволяет использовать механизмы обратного давления, чтобы не перегружать сервисы. Также Spring WebFlux поддерживает широкий спектр протоколов: HTTP, WebSocket, Server-Sent Events, что делает его гибким вариантом для создания микросервисов и использует небольшое количество потоков для обработки большого количества запросов, что позволяет снизить потребление ресурсов и повысить производительность.

Библиографические ссылки

1. Практика реактивного программирования в Spring 5 / Олег Докука, Игорь Лозинский. — М.: ДМК Пресс, 2019. — 508 с.

2. Medium. Kirill Sereda [Электронный ресурс]. — Режим доступа: https://medium.com/@kirill.sereda/reactive-programming-reactor-%D0%B8-spring-webflux-3f779953ed45 — Дата доступа: 18.03.2023.

3. Microservice Architecture [Electronic resource]. — Mode of access: https://microservices.io/ — Date of access: 14.03.2023. …