Asynchronous with EventQueue or MessageQueue in Spring boot

FOGUE KAMGA DILANE – SOFTWARE ENGINEER

AGENDA

- ❑ Concept Study
- ❑ Event-Driven Architecture
- ❑ Project Overview
- ❑ Install and Setup Apache Kafka
- ❑ Create OrderService Microservice
- ❑ Create StockService Microservice
- ❑ Create EmailService Microservice
- ❑ Create BaseDomains Microservice
- ❑ Test the 3 microservices together with Postman
- ❑ Conclusion

Concept Study

Asynchronous communication, mostly know as Event-driven architecture (EDA) is a software design pattern in which decoupled applications can asynchronously publish and subscribe to events via an event broker/message broker.

In an Event-Driven Architecture, applications communicate with each other by sending and/or receiving events or messages.

Asynchronous communication means that the sender and recipient don’t have to wait for each other to move onto their next task. Systems are not dependent on that one message.

An asynchronous example would be text messaging. You send a message and in some cases, you don't even know who you are sending it to or if anyone’s listening, but you are not waiting for a response.

Event-Driven Architecture

❑ Event

An event is defined as a change of state of some key business system. For instance, somebody buys a product, someone else checks in for a flight or a bus is late arriving somewhere. And if one thinks about it, events exist everywhere and are constantly happening, no matter what industry

❑ Advantages Of Event-Driven Architecture

An Event- Driven Architecture has many advantages such as :

- ➢ **Improves Flexibility And Maintainability**
- ➢ **High Scalability**
- ➢ **Improves Availability**

Event-Driven Architecture

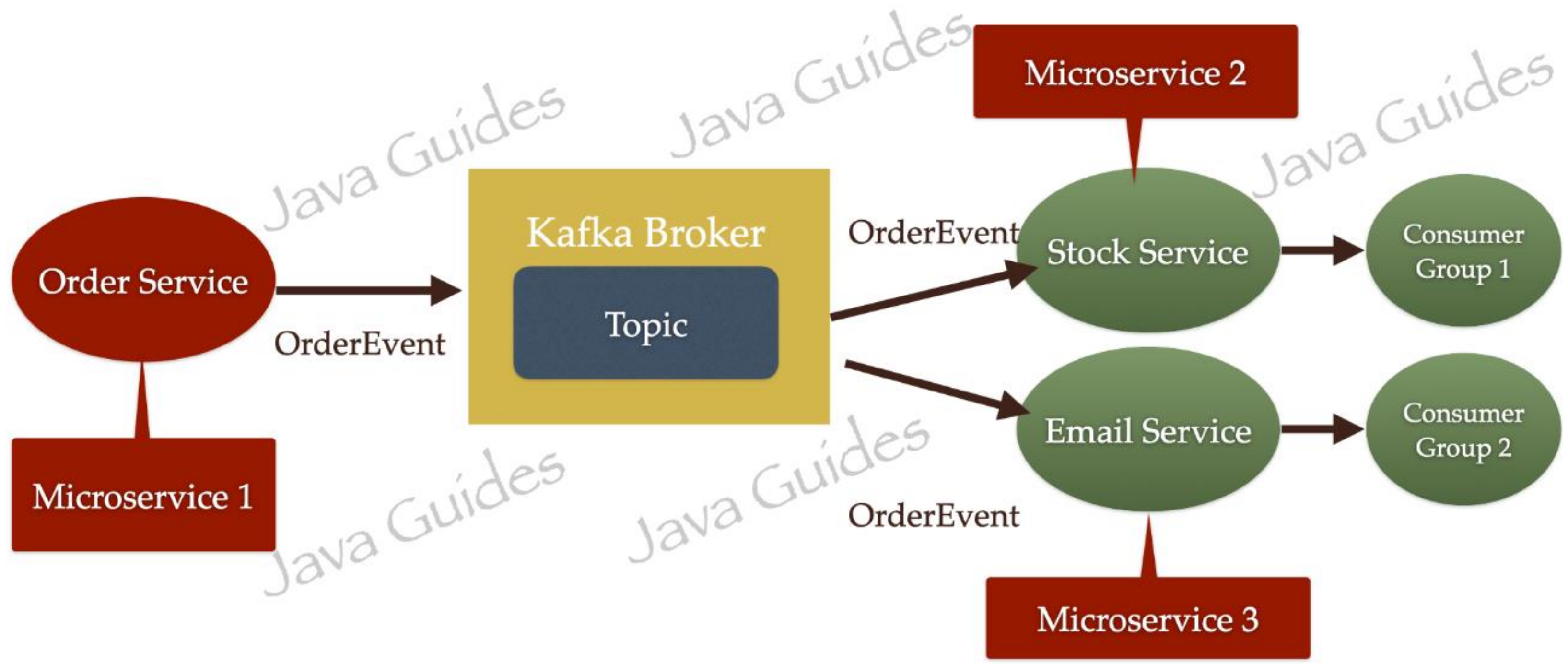
An event-driven architecture is loosely coupled.

Event-Driven Architecture is widely used in Microservices applications.

In Event-Driven Microservices Architecture, multiple microservices use Message Broker for “asynchronous” communication between them. The Message Broker can be a RabbitMQ or Apache Kafka or ActiveMQ etc.

For this presentation, we are going to use Apache Kafka as the Message Broker

Project Overview



We are going to implement the above architecture. In the above architecture, OrderService, StockService, and EmailService microservices are independent of each other. OrderService is a Producer application that sends an event to the Message Broker. StockService and EmailService are Consumers who will consume the events from the Message Broker. We will also see how multiple consumers will subscribe to a single Kafka topic to consume the events/messages.

Install and Setup Apache Kafka

Since, we are windows users. We will install and setup Apache Kafka as follow

- **Download Kafka from the official website at https://kafka.apache.org/downloads**
- **Extract Kafka zip in the local file system**

Run the following commands in order to start all services in the correct order:

- **Start Zookeeper service by using the command below**

```
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

- **Start Kafka Broker by using the command below**

```
.\bin\windows\kafka-server-start.bat .\config\server.properties
```

Create OrderService Microservice

We have decided to implement the OrderService Microservice. First we generate the project with the web dependency as well as the Kafka dependency. Then we proceed as follow

- **Configure Kafka Producer into the application.properties file.**

- **Configure Kafka Topic. We create a config package. Within the config package, create a class named KafkaTopicConfig.**

- **Create Kafka Producer. We create a kafka package. Within the kafka package, create a class named OrderProducer.**

- **Create REST API to Send Order and Test Kafka Producer. We create a controller package. Within the kafka package, create a class named OrderController.**

Create StockService Microservice

We have decided to implement the StockService Microservice. First we generate the project with the web dependency as well as the Kafka dependency. Then we proceed as follow

- **Change server port.** Set it as 8081 into the application.properties file.
- **Configure Kafka Consumer** into the application.properties file.
- **Create Kafka Consumer.** We create a kafka package. Within the kafka package, create a class named OrderConsumer.

Create Email Service Microservice

We have decided to implement the EmailService Microservice. First we generate the project with the web dependency as well as the Kafka dependency. Then we proceed as follow

- **Change server port. Set it as 8082 into the application.properties file.**
- **Configure Kafka Consumer into the application.properties file.**
- **Create Kafka Consumer. We create a kafka package. Within the kafka package, create a class named OrderConsumer.**

Create BaseDomains Microservice

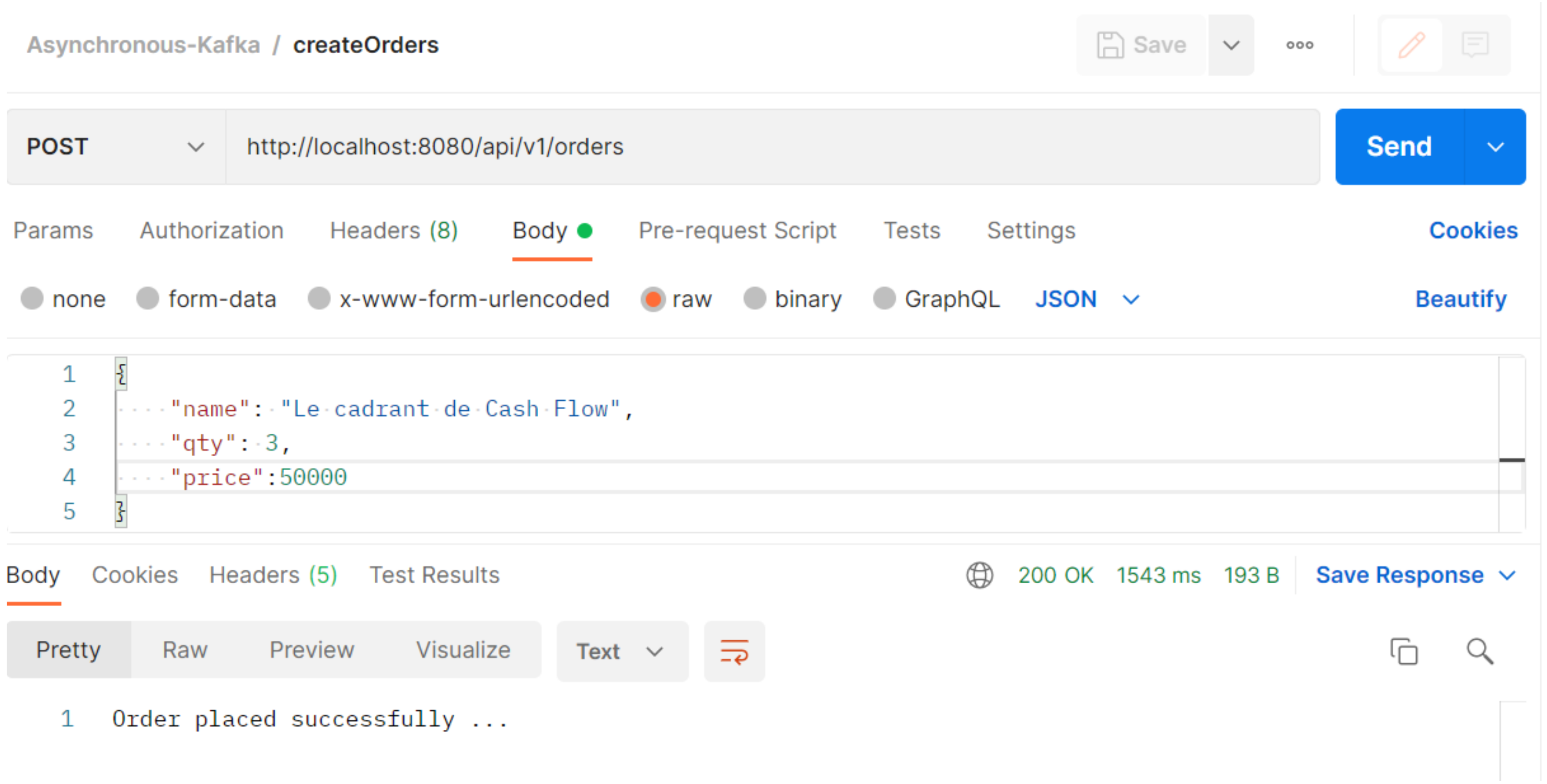
We have decided to implement the BaseDomains Microservice. First we generate the project with the Lombok dependency. Then we proceed as follow

- **Create the Order class into the dto package.**

- **Create the OrderEvent class into the dto package.**

Test the 3 microservices together with Postman

Let’s run the 3 microservices together


Asynchronous-Kafka / createOrders
Save
POST
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (5) Test Results
200 OK 1543 ms 193 B
Save Response
Pretty Raw Preview Visualize Text

```
POST http://localhost:8080/api/v1/orders
```

```
{
    "name": "Le cadrant de Cash Flow",
    "qty": 3,
    "price":50000
}
```

```
Order placed successfully ...
```

Since Order has placed, the order event has been received in stock service as well as in email service

Conclusion

We have learned how to build a simple Event-Driven Microservices application using Spring Boot and Apache Kafka. We have also learned how to create multiple consumers who will subscribe to a single Kafka topic to consume the events/messages. The links of our resources and GitHub repository are attached below

https://www.javaguides.net/2022/07/event-driven-microservices-using-spring-boot-and-apache-kafka.html?spref=tw

https://www.javaguides.net/2022/07/what-is-event-driven-architecture.html

https://github.com/Dilane-Kamga/Asynchronous-with-EventQueue-or-MessageQueue-in-Spring-boot.git

MERCI

Pour votre attention