

## Program szkolenia:

# Apache Kafka - niezbędny w ujęciu dla programistów i programistek Java

### Informacje:

<b>Nazwa:</b>	<b>Apache Kafka - niezbędny w ujęciu dla programistów i programistek Java</b>
<b>Kod:</b>	<b>kafka-java</b>
<b>Kategoria:</b>	Architektura Java
<b>Odbiorcy:</b>	
<b>Czas trwania:</b>	3 dni
<b>Forma:</b>	50% wykłady / 50% warsztaty

W trakcie tego praktycznego warsztatu zapoznamy się z narzędziami i bibliotekami, które pozwolą programistom Java w efektywny sposób implementować oraz testować systemy wykorzystujące klaster Kafka.

Przedstawione zostaną innowacyjne sposoby implementacji, w których Kafka to nie tylko brokera wymiany wiadomości, to też narzędzie do trwałego zapisywania i odczytywania danych, ich agregacji oraz strumieniowania. Omówimy sposoby integracji danych z klastra z popularnymi bazami danych oraz sposoby łączenia ze sobą strumieni danych.

W części warsztatowej położymy nacisk na testowanie automatyczne wytworzonego kodu, zgodność z wzorcami projektowymi oraz dobrymi praktykami wytwarzania oprogramowania.

Szkolenie to niezbędny programistyczny dla każdego dewelopera Java rozpoczynającego swoją przygodę z implementacją aplikacji posługujących się Kafka. Jest to zbiór najbardziej efektywnych praktyk zebranych na bazie doświadczenia ponad sześciu lat pracy z systemami wykorzystującymi Apache Kafka.

## Szczegółowy program:

### 1. Architektura Apache Kafka

#### 1.1. Wprowadzenie do komunikacji asynchronicznej

1.1.1. Komunikacja w monolicie vs mikrouslugi

1.1.2. Przykłady zastosowań komunikacji asynchronicznej

1.1.3. Systemy oparte o zdarzenia

#### 1.2. Architektura Apache Kafka

##### 1.2.1. Wprowadzenie do podstawowych pojęć

1.2.1.1. Topics, Partitions, Offsets

1.2.1.2. Brokers

1.2.1.3. Topics Replication

1.2.1.4. Producers

1.2.1.5. Consumers, Consumers Groups

1.2.1.6. Offsets

1.2.1.7. Acks

##### 1.2.2. Kolejność wiadomości

##### 1.2.3. Replikacja

##### 1.2.4. Jak dobrać odpowiednią ilość partycji?

##### 1.2.5. Poziomy potwierdzenia dostarczenia wiadomości

##### 1.2.6. Co zapewnia Apache Kafka

##### 1.2.7. Skąd bierze się wydajność?

#### 1.3. Architektura Confluent Platform

1.3.1. Kafka Streams

1.3.2. Kafka Connect

1.3.3. KSQL

1.3.4. Rest Proxy

1.3.5. Schema Registry

1.4. Kafka CLI

1.4.1. Zarządzanie Topic'ami

1.4.2. Wysyłanie wiadomości

1.4.3. Odbieranie wiadomości

1.4.4. Monitorowanie i zarządzanie Consumer Groups

1.4.5. Zarządzanie Offset'ami

1.5. Natywny klient Java

1.5.1. Kafka Producer

1.5.1.1. Implementacja niestandardowego partycjonowania

1.5.1.2. Serializacja wiadomości

1.5.1.3. Idempotentność

1.5.2. Kafka Consumer

1.5.2.1. Delivery semantics

1.5.2.2. Deserializacja wiadomości

1.5.2.3. Partition assignment strategies

1.5.2.4. Static membership

1.5.2.5. Implementacja cache w pamięci opartego o Kafka Consumer

1.5.3. Transakcje

1.5.3.1. Jak działają transakcje?

1.5.3.2. Jak dobrać parametr transaction.id?

1.5.3.3. Do czego wykorzystać transakcje na Apache Kafka?

1.5.3.4. Implementacja aplikacji biznesowej przetwarzającej wiadomości w sposób transakcyjny

1.5.4. Jak skonfigurować system do bezstratnego dostarczania wiadomości?

## 2. Kafka Streams

2.1. Koncepcja zastosowania w ekosystemie Confluent

2.2. Architektura aplikacji opartych o Kafka Streams

2.2.1. Dualizm strumień-tablica

2.2.2. Topiki kompaktowe

2.2.3. Lokalna baza danych RocksDB

2.2.4. Changelog topic

2.2.5. Repartition topic

2.3. Analiza topologii strumieni

2.4. Reset tool

2.5. Operacje bezstanowe

2.6. Operacje stanowe

2.6.1. Zapis stanu w KTable

2.6.2. Zapis stanu w GlobalKTable

2.6.3. Agregowanie danych

2.6.4. Łączenie strumieni

2.6.5. Agregacja danych w oknach czasowych

2.7. Odczytywanie lokalnego stanu systemu

2.7.1. Interactive queries

2.7.2. ReadOnlyKeyValueStore

2.7.3. ReadOnlyWindowStore

2.7.4. Standby replicas

2.8. Transformers

2.8.1. Przetwarzanie zgodnie z podejściem Stateful Record-By-Record

2.9. Exactly once processing

2.10. Testowanie aplikacji opartych o Kafka Streams

2.10.1. Testy jednostkowe z wykorzystaniem kafka-streams-test-utils

2.10.2. Testy integracyjne z wykorzystaniem Testcontainers

2.11. Implementacja wzorców

2.11.1. Saga

2.11.2. Read-Process-Write

2.11.3. Sortowanie i deduplikacja zdarzeń

### 3. Wsparcie dla Apache Kafka w Spring Boot

3.1. Podstawowe operacje

3.1.1. Wysyłka wiadomości

3.1.1.1. KafkaTemplate vs. KafkaProducer

3.1.1.2. RoutingKafkaTemplate

3.1.1.3. ReplayingKafkaTemplate

3.1.2. Odbiór wiadomości

3.1.2.1. Wątki

3.1.2.2. Walidacja wiadomości

3.1.2.3. Zarządzanie potwierdzeniami

3.1.2.4. Ponawianie

3.1.2.5. Wysyłka odpowiedzi

3.1.3. Transakcje

3.1.3.1. Publikowanie transakcyjne

3.1.3.2. Transakcje inicjowane przez odczyt wiadomości

3.1.3.3. Zarządzanie transaction.id przez Spring

## 3.2. Prawidłowa konfiguracja aplikacji

### 3.2.1. Zarządzanie błędami

### 3.2.2. Serializacja i deserializacja

### 3.2.3. Nagłówki

## 3.3. Testowanie

### 3.3.1. Przegląd bibliotek do testowania asynchronicznego

#### 3.3.1.1. JUnit - Awaitility

#### 3.3.1.2. Spock - PollingConditions

### 3.3.2. Wykorzystanie mechanizmów Spring w testach

### 3.3.3. Uruchamianie Kafka do testów z wykorzystaniem spring-kafka-test

### 3.3.4. Uruchamianie Kafka do testów z wykorzystaniem Testcontainers

## 3.4. Implementacja aplikacji biznesowej z wykorzystaniem Spring Boot, Apache Kafka i relacyjnej bazy danych

### 3.4.1. Modelowanie procesów biznesowych w podejściu asynchronicznym

### 3.4.2. Implementacja mechanizmów zapewniających idempotentność

### 3.4.3. Wpływ asynchroniczności na modelowanie REST API

### 3.4.4. Różne poziomy spójności

### 3.4.5. Implementacja wzorca Outbox