

Program szkolenia:

Spring for Apache Kafka - wprowadzenie i zagadnienia zaawansowane

Informacje:

Nazwa:	Spring for Apache Kafka - wprowadzenie i zagadnienia zaawansowane
Kod:	ds-spring
Kategoria:	Kafka
Czas trwania:	3 dni
Forma:	50% teoria 50% warsztat

Koncepcja szkolenia została oparta o zestaw praktycznych problemów jakie stają przed developerem – prezentujemy sprawdzone i najlepsze ich rozwiązania w Spring for Apache Kafka.

Szkolenie swoim zakresem wykracza daleko poza powszechnie dostępne materiały dydaktyczne.

Szkolenie oprócz prezentacji technologii Apache Kafka oraz Spring zostało wzbogacone o aspekty: doboru architektury aplikacji, konfiguracji narzędzi developerskich, praktyk projektowych odpowiednich do produktywniej pracy z Apache Kafka i Spring.

Wiedza zdobyta podczas szkolenia pozwoli na pełne wykorzystanie frameworka, zwiększenie rozszerzalności systemów oraz racjonalizację procesu testowania - zwiększając jakość z jednoczesną redukcją kosztów utrzymania testów.

Program szkolenia stanowi bazę do której możemy dodawać wybrane zagadnienia.

Zalety szkolenia:

- Dobór architektury aplikacji
- Wzorce i pułapki
- Nacisk na transakcyjność, wysoką wydajność i spójność
- Elementy systemów rozproszonych

Szczegółowy program:

1. Architektura Apache Kafka

1.1. Wprowadzenie do komunikacji asynchronicznej

1.1.1. Komunikacja w monolicie vs mikrouслуги

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

1.3. Kafka CLI

1.3.1. Zarządzanie Topic'ami

1.3.2. Wysłanie wiadomości

1.3.3. Odbieranie wiadomości

1.3.4. Monitorowanie i zarządzanie Consumer Groups

1.3.5. Zarządzanie Offset'ami

1.4. Natywny klient Java

1.4.1. Kafka Producer

1.4.2. Kafka Consumer

1.4.3. Transakcje

1.4.3.1. Jak działają transakcje?

1.4.3.2. Do czego wykorzystać transakcje na Apache Kafka?

1.4.3.3. Implementacja aplikacji biznesowej przetwarzającej wiadomości w sposób transakcyjny

2. Spring for Apache Kafka

2.1. Architektura frameworku

2.1.1. MessageListener

2.1.2. ContainerProperties

2.1.3. KafkaMessageListenerContainer

2.1.4. KafkaConsumerFactory

2.2. Podstawowe operacje

2.2.1. Połączenie do klastra Apache Kafka

2.2.2. Konfiguracja Topic

2.2.3. Wysyłka wiadomości

2.2.3.1. KafkaTemplate vs. KafkaProducer

2.2.3.2. RoutingKafkaTemplate

2.2.3.3. ReplyingKafkaTemplate

2.2.4. Odbiór wiadomości

2.2.4.1. Konsumowanie wiadomości bez wykorzystania adnotacji

2.2.4.2. Konsumowanie wiadomości za pomocą adnotacji

2.2.4.3. Wątki

2.2.4.4. Walidacja wiadomości

2.2.4.5. Zarządzanie potwierdzeniami

2.2.4.6. Wysyłka odpowiedzi

2.2.4.7. Wstrzymywanie konsumowania wiadomości

2.2.5. Ponowienia i obsługa błędów

2.2.5.1. Blokujące ponowienia

2.2.5.2. Nie blokujące ponowienia

2.2.5.3. Wzorzec Dead letter queue

2.2.6. Transakcje

2.2.6.1. Publikowanie transakcyjne

2.2.6.2. Transakcje inicjowane przez odczyt wiadomości

2.2.6.3. Zarządzanie transaction.id przez Spring

2.2.6.4. Łączenie transakcji bazodanowej z transakcją Kafka

2.3. Prawidłowa konfiguracja aplikacji

2.3.1. Rodzaje i obsługa zdarzeń kontenera Spring

2.3.2. Sterowanie początkowym offsetem

2.3.3. Obsługa wyjątków

2.3.4. Serializacja i deserializacja

2.3.5. Nagłówki wiadomości

2.3.6. Metryki i monitorowanie

2.4. Testowanie

2.4.1. Przegląd bibliotek do testowania asynchronicznego

2.4.1.1. JUnit - Awaitility

2.4.1.2. Spock - PollingConditions

2.4.2. Wykorzystanie mechanizmów Spring w testach

2.4.3. Uruchamianie Kafka do testów z wykorzystaniem spring-kafka-test

2.4.4. Uruchamianie Kafka do testów z wykorzystaniem TestContainers

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

2.5.1. Modelowanie procesów biznesowych w podejściu asynchronicznym

2.5.2. Implementacja mechanizmów zapewniających idempotentność

2.5.3. Wpływ asynchroniczności na modelowanie REST API

2.5.4. Różne poziomy spójności

2.5.5. Implementacja wzorców Inbox i Outbox