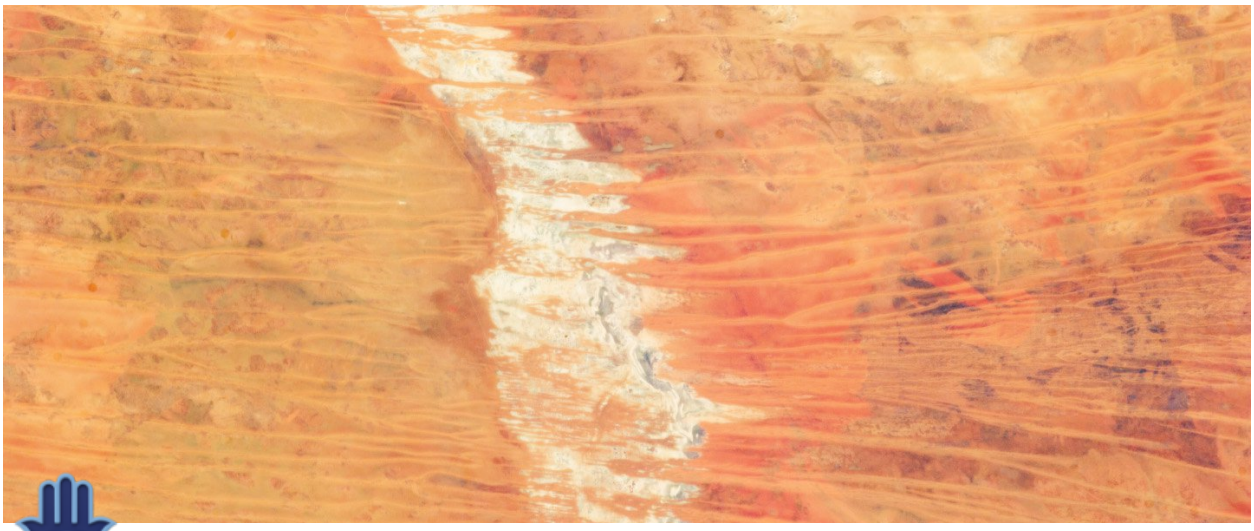


7.3 MOM

Status	In Arbeit
№ ID	NR-73
URL	https://elearning.tgm.ac.at/mod/assign/view.php?id=136640
AI summary	Die Übung demonstriert die Funktionsweise von Message Oriented Middleware (MOM) mit Apache Kafka anhand eines Wahlzentrums, das alle 10 Minuten Wahlergebnisse abfragt. Es werden Eigenschaften von MOM, die Funktionsweise von JMS Queues und Topics sowie Codebeispiele zum Senden und Empfangen von Nachrichten in Kafka behandelt. Die Kommunikation erfolgt über REST-Schnittstellen und die gesammelten Daten werden in JSON oder XML bereitgestellt.
Fach	DezSys SYT
Last edited time	@November 19, 2024 4:41 PM
Letzte Abgabe	@December 20, 2024

Note

TOC

1. Einführung

Diese Übung soll die Funktionsweise und Implementierung von einem Message Oriented Middleware (MOM) mithilfe des **Frameworks Apache Kafka** demonstrieren. **Message Oriented Middleware (MOM)** ist neben InterProcessCommunication (IPC), Remote Objects (RMI) und Remote Procedure Call (RPC) eine weitere Möglichkeit, um eine Kommunikation zwischen mehreren Rechnern umzusetzen.

2. Beschreibung

Die Umsetzung basiert auf einem praxisnahen Beispiel eines Wahlzentrums. Die Zentrale der Wahllokale möchte alle 10 Minuten den aktuellen Auszahlungsstand jedes Wahllokals abfragen.

Mit diesem Ziel soll die REST-Applikation aus MidEng 7.1 Warehouse REST und Dateiformats bei einem entsprechenden Request http:///election2024/send die Daten (JSON oder XML) in eine Message Queue der Zentral übertragen. In regelmäßigen Abständen werden alle Message Queues der Zentrale abgefragt und die Daten aller Standorte gesammelt.

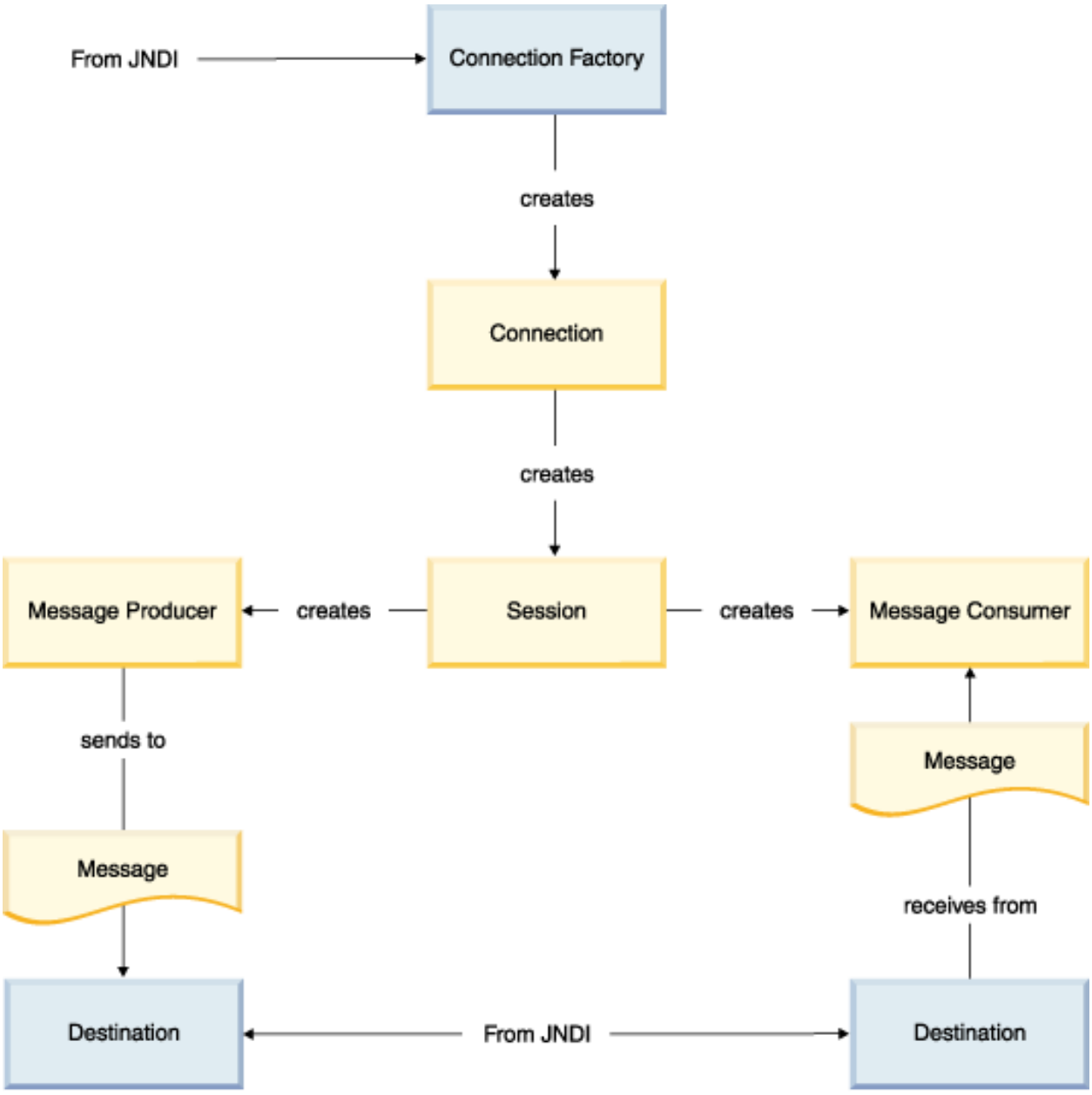
Die gesammelten Wahlergebnisse werden dann erneut über eine REST-Schnittstelle in XML oder JSON zur Verfügung gestellt.

3. Fragen

- Nennen Sie mindestens 4 Eigenschaften der Message Oriented Middleware?
  - Lose Kopplung von Server/Clients
  - Asynchrone/synchrone Kommunikation
  - Server/Dienst muss nicht sofort verfügbar sein
  - Message-Warteschlangen
- Was versteht man unter einer transienten und synchronen Kommunikation?
  - Transistent bedeutet, dass die Nachrichten nicht gespeichert werden, sondern nur in der Verteilung verwendet und danach gelöscht werden
  - Synchron bedeutet, dass Nachrichten, die geschickt werden, direkt an den Client weitergeleitet werden
- Beschreiben Sie die Funktionsweise einer JMS Queue?
  - Wenn man eine Message in die Queue schickt, wird sie je nachdem ob das System transient oder persistent ist gespeichert.
  - Je nachdem, ob ein Client an der Queue hängt, wird die Message weiter geschickt.
  - Jede Nachricht hat eine beliebige Anzahl an Empfängern und sobald diese erreicht wird, wird die Message gelöscht
- JMS Overview - Beschreiben Sie die wichtigsten JMS Klassen und deren Zusammenhang?
  - Connection
    - `ConnectionFactory` → Connection → `Session`
  - Session
    - `Session` → `MessageProducer` / `MessageConsumer`
  - Messages
    - `MessageProducer` → Messages (`BytesMessage` , `TextMessage` , `StreamMessage` , `MapMessage` , `ObjectMessage`)

→ Destination (
`Queue`, `Topic`)

- `MessageListener` + `MessageConsumer` → Messages empfangen



- Beschreiben Sie die Funktionsweise eines JMS Topic?
  - Topics (Themen) werden verwendet, um Nachrichten zu einem bestimmten Typ zuzuordnen.

    - Topics können nicht mit Queues verwendet werden, also entweder oder
- Was versteht man unter einem lose gekoppelten, verteilten System? Nennen Sie ein Beispiel dazu. Warum spricht man hier von lose?
    - Ein lose gekoppeltes System ist ein System, bei dem die Komponenten unabhängig voneinander agieren und nur über eine Schnittstelle kommunizieren.
    - Beispiel
        - Ein JMS System worüber eine Firma zwischen den Abteilungen Daten kommuniziert
    - Warum lose?
        - Die Systeme können unabhängig voneinander funktionieren
        - Die einzelnen Systeme haben wenig, bis keine Information übereinander
        - Die Systeme können einfach ersetzt werden
        - Das System ist standardisiert

Code

Gradle kafka implementation

Damit kafka im spring project funktioniert muss man zu den dependencies von spring noch eine zeile hinzufügen

```
dependencies {
    implementation 'org.springframework.kafka:spring-kafka'
}
```

Senden einer Nachricht an eine Queue

```
@Autowired
private KafkaTemplate<String, String> kafkaTemplate;
```

Zuerst brauchen wir ein Template

```
kafkaTemplate.send("wahl.lokal." + wahllokal, newVotes.toJson()
```

Dann kann man mit der `send` Methode zu einer Queue, in dem Fall `wahl.lokal.*`, eine Nachricht zu der Queue senden.

Nachrichten empfangen

Um Nachrichten zu empfangen, muss man diese Annotation vor die Methode schreiben

- `topicPattern`: ermöglicht es alle queues mit dem prefix `wahl.lokal.` abzurufen
- `groupId`: ermöglicht es die Queue mit Kafka umzusetzen, diese ID muss bei allen Consumern dieser Queues dieselbe sein, damit die Nachricht nur von einem Consumer abgerufen werden kann

```
@KafkaListener(topicPattern = "wahl.lokal.*", groupId = "vote_li
```

Diese Nachrichten werden als `Strings` übergeben und können danach verarbeitet werden

Quellen

- https://www.ibm.com/docs/de/baw/22.x?topic=jms-java-message-service-programming-model **IBM Docs** JMS programming Model
- https://www.ibm.com/docs/en/integration-bus/10.0?topic=structure-jms-message-types **IBM Docs** jms Message Types
- https://jakarta.ee/learn/docs/jakartaee-tutorial/current/messaging/jms-examples/jms-examples.html#_sending_messages **Jarkata EE docs** Sending Messages