MidEng_7.2

Verfasser: **Julian Neuwirt, Felix Aust, Leonhard Stransky, 4AHIT**

Datum: **28.11.2023**

Einführung

Diese Übung soll die Funktionsweise und Implementierung von eine Message Oriented Middleware (MOM) mit Hilfe des Frameworks Apache Active MQ 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.

Die Umsetzung basiert auf einem praxisnahen Beispiel eines Lagerstandorts. Die Zentrale der Handelskette KONSUM moechte einmal pro Tag die aktuellen Lagerbestaende jedes Standortes abfragen.

Mit diesem Ziel soll die REST-Applikation aus MidEng 7.1 Warehouse REST and Dataformats bei einem entsprechenden Request http:///warehouse/sendData die Daten (JSON oder XML) in eine Message Queue der Zentral uebertragen. In regelmaessigen Abstaenden werden alle Message Queues der Zentrale abgefragt und die Daten aller Standorte gesammelt.

Diese Daten werden dann erneut ueber eine REST-Schnittstelle in XML oder JSON zur Verfuegung gestellt.

Projektbeschreibung

Das Ziel dieser Übung ist die Implementierung einer Kommunikationsplattform für eine Handelskette. Dabei erfolgt ein Datenaustausch von mehreren Lagerstandorten mit der Zentrale unter Verwendung einer Message Oriented Middleware (MOM). Die einzelnen Daten des Lagerstandortes sollen an die Zentrale übertragen werden. Es sollen nachrichtenbasierten Protokolle mit Message Queues verwendet werden. Durch diese lose Kopplung kann gewährleistet werden, dass in Zukunft weitere Anlagen hinzugefügt bzw. Kooperationspartner eingebunden werden können.

Fuer die REST-Schnittstelle in der Zentralle muessen die Datenstrukturen der einzelnene Standorte zusammengefasst werden. Um die Datenintegrität zu garantieren, sollen jene Daten, die mit der Middleware übertragen werden in einer LOG-Datei abgespeichert werden.

Aufgaben:

MOMSender:

```
    // Parameter für die Verbindung
    private static String user = ActiveMQConnection.DEFAULT_USER;
    private static String password = ActiveMQConnection.DEFAULT_PASSWORD;
    private static String url = ActiveMQConnection.DEFAULT_BROKER_URL;
    private static String subject = "windengine_001";

    // Creating Session
    session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
```

```
    destination = session.createTopic( subject );
    // Create the producer.
    producer = session.createProducer(destination);
    producer.setDeliveryMode( DeliveryMode.NON_PERSISTENT );
    // Create the message
    TextMessage m = session.createTextMessage(i); p
    roducer.send(m);
```

Dieser Java-Code stellt eine Verbindung zu einem ActiveMQ-Nachrichtenbroker her und sendet eine Textnachricht zu einem bestimmten Thema. Er verwendet Standardverbindungseinstellungen und erstellt dann eine Sitzung sowie einen Produzenten für das gewählte Thema. Die Nachricht wird im nicht persistierenden Modus gesendet, was bedeutet, dass sie nicht dauerhaft auf der Festplatte gespeichert wird. Insgesamt ermöglicht der Code das effiziente Versenden von Nachrichten in Java-Anwendungen über ActiveMQ.

Reciever:

```
consumer = session.createConsumer( destination );
// Start receiving
TextMessage message = (TextMessage) consumer.receive();
while ( message != null ) {
    System.out.println("Message received: " + message.getText());
    message.acknowledge();
    message = (TextMessage) consumer.receive();
}
connection.stop();
```

Dieser Java-Code erstellt einen Consumer für ein Ziel in einer ActiveMQ-Sitzung und empfängt Textnachrichten in einer Endlosschleife. Jede empfangene Nachricht wird in der Konsole ausgegeben und anschließend bestätigt. Der Empfangsprozess läuft, bis keine weiteren Nachrichten vorhanden sind oder die Verbindung gestoppt wird.

Conversion:

```
WarehouseData data = service.getWarehouseData( inID );
ObjectMapper objectMapper = new ObjectMapper();
String json_string = objectMapper.writeValueAsString(data);
System.out.println( ""  + data);
new MOMSender( json_string);
```

In folgendem Text wird das WarehouseData-Objekt mit hilfe eines ObjektMapper in einen String umgewandelt. Dazu wird die Mehtode writeValueAsString verwendet. Dann am schluss wieder der Sender aufgerufen

Quellen: