

MidEng 7.2 Warehouse Message Oriented Middleware [GK]

Das Ziel der Aufgabe war es Daten in eine Queue zu schicken und sie auf einer REST Schnittstelle ausgeben.

Das Sender Project

Im Sender Bereich des Codes hab ich mittels ActiveMQ Connectionfactory eine Verbindung aufgebaut und somit die Kommunikation mit der Queue ermöglicht.

```
1 usage  ± Ivan Milev *
public MOMSender(WarehouseData warehouseData) {

    System.out.println( "Sender started." );

    // Create the connection.
    Session session = null;
    Connection connection = null;
    MessageProducer producer = null;
    Destination destination = null;

    try {

        ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(user, password, url);
        connection = connectionFactory.createConnection();
        connection.start();

        // Create the session
        session = connection.createSession( transacted: false, Session.AUTO_ACKNOWLEDGE);
        destination = session.createQueue(subject); // Use createQueue for point-to-point communication

        // Create the producer.
        producer = session.createProducer(destination);
```

```

// Create the session
session = connection.createSession(transacted: false, Session.AUTO_ACKNOWLEDGE);
destination = session.createQueue(subject); // Use createQueue for point-to-point communication

// Create the producer.
producer = session.createProducer(destination);
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

// Convert the WarehouseData object to a JSON string
Gson gson = new Gson();
String jsonString = gson.toJson(warehouseData);

// Create the message
TextMessage message = session.createTextMessage(jsonString);
producer.send(message);
System.out.println(message.getText());

connection.stop();

```

Er wandelt die JSON in einen String um mittels der Google Api GSON und schickt diese als Message an die Queue "Warehouse" welche am Server läuft.

Die Ausführung erfolgt mit dem Befehl `gradle bootRun --args='sender'`

```

no usages
private WarehouseService service;

Ivan Milev
public static void main(String[] args) { SpringApplication.run(WarehouseMomApplication.class, args); }


Ivan Milev *
@Override
public void run(String... args) throws Exception {

    String flag = new String(original: "receiver");
    for(String arg:args) {
        flag = arg;
    }

    if ( flag.toLowerCase().equals("sender") )
        new MOMSender(WarehouseService.getWarehouseData());
    else
        new MOMReceiver();
}

```

Führt man nun dieses Programm aus bekommt die Queue eine Message. Das kann man gut an der GUI des Servers erkennen.



Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Browse Warehouse

Message ID	Correlation ID	Persistence	Priority	Redelivered	Reply To	Timestamp	Type	Operations
ID:LAPTOP-SH5MCOOA-64944-1701784176080-1:1:1:1		Non Persistent	4	false		2023-12-05 13:49:36:258 UTC		Delete

View Consumers

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Headers

Message ID

ID:LAPTOP-SH5MCOOA-64944-1701784176080-1:1:1:1

Destination

queue://Warehouse

Correlation ID

Group

Sequence

0

Expiration

0

Persistence

Non Persistent

Priority

4

Redelivered

false

Reply To

Timestamp

2023-12-05 13:49:36:258 UTC

Type

Properties

Message Actions

Delete

Copy

Move

-- Please select --

Message Details

{ "warehouseID": "955", "warehouseName": "Wien Jedlersdorf", "timestamp": "2023-12-05 14:49:36.034", "warehouseCountry": "Austria", "warehouseCity": "Vienna", "address": "Jedlersdorfer str 14" }

Der Receiver

```

@GetMapping("/getQueue")
public ResponseEntity<String> getQueueMessages() {
    List<String> queueMessages = MOMReceiver.receiveMessages();
    JsonResponse jsonResponse = new JsonResponse();
    jsonResponse.setMessages(queueMessages);

    String jsonString = convertListToJson(queueMessages);

    System.out.println("Received messages: " + queueMessages);
    return ResponseEntity.ok(jsonString);
}

```

usage

```

public static String convertListToJson(List<String> stringList) {
    StringBuilder jsonBuilder = new StringBuilder();
    jsonBuilder.append("[");

    for (int i = 0; i < stringList.size(); i++) {

```

Der Code ermöglicht es auf der URL auf /getQueue die Daten die es von der Queue bekommen hat anzuzeigen.

```

public static List<String> receiveMessages() {
    List<String> messages = new ArrayList<>();

    try {
        ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(user, password, brokerUrl);
        Connection connection = connectionFactory.createConnection();
        connection.start();

        Session session = connection.createSession(transacted: false, Session.AUTO_ACKNOWLEDGE);
        Destination destination = session.createQueue(subject);
        MessageConsumer consumer = session.createConsumer(destination);

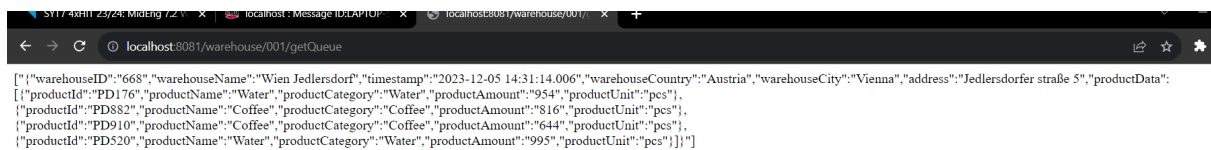
        while (true) {
            Message message = consumer.receive(timeout: 1000); // Timeout set to 1 second

            if (message instanceof TextMessage) {
                TextMessage textMessage = (TextMessage) message;
                messages.add(textMessage.getText());
            } else if (message == null) {
                break; // No more messages
            }
        }

        consumer.close();
        session.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Der Receiver ist so aufgebaut das er wieder eine Connection zur Queue Warehouse aufbaut und dort die Nachrichten empfängt und abspeichert.



```
[{"warehouseID":"668","warehouseName":"Wien Jedlersdorf","timestamp":"2023-12-05 14:31:14.006","warehouseCountry":"Austria","warehouseCity":"Vienna","address":"Jedlersdorfer straÙe 5","productData":
[{"productId":"PD176","productName":"Water","productCategory":"Water","productAmount":"954","productUnit":"pcs"},
{"productId":"PD882","productName":"Coffee","productCategory":"Coffee","productAmount":"816","productUnit":"pcs"},
{"productId":"PD910","productName":"Coffee","productCategory":"Coffee","productAmount":"644","productUnit":"pcs"},
{"productId":"PD520","productName":"Water","productCategory":"Water","productAmount":"995","productUnit":"pcs"}]]
```

Die Daten werden dann dementsprechend als JSON ausgegeben.