MidEng 7.2 Warehouse Message Oriented Middleware [GK]

Sergej Rychkov 4DHIT

Fragestellungen:

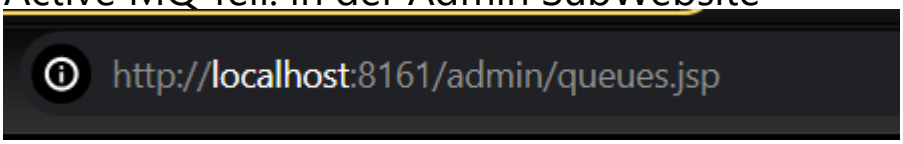
- Nennen Sie mindestens 4 Eigenschaften der Message Oriented Middleware?
    - Nachrichtenorientiert --> MOM ermöglich die Austausch zwischen 2 oder mehreren Anwendungen mithilfe von Nachrichten
    - Skalierbarkeit --> MOM ermöglicht die einfache Skalierung von Systemen.
    - Persistenz --> Bei MOM wird die Nachricht falls gebraucht auch dauerhaupt gespeichert, bis der Empfänger verfügbar ist.
    - Zuverlässigkeit --> Eine sichere und zuverlässige Zustellung von Nachrichten ist garantiert. Fehlererkennung, Fehlerbehandlung und Wiederholungsmechanismen, um sicherzustellen, dass Nachrichten ordnungsgemäß übertragen werden sind dabei.
- Was versteht man unter einer transienten und synchronen Kommunikation?
    - Synchrone Kommunikation: Absender erwartet eine Antwort von dem Empfänger sofort. Kann zu Wartezeiten führen, wenn der Empfänger nicht direkt antwortet.
    - Transiente Kommunikation: Nachrichten werden temporär gespreichert. Das bedeutet, dass Nachrichten, die nicht sofort empfangen werden können, verloren gehen/nicht wieder gesendet werden.
- Beschreiben Sie die Funktionsweise einer JMS Queue?
    - MS Queue ist eine Warteschlange für Nachrichten, bei der die Nachrichten in der Reihenfolge ihres Eintreffens verarbeitet werden. Sender senden Nachrichten an die Queue, und Empfänger holen Nachrichten aus der Queue ab
- JMS Overview - Beschreiben Sie die wichtigsten JMS Klassen und deren Zusammenhang?
    - ConnectionFactory(connection zum JMS Broker), Connection(Bietet eine Verbindung zu einem JMS Broker.), Session (Stellt eine Transaktions- oder Nicht-Transaktions-Sitzung für die Nachrichtenproduktion und -konsum bereit.), Destination: Repräsentiert das Ziel (Queue oder Topic), MessageProducer(Sendet Nachrichten an eine Destination),MessageConsumer: (Empfängt Nachrichten von einer Destination).
- Beschreiben Sie die Funktionsweise eines JMS Topic?
    - JMS Topic ist ein Austauschpunkt für Nachrichten, bei dem mehrere Consumer Nachrichten empfangen können.(Auf einem Publisher-Subscriber Prinzip)
- Was versteht man unter einem lose gekoppelten verteilten System? Nennen Sie ein Beispiel dazu. Warum spricht man hier von lose?

- Es ist eine Architektur, bei der die einzelnen Komponenten unabhängig voneinander arbeiten und wenig Wissen über die Implementierungsdetails der anderen Komponenten haben. Beispiel: Web-Services-Architektur Warum: Unabhängigkeit, Flexibilität, Skalierbarkeit und Fehlerisolierung

Was ich gemacht habe:

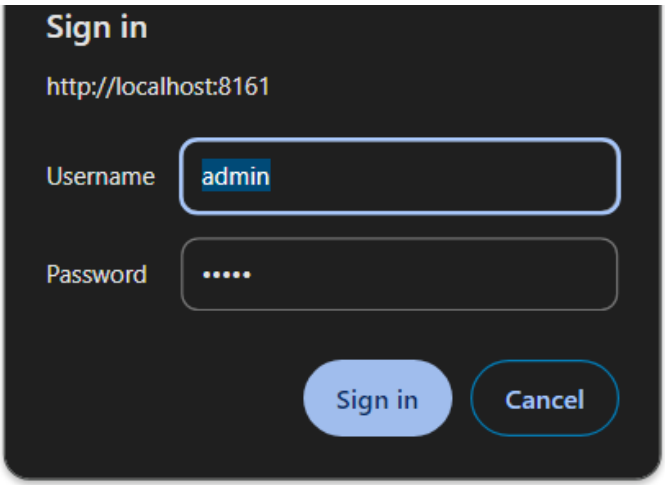
ActiveMQ installiert, In dem richtigen Verzeichniss: .\bin\activemq.bat start im Terminal ausführen,

Active MQ Teil: in der Admin SubWebsite

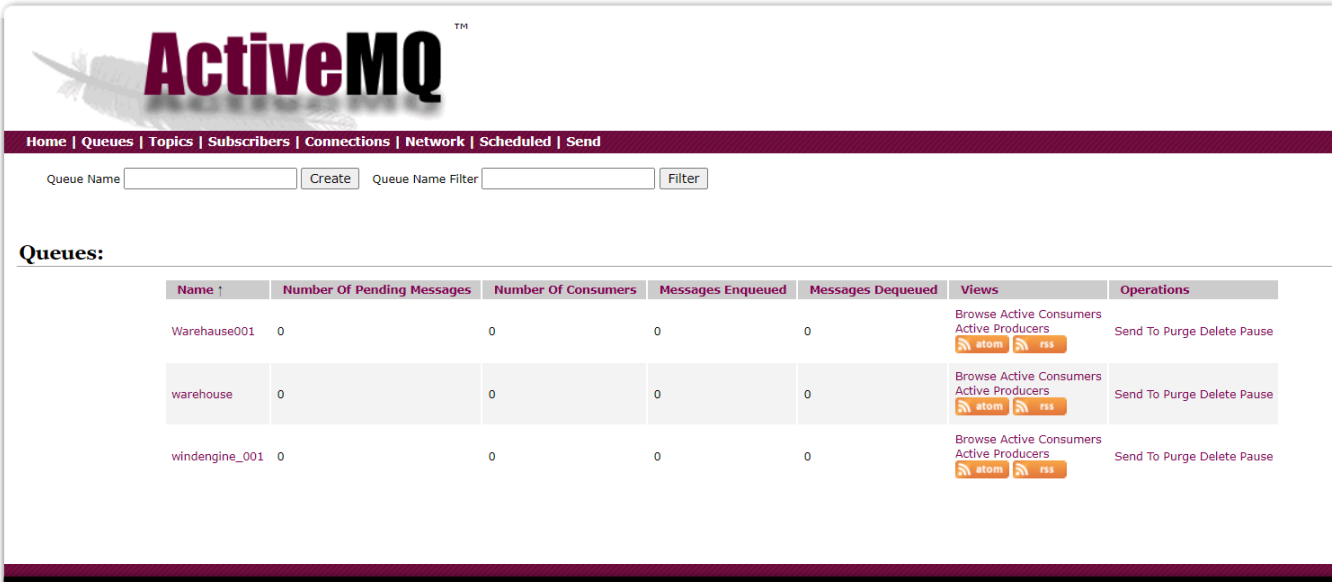


In die Queue gehen.

hier logt man sich ein

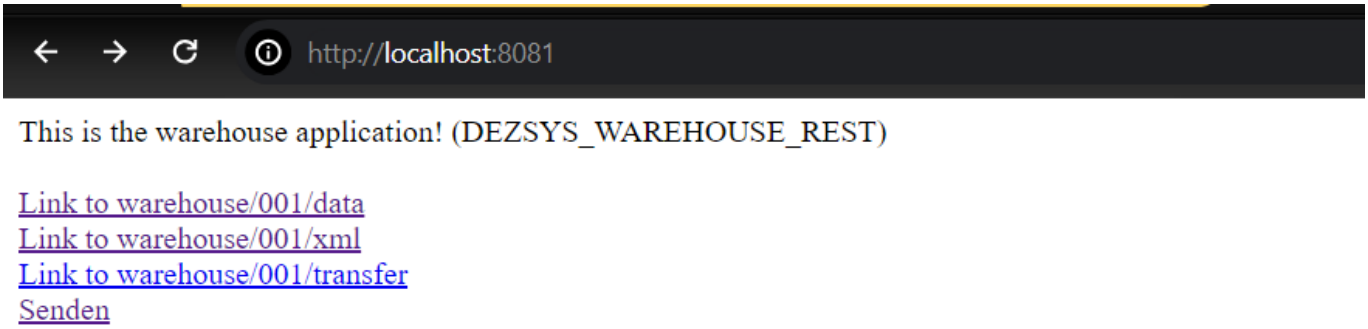


hier sind unsere queues:

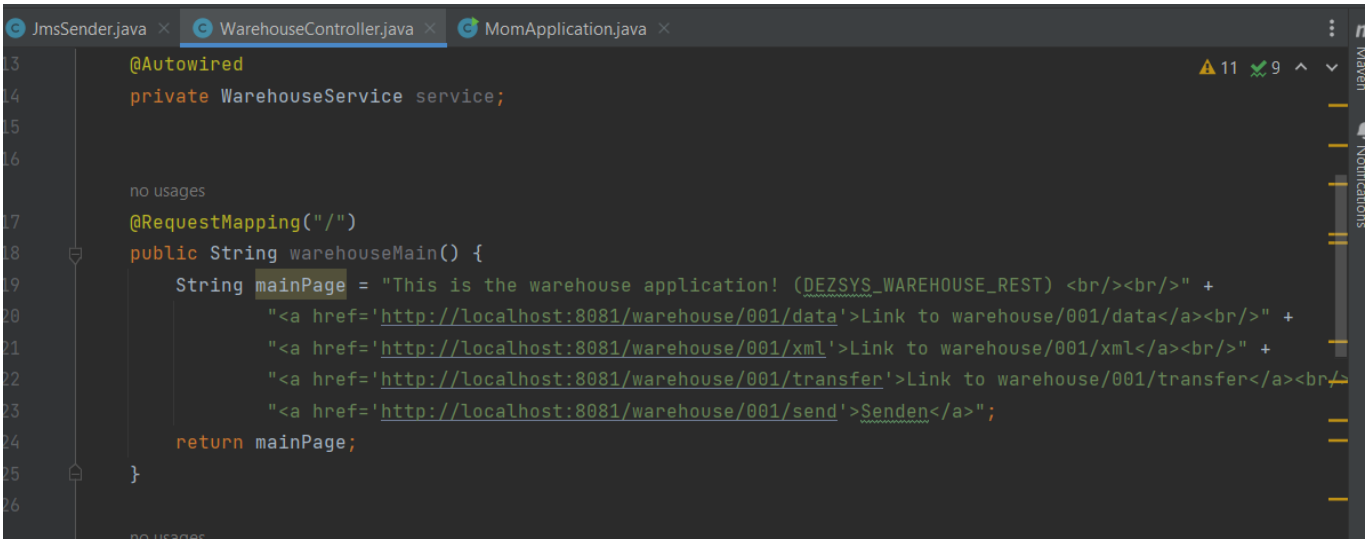


Senden:

Hier ist die Webanwendung



So sieht der Code davon aus:

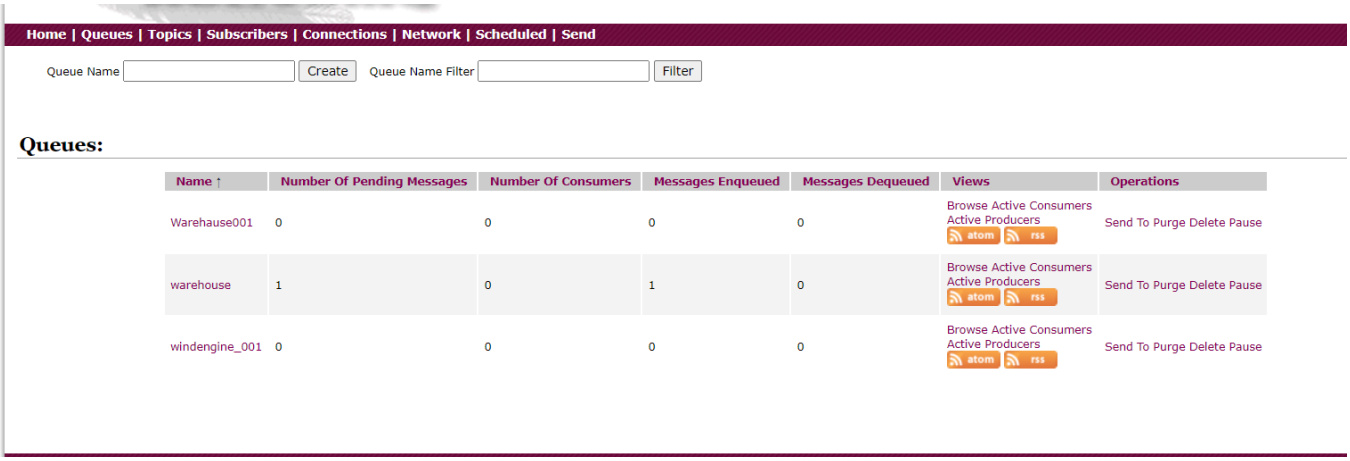


Beim drücken auf den Senden-Link, wird der folgende Code ausgeführt:

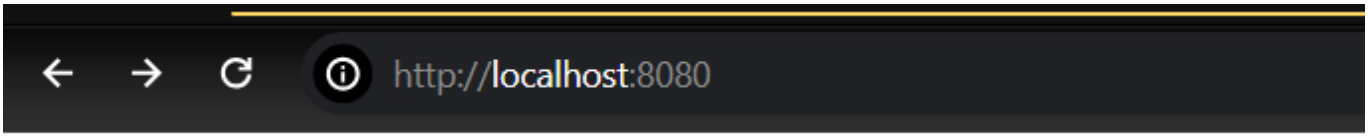
JmsSender Class:

```java
1 usage
public void sendMessage(Serializable obj) {
    try {
        ObjectMessage message = session.createObjectMessage(obj);
        producer.send(message);
        System.out.println("Send data: " + obj.toString());
    } catch (JMSException e) {
        System.err.println("Error while sending Message: " + e);
    }
}
```

So sieht es auf der Queue Seite danach aus:

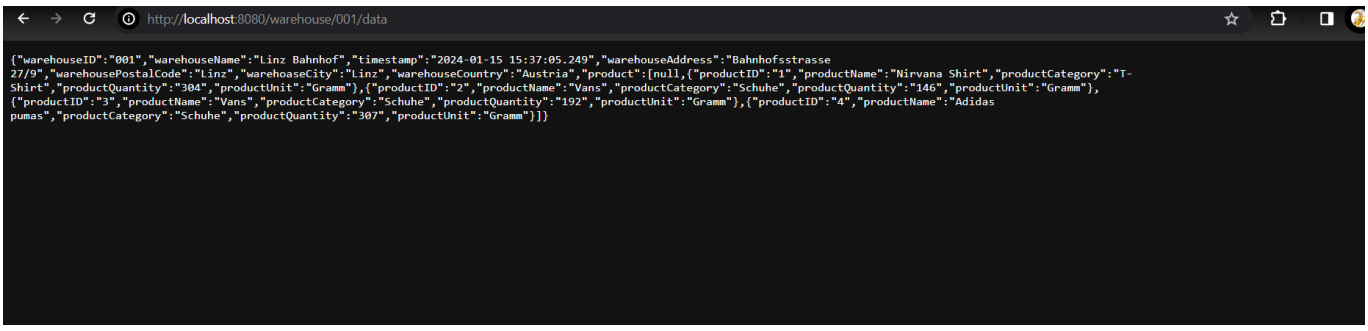


Auf der anderen Lager-Website drücken wir auf /data:

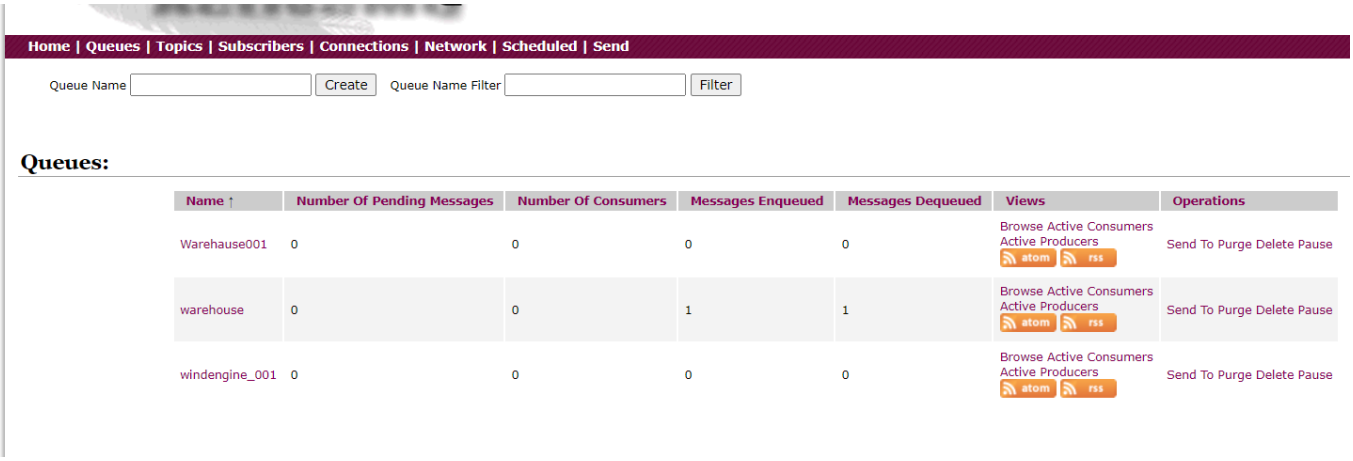


This is the warehouse application! (DEZSYS_WAREHOUSE_REST)

Link to warehouse/001/data
Link to warehouse/001/xml
Link to warehouse/001/transfer



Und so sieht es auf der Queue Seite aus:



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

Queue Name Create Queue Name Filter Filter

Queues:

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
Warehause001	0	0	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
warehouse	0	0	1	1	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
windengine_001	0	0	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause

So sieht der Code beim Receiver aus:

```
// Start receiving

Message message = consumer.receive(); // Blocking call to receive a message

if (message instanceof ObjectMessage) {
    ObjectMessage objectMessage = (ObjectMessage) message;
    Object receivedObject = objectMessage.getObject();
    System.out.println("Message received: " + objectMessage);

} else {
```