

SYT7 4xHIT 23/24 / Middleware Engineering & Dezentrale Systeme / MidEng 7.2 Warehouse Message Oriented Middleware [GK] – 4h

MidEng 7.2 Warehouse Message Oriented Middleware [GK] – 4h

✓ Done: View ✓ Done: Make a submission ✓ Done: Receive a grade

Opened: Monday, 18 September 2023, 12:00 AM

Due: Friday, 22 December 2023, 12:00 AM

DEZSYS_GK72_WAREHOUSE_MOM

Join GIT repository: https://github.com/ThomasMicheler/DEZSYS_GK772_WINDPARK_MOM.git

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 Lagerstandortes. Die Zentrale der Handelskette KONSUM moechte einmal pro Tag die aktuellen Lagerbestände jedes Standortes abfragen.

Mit diesem Ziel soll die REST-Applikation aus MidEng 7.1 [Warehouse](#) REST und Dataformats bei einem entsprechenden Request <http://warehouse/sendData> die Daten (JSON oder XML) in eine Message Queue der Zentral uebertragen. In regemaessigen 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.

1.1 Ziele

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 Zentrale muessen die Datenstrukturen der einzelne Standorte zusammengefasst werden. Um die Datenintegrität zu garantieren, sollen jene Daten, die mit der Middleware übertragen werden in einer LOG-Datei abgespeichert werden.

1.2 Voraussetzungen

- Grundlagen Architektur von verteilten Systemen
- Grundlagen zur nachrichtenbasierten Systemen / Message Oriented Middleware
- Verwendung des Message Brokers Apache ActiveMQ
- Verwendung der XML- oder JSON Datenstruktur der Lagerstandortes
- Verwendung der Demo-Applikation MOMApplication (inklusive MOMReceiver und MOMSender) (siehe Repo)
- Verwendung von

1.3 Aufgabenstellung

Implementieren Sie die Handelskette-Kommunikationsplattform mit Hilfe des Java Message Service. Verwenden Sie Apache ActiveMQ (<http://activemq.apache.org>) als Message Broker Ihrer Applikation. Das Programm soll folgende Funktionen beinhalten:

- Installation von Apache ActiveMQ in der Zentrale.
- Jeder Lagerstandort hat eine Message Queue mit einer ID am zentralen Rechner.
- Jeder Lagerstandort legt in regelmässigen Abständen die Daten der Anlage in der Message Queue ab.
- Bei einer erfolgreichen Übertragung sendet die Zentrale die Nachricht "SUCCESS" an den Lagerstandort retour.
- Der zentrale Rechner fragt in regelmässigen Abständen alle Message Queues ab.
- Der Zentralrechner fuegt alle Daten aller Lagerstandorte zusammen und stellt diese an einer REST Schnittstelle im JSON/XML Format zur Verfügung.

1.4 Demo Applikation

- Starten des Message Broker Apache Activemq
`[Installationsverzeichnis Apache Activemq]/bin/activemq start`
- Administration von Apache Activemq via Webinterface
`http://localhost:8161/admin`
`Username: admin, Password: admin`

1.4.1 warehouse_demo1

Demo 1 beinhaltet eine Implementierung, die alle Einzelschritte zur Implementierung von Java und JMS beinhaltet und uebersichtlich darstellt. Die JMS Teile Sender und Empfaenger werden einzeln aufgerufen.

- Starten des Empfaengers MOMReceiver

```
mvn clean spring-boot:run -Dspring-boot.run.arguments=receiver
-OR-
gradle clean bootRun -Pargs=receiver
```
- Starten des Sender MOMSender

```
mvn clean spring-boot:run -Dspring-boot.run.arguments=sender
-OR-
gradle clean bootRun -Pargs=sender
```

1.4.2 warehouse_demo2

Demo 2 beinhaltet eine Implementierung, die eine Umsetzung von JMS mit Springboot darstellt und aktuelle Bibliotheken und Annotation beinhaltet. Der Sender wird als REST Controller aufgerufen und der Empfaenger wird mittels einem Listener umgesetzt.

- Starten der Applikation

```
mvn clean spring-boot:run
-OR-
gradle clean bootRun
```
- Aufruf des Senders mittels Postman
Laden Sie in [Postman](#) die Datei `warehouse_demo2.postman_collection.json` und senden Sie den vorbereiteten HTTP POST Request.

1.5 Bewertung

- Gruppengröße: 1 Person
- Abgabemodus: per Protokoll, bei EK kann ein Abgabegespraech erforderlich sein
- Anforderungen "**überwiegend erfüllt**"
 - Implementierung der Kommunikation zwischen **EINEM** Lagerstandort und dem Zentralrechner (JMS Queue)
 - Ausgabe der empfangenen Daten am Zentralrechner (Konsole oder Log-Datei)
 - Beantwortung der Fragestellungen
- Anforderungen "**zur Gänze erfüllt**"
 - Zusammensetzung der Daten aller Lagerstandorte in einer zentralen JSON/XML-Struktur
 - Implementierung der REST Schnittstelle am Zentralrechner
- Erweiterte Anforderungen **überwiegend erfüllt**
 - Implementierung der Kommunikation mit **MEHREREN** Lagerstandorten und dem Zentralrechner
 - Logging der Daten bei allen Lagerstandorte und dem Zentralrechner
- Erweiterte Anforderungen **zur Gänze erfüllt**
 - Rückmeldung des Ergebnisses der Übertragung vom Zentralrechner an den Lagerstandort (JMS Topic)

1.6 Fragestellung für Protokoll

- Nennen Sie mindestens 4 Eigenschaften der Message Oriented Middleware?
- Was versteht man unter einer transienten und synchronen Kommunikation?
- Beschreiben Sie die Funktionsweise einer JMS Queue?
- JMS Overview - Beschreiben Sie die wichtigsten JMS Klassen und deren Zusammenhang?
- Beschreiben Sie die Funktionsweise eines JMS Topic?
- Was versteht man unter einem lose gekoppelten verteilten System? Nennen Sie ein Beispiel dazu. Warum spricht man hier von lose? "

1.6 Links & Dokumente


- Grundlagen Message Oriented Middleware: [Presentation](#)
- Middleware:
 - [Apache ActiveMQ Installationspaket](#)
- Apache ActiveMQ & JMS Tutorial:
 - <http://activemq.apache.org/components/classic/documentation>
 - <https://spring.io/guides/gs/messaging-jms/>
 - <https://medium.com/@malshine/activemq-getting-started-with-springboot-a0c3c960356e>
 - <http://www.academictutorials.com/jms/jms-introduction.asp>
 - <http://docs.oracle.com/javasee/1.4/tutorial/doc/JMS.html#wp84181>
 - <https://www.oracle.com/java/technologies/java-message-service.html>

<http://www.oracle.com/technetwork/articles/java/introjms-1577110.html>
<https://spring.io/guides/gs/messaging-jms>
<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-messaging.html>
<https://dzone.com/articles/using-jms-in-spring-boot-1>

Edit submission

Remove submission

Submission status

Submission status	Submitted for grading		
Grading status	Graded		
Time remaining	Assignment was submitted 25 days 16 hours late		
Last modified	Tuesday, 16 January 2024, 4:11 PM		
File submissions	<div><div> Warehouse2.pdf</div><div>16 January 2024, 4:11 PM</div></div>		
Submission comments	<div>▶ Comments (0)</div>		

Feedback

Grade	zur Gänze erfüllt		
Graded on	Tuesday, 21 November 2023, 4:50 PM		
Graded by	<div><div>MT</div>Micheler Thomas</div>		

Feedback comments

◀ MidEng 7.1 Warehouse REST & Dataformats [GK] – 4h

Jump to...

MidEng 7.3 Spring Boot using gRPC Framework [EK] ▶