How to make Sender of App PTP Model sending message continuosly to MOM software

=> we can take the support of scheduling concept.. For that we need to add @Scheduling annotation on the b.method of Sender App.

note:: @Scheduling can not be applied on the method with args.. So make sure that ur

b.method is designed having no args/params.

```
@Component
public class ActiveMQMessageSender {
@Autowired
private JmsTemplate template;
@Scheduled(cron="*/10
public void sendMessage() {
**")
//using LAMDA style anonyomous inner class
template.send("testmq1",ses-> ses.createTextMessage("From sender at"+new Date()));
System.out.println("Message sent");
}//run }//class
```

note:: while developing sender and Reciver Apps .. placing @EnableJms on the top of main class is optional.

Developing ActiveMQ Pub-sub model App

==========

=>All are same in both Sender/Publisher App and

also in Reciever/Subscriber App but change

spring.jms.pub-sub-domain value to true

in application.properties

```
spring.jms.pub-sub-domain=true
```

✓MS BootJMSProj2-PublisherApp-PUB-SUB [boot]

> Spring Elements

#src/main/java

com.nt

> BootJmsProj1ProducerAppApplication.java

com.nt.publisher

> ActiveMQMessagePublisher.java

M5 BootJMSProj2-SubscriberApp-PUBSUB [boot] >Spring Elements

#src/main/java

com.nt

> BootJmsProj1 RecieverAppApplication.java com.nt.subscriber

> JmsMessageSubscriber.java

#src/main/resources

application.properties

src/test/java

#src/main/resources

application.properties

> #src/test/java

>

JRE System Library [JavaSE-11]

>

JRE System Library [JavaSE-11]

>

Maven Dependencies

>

Maven Dependencies

> src

>

src

>

target

> target

WHELP.md

mvnw

mvnw.cmd

Mpom.xml

w HELP.md

mvnw

mvnw.cmd Mpom.xml

Order of execution

==========

=>Run subscriber App for multiple times (more than 1 time) (To give the feel more subscribers) =>Run publisher App

=> Check the console windows of subscribers application

topic1

2

1

(web console screen /page of active MQ)

Send To Active Subscribers

Active Producers

Delete

(It is like satilite TV channel broad casting system)

Keypoints on pub sub model

==========

(a) One message published by publisher can be consumed by more than one subscriber

(b) The subscribers must done their subscription before publisher publishes the message

(c) The message published by publisher will be duplicated and will be delivered to multiple subscribers

(d) The Publisher and Subscribers all must be in active mode at a time.

(e) In this model the Destination name is Topic.

(f) Once the subscriber consums the message the message from Topic destination of MOM s/w will not be deleted.

Can we send java object data over the network?

a) yes, by making the object as Serializable object.

process of

note: Serializing object means the converting object data into stream of bits -bytes that are required to send data over the network or to write to destination file.

How does normal object become Serilizable object?

=> By making the class of the object as the Serializable class .. (class must implement java.io.Serializable(I))

an

java.io.Serializable(1) is empty interface (marker interface) then how does it makes its impl class object as the Serializable object?

Ans) Marker Interface does nothing directly.. By seeing the marker interface implementation the underlying JVM /Server/Framework /Container provides special runtime capabilities to implementation class objs they create.

=> if JVM

data

create object for normal class.. that object data is in java format and can not be converted to stream of bits and bytes i.e we can not send/write normal object over the network or to a file.

Student obj

101 (Normal object)

raja

hyd 45.66

=> if JVM is creating object for Serializable class (class implementing java.io.Serializable(1)) then the JVM provides capability to the object to convert its data as stream of bits and bytes when needed to send the object's data over the network or to write to a file.

(DeSerialization)

converting bits and pytes into java notation

bits

Student obj

101 (Serializable obj)

send over the

and bystés

data and creating obj having data

raja

we can get

network

hyd

(serialization)

45.66

bits and bytes form of object data

File

write to the file

bits

and bytes

=>Most of marker interfaces are empty interfaces. but we can not say every empty interface

is marker interface.

=> We can develop our own custom marker interface .. but we need to create custom Container to provide runtime capabilities to the impl class objs of custom marker interface.

other marker interfaces are

=====

==========

To decide wheather interface is marker or not...

java.io.Serializable(1)

java.lang.Cloneable(1)

java.rmi.Remote(I)

Repository(1) of spring data jpa java.lang.Runnable (not empty)

and etc..

do not take emptiness as the criteria.. Take wheather

the underlying JVM/container/server/Framework/... providing special runtime capabilities to the impl class object or not.

Developing ActiveMq PTP application to send Object as message

=> since Sender and Reciever Apps can be there in two different machines of a

network or there is possibility of running MOM software on different machine so we need

to take the object as Serializable object..

Sender App/Publisher App java obj

network

MOM software Destination(Queue/topic) java.obj

network

Reciver App/Subscriber App

bits &bytes

java.obj

as

bits &bytes

Serializable obj

(Serializable obj),

(Serializable obj)-

Sender App code

step1) create spring boot project adding activemq, lombok dependencies

step2) Develop the Java bean class as the Model class.

//Model class

==========

```
package com.nt.model;
import java.io.Serializable;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ActorInfo implements Serializable {
private Integer actorId;
}
private String actorName;
private String actorAddrs;
```

step3) Develop the Sender App by Enabling Scheduling

//Sender class

```
package com.nt.sender;
import org.springframework.beans.factory.annotation.Autowired; import org.springframework.jms.core.JmsTemplate;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;
import com.nt.model.ActorInfo;
@Component
public class ObjectMessageSender {
@Autowired
private JmsTemplate template;
```

```
@Scheduled(cron = "0/20 * * * * *")
public void sendObjectDataAsMessage() {
//prepare object
ActorInfo actor=new ActorInfo(1001, "ranveer", "mumbai"); //send object as the message
template.convertAndSend("obj_mq1", actor); System.out.println("Object is send as Message ");
The SimpleMessageConvertor class contains logic
to convert given Serializable object into JMS style Message object.
}
}
```

main class

```
@SpringBootApplication
@EnableScheduling
@EnableJms
public class BootJmsProj3SendingObjectSenderAppApplication {
public static void main(String[] args) {
SpringApplication.run(BootJmsProj3SendingObjectSenderAppApplication.class, args);
}
}
```

step4) Add properties in application.properties file

```
MOM connectivity Details
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin
#enable PTP communication
#true enables pub-sub model and false enables ptp model
spring.jms.pub-sub-domain=false
# make all packages as the trusted packages (especically model class pkg) |
# to send/recieve model class object data as message to MOM software
spring.activemq.packages.trust-all=true
```

if we do not place this entry

then there is a possibility of

```
# or
#spring.activemq.packages.trusted=com.nt.model
```

step5) Makesure Activem MQ software (MOM software) is in running mode

use E:\ActiveMQSoft\apache-activemq-5.16.3\bin\win64\activemq.bat file

step6) Run the Send App ...

Reciver App Code

step1) create spring boot project adding activemq, lombok dependencies

step2) Develop the Java bean class as the Model class.

//Model class

(copy from sender App)

==========

```
package com.nt.model;
import java.io.Serializable;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok. NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ActorInfo implements Serializable {
private Integer actorId;
}
```

step3)

```
private String actorName;
private String actorAddrs;
```

getting the following error

↓

This class is not trusted to be serialized as ObjectMessage payload. Please take a look at http://activemq.apache.org/objectmessage.html for more information

on how to configure trusted classes.

Add properties in application.properties file

```
MOM connectivity Details
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin
#enable PTP communication
#true enables pub-sub model and false enables ptp model
spring.jms.pub-sub-domain=false
# make all packages as the trusted packages (especically model class pkg)
to send/recieve model class object data as message to MOM software
#
spring.activemq.packages.trust-all=true
# or
```

#spring.activemq.packages.trusted=com.nt.model

step4) Deveop the ReciverApp as JMSListener...

package com.nt.reciever;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jms.annotation.JmsListener;

import org.springframework.jms.core.JmsTemplate;

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.stereotype.Component;

import com.nt.model.ActorInfo;

@Component

public class ObjectMessageReciever {

@JmsListener(destination = "obj_mq1")

public void consumeObjectDataAsMessage (ActorInfo actor){

System.out.println("Recived Object Data ::"+actor);

}

}

step5) Run the Reciver App ...

Limitation of JMS

a) JMS is java language dependent technology i.e we need to develop both sender and reciver App

in same java language.. (JMS can not be used outside of Java Domain)

b) JMS based MOM softwares can recieve and send messages only by using protocol TCP i.e we can not use

other than protocol TCP like http, smtp and etc..

c) There is a possibilty of loosing data /message if the MOM software is down or MOM software is not responding

while publisher or sender is sending the messages

d) if the Message is very big / large scale then MOM software behaves very slow (i.e gives the performance issue)

e) There is no ability of creating multiple instances of single MOM software .. if multiple senders/publishers are sending messages

simultaenously the Performance of single copy MOM software may not Suitable for industry needs.

support

(It indirectly say JMS style MOM software does not "horizontal scaling.. it supports only vertical scaling)

MOM software

Cluster

MOM Software

MOM software

Veritical scaling

(Increase its capacity

by adding more RAMs,

MOM software

Horizontal scaling

CPUs, HDD to computer)

Conclusion:: use JMS style messaging only when no.of message are less and no.of senders/publishers are less towards

sending messages. In other situations prefer using kafka..