Messaging (Communication using Messages)

=========:

========

The Client Server Communication is of two types a) Synchronous communication

b) Ashynchronous Communication

a) Synchronous communication

==========

So far we have seen request-response model communication in web applications and method calls based communication in Distributed Apps.. The Messaging programming speaks about Communication using Messages.

(request -response or mehtod calls

based communication is synchronous communication by default)

In this communication, the client App can send next request to Server App has come back. only after the response for the already given request i.e The Client App should wait for given request related response from server App in order to make request to server App or to perform some client side operations.

next

Client App

Server App

request1

(1)

(2)

networ

response1 (3)

request2

(4) (or)

(4)

client side

operations

(collectiong inputs

from keyboard or

reading data from files)

in synchrnous communication .. both Client App and server

App must be active at time

Here the Client App is blocked to generate

next request or to perform client side operations the

until given request related response comes

back from server App.

note:: By default all Client-server communications are synchronous communincations

eg:: gmail login form submission gives gmail InBox.. until this Inbox page comes user can not any operations in the gmail (indicates the synchronous Communication)

b) Ashynchronous Communication

=>Here Client App is free (not blocked) to generate the next request

or to perform client side operations with out waiting for the given request related response from server App..

Client side operations (4)

Client app

Server App

request1 (1)

(2) req1 processing

next request (2)

eg: The way we use

response1 (3)

gmail inbox links and chat box parellel comes under asynchronous communication

(2)

Client side operations

get

=>In web applications we can use AJAX (Asynchronous Java Script and Xml) to asynchronous communication b/w browser(Client) and web application(server) (now ajax is part of UI Technologies)

=> we can use "Messaging concept" (Messages based communication) to get Asynchronous communication b/w two software Apps /comps.

(App1/comp1) SEnder App

msg

sending ths!

MOM(Message Oriented Middleware)

(3) Acknowlegdement

(S/W App/Comp)

next ms (4)

(message store) Destination

(App2/Comp2) Reciever App

msg

(5)

(2)

(60) recives msg

msg

(s/w App /comp)

jquery,angular js, react js and etc..

(This our main topic)

note:: In java domain, we can implement messages based communication using JMS or apache kafka

In Messages based communication

there will not be any method calls

or any request-reponse model interactions

(purely it is messages based communication)

(Messages based Asynchornous Communication)

=> MOM is software that acts middle man for both sender and reciever and it contains memory called Destination to hold messages sent by sender App and to deliver the same messages to the Reciver App

note :: In Messaging the messages will go from one App to another App in continuos flow like stream. note :: SMS messaging, whatup messaging, mailing does not come under messages based communication becoz that deals with person to person communication.. the actual messaging takes place b/w two software Apps or comps.

use-cases for Messaging ::

==========

a) Data streaming based on Scheduled /continous Flow

(eg:: uber cab availability status, Goods delivery App store availabiity status, Live Train Running status, Live cricket score status and and etc..)

b) Web activities and serach results

(eg: The advertisement agety App gathers google search queries continuosly to arrggate search queries)

c) Log Aggregation

(eg:: Collecting log messages generated by Production env. App and aggregrating their those messages)

Car manifacturing company

(1 shift)

Accounts Dept (9 am to 5 pm)

Manifacturing dept

(3 shifts) (24/7)

Management (No timings)

(No timings)

Cricket stadium

MOM

Destination

Since all departments related apps/modules can not be active

at a time, So synchronous communication among departments is not possible.. So prefer messages based Asynchronous communication.

Inventory Dept (10 am to 10 pm) (2 shifts)

ICC Cricket Score Live Stream

(another use-case)

====

score

ICC App (Rest Comp)

every 30 secs

MOM

=>The Cricket stadium Live Game Monitoring App

continuosly sends score as the Messages through MOM

to ICC App(Rest comp) and this App sends to cricket websites as messages through MOM

CrickBuzz

crickinfo

destination

espn.com

This Messaging based communication can be done in two ways

a) as Point to Point (PTP) communication

b) as Publisher and subscriber (Pub -sub) communication)

a) as Point to Point (PTP) communication

=>Here each message send to destination by sender will have only one consumer/Reciver i.e

once consumer consumes the message the message will be removed from the destination.

MOM

Sender App

reciver App

destination

sends the msg

(1)

Here the message deliver mechanism is 1 to 1

(3)

revcives the msg

(4) Message deletion

b) as Publisher and subscriber (Pub -sub) communication)

MOM

(One message can be deliverd to one consumer)

Real life example:: sender ---> post office ----> reciever letter

Subscriber1

(1) subscribe

Destination

publisher

sends (2) the msg (publish)

delivers (3)

Subscriber2

(1)

Here the message deliver machanism is 1 to Many

subscribe

subscriber1

delivers (3)

Real life example

Subscriber3

TV channel ---------> Broading system (TV Program)

subscriber2

(1)

subscriber3

subscribe

delivers (3)

=> Here each messages sent by the publisher will

go multiple subscribers who have done their subscription

to destination before publisher publishes the message. (each message can have 0 or more subscribers)

=> By Default all Client app -server app communication is synchronous communication

eg:: browser to web application, Rest Client to Rest Service, Ms Client to Ms, MS to MS.

=> In web application to get asynchronous communication b/w browser and web application take the support of AJAX.

=> Between two Java Apps or two MS or Rest Client And RestService if u r looking for Messages based asynchronous communication then for Messaging/Message Quees with the support of MOM software.

=> Messaging/Message Queue can be implemented using two types of porotocols in Java a) Using Basic MQ Protocol (BMQP) :: For this we need to use JMS

b) Using Adavanced MQ protocol (AMQP) :: For this we need to use RabitMq, Apache kafka and etc..

==========

Messaging/Message Queue using JMS

==========

jakarta

(JMS :: Java Messsaging Service)

Java Mail Api is different from JMS

=> JMS is the software specification given by Sun Ms in JEE module having set of rules and guidelines

to provide messsages based communication b/w java comps or Apps. by connecting to MOM software.

Sun Ms/Oracle corp

JMS (specification having rules and guidelines)

MQ: Message Queue

BMQP :: Basic Message Queue Protocol

AMQP :: Advanced Message Queue Protocol

=>Java mail api is given for mailing operations

=> JMS (Java Messaging Service) is given interaction b/w App to App using Messages

Vendor1

Active MQ

(MOM) (Best)

vendor2

weblogic MQ (MOM)

for messaging based communication

The implementation software

Vendor3 Wildfly MQ (MOM)

of JMS specification is technically called MOM software.

=> Each Implementation software of JMS given by different vendors is called one MOM software

=> Every Application server s/w like weblogic, wildfy and etc.. gives one built-in MOM software

=> Tomcat is not providing any MOM software..so if ur application is using tomcat server then prefere working with Active MQ as the MOM software.

an

active MQ is independent

MOM software (not part of any

=> JMS supports both PTP and pub-sub Models of Messaging

the

=> The MiddleMan software between send and reciver/consumer who takes messages given

software

by sender and holds them for Reciver to come and cosume is called MOM

web server / app server s/w)

=> MOM contains destinations as the logical mememories to recive and hold the messages..

In PTP model this destination is called "Queue" and in pub-sub model this destination is called "Topic"

if the App is sending/recieving the messsage to/from Queue Destination then we can say

we are dealing with PTP model

reciver App

if the App sending /recieving messages

to/from Topic Destination then we can say we are dealing with pub-sub model

MOM

Sender App

destination (Queue)

sends

(1)

the msg

(3)

revcives the msg

PTP communication

Once the message is received by

Reciever App then will be deleted from the Destination

Subscriber1

MOM

(1) subscribe

Destination (Topic)

publisher

sends

(2)

the msg

(publish)

delivers (3)

Subscriber2

(1

subscribe

pub-sub model

delivers

(3)

Subscriber3

(1)

subscribe

delivers

(3)

note:: The JDBC API/technology based impl softwares are called JDBC driver s/ws note:: The JMS API/technology based impl softwares are called MOM s/ws

=> JMS provides api representing rules and guidelines in the form of interfaces,classes placed in javax.jms or jakarta.jms and its subpkgs. (It is like JDBC API)

(old)

(new)

=> Active MQ, weblogic MQ, WildlfyMQ and etc.. are MOM softwares which are internally providing impl classes for JMS API intefaces

(These are like JDBC DRIVERS)

=>Programmers takes impl classes through JMS interrfaces .. to create objects and

to consume services [The Sender and reciver Apps will be developed using JMS Api referring one or another MOM Impl sofwares) (These are like Java Apps using JDBC drivers through JDBC API)

sample reference code to understand API, Impl software and App development using API

====

interface Operation1{

public void process();

It is like JMS API given

(Party1 :: Technology rules and

}

interface Operation2{

by Sun Ms (JMS specification) Technology

guidelines creator as technology API)

public String process(String msg);

}

[Course brouhers having topic names]

public class Operation1Impl implements

public void Process(){

=>mpl classes provided by the Vendor Operation1

....

(It is like active Mq, weblogic mq and etc..

(Party2:: Software Vendor company

}

MOM softwares given by different Vendors

who giving impl software based on the technology rules and guidelines)

}

public class Operation2Impl. implements

public String Process (Stirng msg){ Operation2

[These are like faculties who made them selves ready

for conducting classes for course broucher topics)

}

}

In Sender/Reciever App developmet

Operation1 op1=new Operation1Impl();

It Like JMS application

op1.process();

devleoped the Programmer

Operation2 op2=new Operation2Impl();

(JMS sender or Reciever app)

String result= op2.process("hello");

Functional Interface

(It is like Student who uses Faculties services through course broucher topic names)

Party3 :: Developer who develops the Sender /Reciever app

In any java technology there will be 3 parties party1 :: Technology specification as technology API party2 :: Vendor companies who gives impl softwares based on the technology Rules and guidelines part3 :: Developer who develops the software apps using implementation softwares for completing the task

==========

=>The interface that contains only one abstract method direclty or indirectly is called

functional Interface.

@FunctionalInterface

interface Operation1{

}

Impl1 ::

public void process(String msg);

Functional interface

===== public class Operation1Impl implements Operation1{

public void proess(String msg){

}

}

Impl2:: anonymous inner class

(inline impl)

======

Normal Impl class

Every java technology API will be used in two angels

a) Vendor companies use technology API rules and guidelines to develop the implementation software s

b) Developers use same technology API as base to devleope apps towards completing the task

Operation1op1= new Operation1(){ public void process(Stirng msg){

}

};

Here Anonymous inner class is created implementing Operation1 interface and process(-) is implemented in that class.

More over object is created for anynomous inner (class and object is refered by interface ref variable (anonymous inner clss Impl class obj + method impl)

Impl3 :: Lamda based anonymous inner class

(InLine impl)

Here 3 operations are happena

(a) anonymous inner class is created implementing Operation1(1)

(b) In that anonymous inner class process() method is implemented

'c) Object is created for the anonymous inner class

and that is refered with Interface ref variabl

Operation op1= (msg)->{ ....

spring

}

all the 3 operations of impl2 is done..but the code is simplifed.

The JMS api is providing one Funcational Interface called "MessageCreator" as shown below

@FunctionalInterface

This session obj represents the connectivity b/w

public interface MessageCreator {

Message createMessage(Session session);

}

Impl1 (Using mous innner class)

sender or reciver app and MOM software (Session object in JMS is like JDBC con object)

MessageCreator mc-new MessageCreator(){

public Message createMessage(Sesssion ses){

}

}

....

// logic

return message obj;

Impl2 (Using Lamda anonymous inner class) (Simple and Best code)

MessageCreator mc=(ses)->{ .....

(or)

...

return message obj;

}

MessageCreator mc-ses-> message obj; (if method definition contains single line)

technology

SpringBoot JMS/spring JMS provides abstraction on plain JMS to simplify the messages based communication

with the support of "JMSTemplate" (template method DP)

=======

Procedure to keep ActiveMq as MOM software

====================

==============================

step1) Dowload ActiveMQ software as zip file from Internet
https://activemq.apache.org/components/classic/download/

ActiveMQ 5.17.4

Release Notes | Release Page | Documentation | Java compatibility: 11+

Windows

apache-activemq-5.17.4-bin.zip

SHA512

GPG Signature

Unix/Linux/Cygwin

Source Code Distribution:

apache-activemq-5.17.4-bin.tar.gz

SHA512

GPG Signature

activemq-parent-5.17.4-source-release.zip SHA512

GPG Signature

step2) Extract the zip file to a folder.

step3) start the ActiveMQ software

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

step4) open admin console page of Active MQ software

http://localhost:8161

username: admin

password: admin

step5) Observer Topic and Queue sections in admin console page

home page ---> manage active mq ...>

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

Spring JDBC provides abstraction on plain JDBC technology Spring ORM/ spring data jpa provides abstraction on Hibernate framework Spring MVC provides abstraction on servlet,jsp technologies Similarly Spring JMS provides abstraction on Plan JMS technology

=>All template classes in spring /spring boot takes care of common logics and lets the developers to develop only application specific logics.. all these Template classes are based on "template method" method design Pattern eg: JdbcTemplate, RestTemplate, Hibernate Template, JMSTemplate and etc..

Procedure to develop JMS PTP (Queue) application using active MQ MOM software

add

=>In any JMS App (Producer /Sender or Reciver /Subscriber/consumer) once we spring-boot-starter-activemq starter as dependency we get JMSTemplate class object through AutoConfiguration that can be injected to

Sender / Reciver App through Autowiring.

For Producer App

====

(@Autowired)

step1) create spring boot project adding active mq starter..

Service URL

https://start.spring.io

Name

Use default location

Location

BootJMSProj1-ProducerApp

G:\Worskpaces\Spring\NTSPBMS714-BOOT\BootJMSProj1-Produ Browse

Type:

Maven

Packaging:

Jar

Java Version:

11

✓ Language:

Java

Group

nit

Artifact

BootJMSProj1-ProducerApp

Version

0.0.1-SNAPSHOT

Description

Messaging-JMS

Package

com.nt

Working sets

Add project to working sets

New...

Working sets:

Select...

<dependency>

<groupId>org.springframework.boot</groupId>

active mq starters

in pom.xml looks like this

<artifactId>spring-boot-starter-activemq</artifactId>

</dependency> step2) add the following properties in application.properties file

 # MOM connectivity Details #8161 for admin console, 61616 for actual MOM service

spring.activemq.broker-url=tcp://localhost:61616

spring.activemq.user=admin

spring.activemq.password=admin

#enable PTP communication

JMSTemplate,JdbcTemplate,RestTemplate

JndiTemplate and etc..

classes are given based on Template method design pattern which says the common logics of certain task will be taken care internally and only specific activies should be taken care by the programmers

#true enables pub-sub model and false enables ptp model

spring.jms.pub-sub-domain=false

step3) Develop runner class as the Message sender

tion

=>JmsTemplate class is having send(-,-) method taking destiona (queue/topic) logical name(generally new name)

and MessageCreator(I) (Functional interface).

public void send(String

destination, MessageCreator messageCreator) throws JmsException

Runner class having sender logic

package com.nt.sender;

import java.util.Date;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.Session;

For this we can pass either anonymous inner class obj

or Lamda style inner class obj

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

import org.springframework.boot.CommandLine Runner;

import org.springframework.jms.core.JmsTemplate;

import org.springframework.jms.core.MessageCreator;

import org.springframework.stereotype.Component;

@Component

✓ BootJMSProj1-ProducerApp [boot]

> Spring Elements

#src/main/java

>com.nt

com.nt.sender

> ActiveMQMessageSenderRunner.java

#src/main/resources

application.properties

> #src/test/java

> JRE System Library [JavaSE-11]

> Maven Dependencies

> src

>

target

w HELP.md

mvnw

mvnw.cmd

M pom.xml

public class ActiveMQMessageSenderRunner implements Command Line Runner { @Autowired

private JmsTemplate template;

/*

@Override

public void run(String... args) throws Exception {

//using anonymous inner class logics

template.send("testmq1", new MessageCreator() {

@Override

public Message createMessage(Session ses) throws JMSException {

Message message=ses.createTextMessage("From Sender at ::"+new Date());

return message;

}); */

(or)

/* using LAMDA style anonyomous inner class

template.send("testmq1",ses->{

return ses.createTextMessage("From sender at"+new Date());

});*/

}//run

}//class

(or)

//using LAMDA style anonyomous inner class

template.send("testmq1",ses-> ses.createTextMessage("From sender at"+new Date()));

System.out.println("Message sent");

step4) Make sure that Active MQ started

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

step5) Run the Sender App and ActiveMQ Server console

Queues:

Name

Number Of Pending Messages Number Of Consumers

testmq1

1

0

2

Messages Enqueued Messages Dequeued Views

Browse Active Consumers

Active Producers

atom rss

Procecedure to develop Consumer/Reciever App of PTP model using ActiveMQ MOM software

step1) create spring boot project adding active mq starter..

=====

Service URL

https://start.spring.io

Name

BootJMSProj11-ReciverApp

Use default location

Location

G:\Worskpaces\Spring\NTSPBMS714-BOOT\BootJMSProj1-Produ Browse

Type:

Maven

✓ Packaging:

Jar

Java Version:

11

Language:

Java

Group

nit

Artifact

BootJMSProj1-ProducerApp

Version

0.0.1-SNAPSHOT

Description

Messaging-JMS

Package

com.nt

Working sets

Add project to working sets

New...

Working sets:

Select...

<dependency>

<groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-activemq</artifactId>

Operations

Send To Purge Delete Pause

</dependency>

step2) add the following properties in application.properties file

#MOM connectivity Details #8161 for admin console, 61616 for actual MOM service

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

step3) Develope the java class having @JMsListener method as shown below.

package com.nt.reciever;

import org.springframework.jms.annotation.JmsListener;

import org.springframework.stereotype.Component;

@Component

public class JmsMessageConsumer {

@JmsListener(destination = "testmq1")

public void readMessage(String text) {

executes automactically

This method

to read

message

from queue destination

System.out.println("Recived Message::"+text);

By seeing the @JMSListener annotation

the method readMessage(-) becomes

callback and to read and display the message

method to connect to given destination

✓MS BootJMSProj1-RecieverApp [boot]

Spring Elements

src/main/java

✓ com.nt

(becoz of @JMSListener)

step4) Run the App Consumer App

(reads the message from Quue destinaiuon)

step5) Observe the console

Queues:

Name Number Of Pending Messages Number Of Consumers

on

testmq1 0

Keypoints PTP model messaging

0

(a)

The

destination name is "Queue". (FIFO rule)

2

› BootJmsProj1 RecieverAppApplication.java

com.nt.reciever

> JmsMessageConsumer.java

#src/main/resources

application.properties

>src/test/java

> JRE System Library [JavaSE-11]

Callback methods methods will be

automatically

>

Maven Dependencies

> src

> target

w HELP.md

mvnw

mvnw.cmd

M pom.xml

Messages Enqueued Messages Dequeued

2

Views

Browse Active Consumers

atom rss

Active Producers

(b) Both Sender and reciver need not be active at a time

(c) One Message will have only one Reciever/Consumer

(d) The Queue Destination can send message to reciever App who connected to the Destination before Sender sends the message.will also the recieve the message when the sender sends the messsage.

(e) if multiple consumer are waiting for a message sent by the *SEnder then only first reciver the recieves the* ***message.***

(f) The queue destination delivers the message to Reciever App and deletes the message.

usecase:: our car factory usecase needs this model (PTP) communication