Date : 21-May-21
Spring Boot 9AM
Mr. RAGHU

JMS : Spring Boot

JMS - Java Message Service
=> It is used for continioues data flow between two system using mediator.
=> 2 Applications
   a. Producer Application
        JmsTemplate#send(destinationName,messageCreator)
   b. Consumer Application
        @JmsListener(destination="")
        read message

=> These two applications communicates with each other using TCP prtocol

------Step#1 -- ActiveMQ Setup----------

Download : https://activemq.apache.org/components/classic/download/
> click on OS based link : Windows apache-activemq-5.16.2-bin.zip
> Extract to a folder ex: 'apache-activemq-5.16.2'
> Open folder location '\apache-activemq-5.16.2\bin\win64'
> Click on activemq.bat file
> Enter URL : http://127.0.0.1:8161/admin
  Un : admin
  Pwd: admin
** check \apache-activemq-5.16.2\conf\users.properties

> press ctrl+C to terminate batch.

------Step#2 --Producer Application----------

Name : SpringBoot2JmsProducer
Dep : Spring For Apache ActiveMQ5

*) application.properties
```
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin
#P2P Communication
spring.jms.pub-sub-domain=false
```

*) Producer Service
```
package in.nareshit.raghu.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.stereotype.Component;

@Component
public class ProducerSevice {
```

```
        @Autowired
        private JmsTemplate jt;

        public void send(String destination,String message)
        {
                jt.send(destination, (ses)-
>ses.createTextMessage(message));
        }
}

*) Runner class
package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.service.ProducerSevice;

@Component
public class TestProducerRunner implements CommandLineRunner {

        @Autowired
        private ProducerSevice ps;

        public void run(String... args) throws Exception {
                ps.send("my-q1", "Hello Data Sample");
                System.out.println("SENT...");
        }

}


------Step#3  -- Consumer Application ------------------------------
--------
Name : SpringBoot2JmsConsumerEx
Dep  : Spring For ActiveMQ5

*) application.properties
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin
#P2P Communication
spring.jms.pub-sub-domain=false

*) At starter class: @EnableJms

*) Consumer:
package in.nareshit.raghu.consumer;

import org.springframework.jms.annotation.JmsListener;
import org.springframework.stereotype.Component;

@Component
public class MessageConsumer {

        @JmsListener(destination = "my-q1")
```

```
        public void readMessage(String message) {
                System.out.println("At Consumer!" +message);
        }
}


--Execution Order--------------------
1. Start ActiveMQ
2. Consumer App
3. Producer App
=================================================================
*) Notes:
*) If we use spring.jms.pub-sub-domain=false then code behaves like
P2P
  communicate, if it is set to true then it is PUB/SUB Communication.

---------------------------------------
Functional Interface
Lambda Expression :  (methodParam) -> {methodBody};


interface MessageCreator {
 Message createMessage(Session session) ;
}
--Normal Impl class----
class A implements MessageCreator {
   Message createMessage(Session session) {
        TextMessage tm = session.createTextMessage("Data");
        Message m = tm;
        return m;
   }
}

MessageCreator mob  = new A();
----------------------------------------
Lambda Expression:
MessageCreator mob = (session) ->   {
                        TextMessage tm =
session.createTextMessage("Data");
                        return tm;
                    };
Simplified One:
(ses) ->  ses.createTextMessage("Data");

======Spring Boot AutoConfiguration looks
like================================================
package org.nareshittech.app.config;
//ctrl+shift+O
@Configuration
public class AppConfig {
        @Bean
        public ConnectionFactory connectionFactory() {
                ActiveMQConnectionFactory cf=new
ActiveMQConnectionFactory();
                cf.setBrokerURL("tcp://localhost:61616");
                cf.setUserName("admin");
                cf.setPassword("admin");
```

```
                return cf;
        }
        @Bean
        public JmsTemplate jmsTemplate() {
                JmsTemplate jt=new JmsTemplate();
                jt.setConnectionFactory(connectionFactory());
                return jt;
        }
}
```
==========