**Creare ActiveMQ(OLD UNUSED Technology)**

Va trebui dependenta

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

public static void main(String[] args) throws JMSException {

ConnectionFactory connectionFactory = new ActiveMQConnectionFactory("tcp://localhost:61616");

Connection connection = connectionFactory.createConnection();

Session session = connection.createSession(false,Session.*AUTO\_ACKNOWLEDGE*);

connection.start();

Queue queue = session.createQueue("ORDERS.Q"); //putem pune orice nume si . la fel

MessageProducer producer = session.createProducer(queue);

TextMessage textMessage = session.createTextMessage();

textMessage.setText("A test message");

producer.send(textMessage);

}

portul nu e cel de la consola din web!!!

- createQueue(“nume”) – creaza un queue in activeMQ si il putem vedea in web console

- createProducer(queue) – cream un producer de mesaje pentru queue data

- creareTextMessage() – cream un obiect cu text de trimist

Observatii:

Interfetele vin de la JMS, si ActiveMQ are implementarile lor:

ConnectionFactory -> ActiveMqConnectionFactory

Session

Queue

- send(TextMessage) – trimite textul la consumer

La rulare, vedem ca am primit un mesaj in queue:



Acum vom extrage un mesaj din queue:

public class Main {

public static void main(String[] args) throws JMSException {

ConnectionFactory connectionFactory = new ActiveMQConnectionFactory("tcp://localhost:61616");

Connection connection = connectionFactory.createConnection();

Session session = connection.createSession(false,Session.*AUTO\_ACKNOWLEDGE*);

connection.start();

Queue queue = session.createQueue("ORDERS.Q"); //putem pune orice nume si . la fel

MessageConsumer messageConsumer = session.createConsumer(queue);

TextMessage textMessage = (TextMessage) messageConsumer.receive();

textMessage.acknowledge();

System.*out*.println(textMessage.getText());

}

Mesajul a fost extras si scos din queue.

- createConsumer(queue) – creaza un consumer ce va lua mesajele din queue

- receive() – returneaza un obiect de tip Message, cu un mesaj extras din queue.

**receive()** - blocheaza threadul pana nu apare vreun mesaj. Anume el preia mesajul. Totusi, problema e ca el ia mesajul, dar nu confirma lui Broker ca mesajul a fost interceptat, si el nu e sters din queue. Pentru a confirma ca a fost primit, folosim metoda **acknowledge()**

* createQueue – daca o asa queue deja exista, o returneaza, daca nu, o creaza.

**Transacted Session**

* S-ar putea sa vrem sa trimitem mai multe mesaje, de ex 2, dar ele sa fie trimise ca un tot intreg.Deci, fie se trimit ambele, fie nici-una daca una da fail. Daca apare vreo problema la una, se da rollback
* 

Deci, avem aplicatia care trimite mesage si brokerul

Aici avem 2 mesaje.

Cand dam send() la primul, el inca nu e trimis, e doar in memorie stocat

Cand dam send la al 2, e la fel

Abea cand dam commit() ele ambele se trimit la broker

* Anume prin asta si e diferita tranzactia, prin faptul ca nimic nu se trimite pana nu dam commit.

Session session = connection.createSession(false, Session.*AUTO\_ACKNOWLEDGE*);

false inseamna ca sesiunea nu e tranzactionala!Pentru a o face tranzactionala, punem pur si simplu true

* Mai trebuie sa facem:

session.commit();

* Deci vom avea asa:

public static void main(String[] args) throws JMSException {  
 ConnectionFactory connectionFactory = new ActiveMQConnectionFactory("tcp://localhost:61616");  
 Connection connection = connectionFactory.createConnection();  
 Session session = connection.createSession(true, Session.*AUTO\_ACKNOWLEDGE*);  
 connection.start();  
  
 Queue queue = session.createQueue("MyQueue.Q");  
 MessageProducer producer = session.createProducer(queue);  
 TextMessage textMessage = session.createTextMessage();  
 textMessage.setText("O tranzactie");  
  
 producer.send(textMessage);  
  
 session.commit();  
}

* Citirea lor e la fel