**DEMO:**

**SJMS-100-ClientJSE:**

**Codice da spiegare:**

JndiUtils: (da configurazione a classi Java per Producer e Consumer)  
Producer: (bean spedito su destinazione)  
Consumer: (modalita consumo)  
Eccezioni: (validazione consumer => per messaggio poison)

**Demo protocollo (TC)**

**Caratteristiche Queue/Topic**- TC1: Durable (interrompere consumer e riprendere)  
- TC2:Consumo esclusivo(avviare piu thread consumer)  
- TC3:Consumo Polling (main/test classi)  
- TC4: Consumo ED (main/test classi)  
 **Acks singolo messaggio:**- TC5: Polling props ack (auto e client) (test in debug e verifica in DB)  
- TC6: ED props Ack (auto e client) (test in debug e verifica in DB)

**Transazioni**  
Prod: creano messaggi vuoti -> RE per messaggio vuoto  
Cons: props session>JmsClient   
  
- TC7: Polling RE Transazioni true/false  
RE ogni N messaggi in Consumer (verifica in DB rimossi dopo commit)  
RE ogni N messaggi in Producer (aggiunti in DB dopo commit)

- TC8: ED e RE (conf DeadLetterQueue)   
Verifica da UI in ActiveMQ.DLQ dopo 20 tentativi (sembra)  
Dettagli:  
<http://activemq.apache.org/message-redelivery-and-dlq-handling.html>

**Note:**Le classi di test devono essere interrotte a mano perche’ la ConnectionFactory ha un TP di  
connections da gestire che previene la chiusura del metodo main.

**Eccezioni:**Il Broker mette un messaggio in redelivery quando:

1. A transacted session is used and **rollback()** is called.   
   ( Caso di ED)
2. A transacted session is closed before **commit()** is called.  
   (caso ED viene chiuso prima di completare ricezione + ack.)  
   onMessage e’ cominciato ma la connessione viene chiusa prima di dare commit.
3. A session is using **CLIENT\_ACKNOWLEDGE** and **Session.recover()** is called.
4. A client connection times out (perhaps the code being executed takes longer than the configured time-out period).  
   Simile a 2)

**Code review:**Consumer possibilita:  
ActiveMQMessageConsumer.dipatch

**Forme di Ack: (da verificare in debug, quando viene rimosso dalla queue)**

**AUTO\_ACK**Il messaggio si considera ricevuto (e quindi cancellabile dal MOM) alla conclusione di  
1) onMessage del Listener  
2) receive del Poller  
  
Non Ack se:  
1) onMessage va in eccezione  
2) errore delivery del MOM sul metodo receive (non simulabile)

**DUPS\_OK\_REDELIVERED:**L’ autoack aggiunge un overhead dovuto alla verifica che la delivery non venga provata piu di  
una volta.  
Questa opzione di ack si basa sul fatto che verranno fatti N tentativi di consegnare lo stesso messaggio, e ci si aspetta che il consumer scarti quelli che ha gia ricevuto.

**CLIENT\_ACKNOWLEDGE:**L’ack viene dato esplicitamente da codice.  
Fino a che non viene ricevuto dal MOM, il messaggio si considera non ricevuto.  
E quindi verranno effettuati altri tentativi.

API: message.acknowledge()

**Transazioni JMS: (se transacted=true, default = false)**

**Scopo:** Raggruppare il destino di N messaggi tra loro.   
Tutti vengono spediti (prod) o ricevuti( cons) **Quando**:   
Raro dal lato consumer.   
Lato producer puo’ accadere che il send di un messaggio sia parte di una transazione per cui sia necessario che N operazioni si completino prima di poter dare la commit a N messaggi.  
  
**Come:**Ogni transazione e’ demarcata dalla commit successiva. Si attende una session.commit  
Alla commit:

**Producer:**   
il MOM salva i messaggi. (Prima se li tiene in cache e non si considerano acquisiti dal MOM)  
  
**Consumer:**Il MOM e’ autorizzato a cancellare gli N messaggi spediti tutti in una volta.

**Note Avanzato:**  
Sembra che se ci sono N consumer in polling, anche se non consumano, alcuni messaggi  
vengano riservati per loro. E’ possibile riprodurlo con consumer in polling che non partono, ed uno in ED che consuma solo parte dei messaggi.  
Ad un Restart di entrambi sembra che i messaggi riservati (non essendo stati acked) vengano poi ripresi dall unico consumer successivamente.

**SJMS-200-ClientSpring:**

* Stessi TC precedenti. Enfasi su Config
* JndiTemplate
* JmsTemplate
  + panoramica api
  + MessageConverters (impl da iniettare in JmsTemplate
* SessionAwareMessageListener

**SJMS-230-SpringJms: (vedi capitolo Apress 803-828)**

* Common con classi Spring jms:
  + JmsGateway (polling)
  + DefaultMessageListenerContainer (event driven)
* TC:
  + P: Configurazione session
  + ED: Consumers su thread diversi
* Panoramica altre classi Spring JMS:
  + SimpleMessageListenerContainer
  + DefaultMessageListenerContainer
* @Configuration
  + @EnableJms
  + @JmsListener
* Jms namespace

**SJMS-100-ClientJSE:**

**TC: Eccezioni, session.commit e autoack**

* Code: Session.commit
  + autoack=true
  + autoack=false
* eccezione prima di commit
  + autoack=true
  + autoack=false
* eccezione RuntimeException
  + spedire messaggio vuoto (dopo N corretti)
    - JmsConsumer.validateMessage (va in eccezione)

**Classi Spring JMS:**