**SJMS LABS:**

**STOCK MARKET TRADING SYSTEM:**- StockAgent Client spedisce ordini Stocks su StockExchangeSystem (BORSA)  
**-** StockExchangeSystem riceve ordini e li processa (log nei test)

**SA:**

**Avviare ActiveMQ:**cd C:\Servers\apache-activemq-5.14.3\binactivemq start **Accesso WebApp:**<http://127.0.0.1:8161/admin/>  
admin  
admin

**Creare Queues: (se non create, verranno create da App)**se.jms.queues.order  
se.jms.queues.orderReplyQueue

**SJMS-LAB-StockAgent:**Spedisce su **Queue se.jms.queues.order**Ascolta risposte su **se.jms.queues.orderReplyQueue**

**services:**

**StockAgentOrderProducer:**Spedisce Order come jms Messagesu **se.jms.queues.order  
  
OrderOutcomeProcessor**Logga risposte (OrderOutcome **)** StockExchangeSystem

**Jms:  
 StockAgentOrderOutcomeConsumer:** Da implementare come listener event driven su **se.jms.queues.orderReplyQueue** Delega elaborazione a **OrderOutcomeProcessor**

**StockAgentOrderOutcomePoller:** Logica come precedente. Esegue Polling su Queue.

**SJMS- LAB-StockExchangeSystem:**

**StockExchangeSingle:**

**Creare Consumer per** se.jms.queues.order:

1. Event Driven
2. OrderProcessor:

**EVOLUZIONE: (Da fare insieme)  
  
TAG CLUSTERED:**

1. **Performances JmsTemplate:**Usare CachingConnectionFactory  
   Verificare possibilita offerte da ActiveMq ConnectionFactory
2. **Dynamic Destinations:**Da creare in base a configurazione (N SA ognuno con (almeno) la sua queue di reply dedicata)  
   DestinationResolver. (JndiDestinationResolver, DynamicDestinationResolver)  
   jsmTemplate.send(destinationLogicName, message)  
     
   JndiDestinationResolver (nome jndi con cui e’ registrato)  
   DynamicDestinationResolver (nome risorsa (fisica))

**StockExchange:**

**Per ogni StockAgent amministra due Queue:**

**- IN:** OrderQueue  
Riceve ordini da questo operatore finanziario.  
-**OUT**: OrderReplyQueue (Accepted, Eseguito, Rejected)  
Manda messaggi in merito all aggiornamento status ordine.  
- Accepted/Rejected (alla ricezione)  
- Executed (quando domanda incontra offerta)

**Shared**

**- OUT: Topic per Op Fin registrati al servizio**:  
Notifiche ordine cambia quotazione corrente titolo. (

**StockAgent:**

Ha accesso readonly alle sue due Queue **- OUT:** OrderQueue  
Spedisce ordini (Generator)  
**- IN:** OrderReplyQueue (inseritoInBook, Eseguito)  
Riceve notifiche avanzamento status ordine

**TopicSubscriber:**Sottoscrive servizio notifica quotazioni

Messaggio (payload da SI)

**Headers:**

1. CorrelationId (IdCliente)
2. replyTo (nome Logico Cliente)
3. Ordini da processare entro
4. OrderType (custom header)

**SOLUZIONE SINGLE 1 STOCK AGENT: ( REQUISITI)**

**- IN:** OrderQueue  
Riceve ordini da questo operatore finanziario.  
  
-**OUT**: OrderReplyQueue (Accepted, Eseguito, Rejected)  
Manda messaggi in merito all aggiornamento status ordine.  
- Accepted/Rejected (alla ricezione)  
- Executed (quando domanda incontra offerta)

**SOLUZIONE CON N STOCK AGENTS: ( REQUISITI)**

**SCALABILITA: (Soluzione scalabile)  
  
Risorse da gestire**- Servers MOM  
- Server Gateway App (threads)

1. **JMS Brokers:** 
   1. 1 Shared
   2. Pool Shared
   3. Dedicato (con App locale o remota)
2. **Gateways:**
   1. 1 Shared
   2. Pool Shared
   3. Dedicato con MOM (locale, remoto)

**BORSA Jms Gateway per ogni SA**

* 1. Server dedicato
  2. MOM locale
  3. App Locale
  4. Validazioni
     1. formali
     2. business (servizi esterni)
  5. Send to **CommonInternalQueue** for processing

1. **BORSA Service**
   1. Riceve Da Shared **CommonInternalQueue**
   2. Salva e aggiorna N sistemi (complesso)
   3. Risponde su **QueueReply** Common
2. **BORSA ReplyRouter:**
   1. Riceve su **QueueReply** Common
   2. Routing logic
   3. Reply to Queue Dedicata
3. Code:
   1. JmsRegistry ( <SA-ID>, JmsTemplateProducer) e <SA-ID
4. Config:
   1. Shared JMS Resources
      1. N Connection Factories (N Servers)
      2. Incoming Order Queues (scale)
      3. StockEvent Topics
      4. MessageListeners
   2. Per ogni SA
      1. File di config (UnaSharedCF, OrderQueue, OrderReplyQueue)

**STEPS:  
- Identificare tutti gli oggetti e le loro dipendenze:**

1. Partire da config Queue (incoming) Order (TEST milestone)
   1. MLC
   2. ML
   3. Queue (da quelle shared)
   4. connectionFactory (da quelle shared)
2. Aggiungere possibilita di reply in ML (TEST completo)
   1. JmsTemplate (possibile shared?) -> no perche’ reply e’ locale al SA
   2. Default Destination
   3. CF
3. Gestione eccezioni
   1. Unparseable (Jms Message Id)
   2. Internal Exception (solo IN MVP)
   3. Formato messaggi
   4. Validazioni Business

**PERFORMANCES-NOTE:**

**JmsTemplate (sender):  
-** Sempre da usare con CachingConnectionFactory se no Connection, Session e producer vengono create e dismesse ogni volta.  
- Verificare le impostazioni del Broker e la gestione di ack, commit e asynch send.  
- E’ possibile un trade off tra Reliability e Performances (usare send asynch, in cui si spedisce senza aspettare ack di ricezione dal MOM)

**MLC: (Creato con un pool di threads).**- Cercare di mantenere al minimo la loro creazione.  
- Pesanti da creare e a runtime (numero di thread cresce)

**SOLUZIONE E TRADE OFF:**

**Esperienza:**- Piu semplice definizione e manutenzione infrastruttura di Incoming.  
- La gestione dei messaggi outbound comporta in genere oggetti dedicati (no singleton) e computazioni piu complesse per il routing.