Simulate a Simple Bond Trading Processing

# Overview

This system need to simulate a simple bond trading processing. We simplified the logic, so it might be not the same as the trading processing in the real world.

In this system, we suppose it is made up of 5 subsystems, front-end .Net UI system, front-end Java server, simulator, back-end trading service and messaging channel (JMS broker, such as ActiveMQ). The architecture diagram is shown as below.

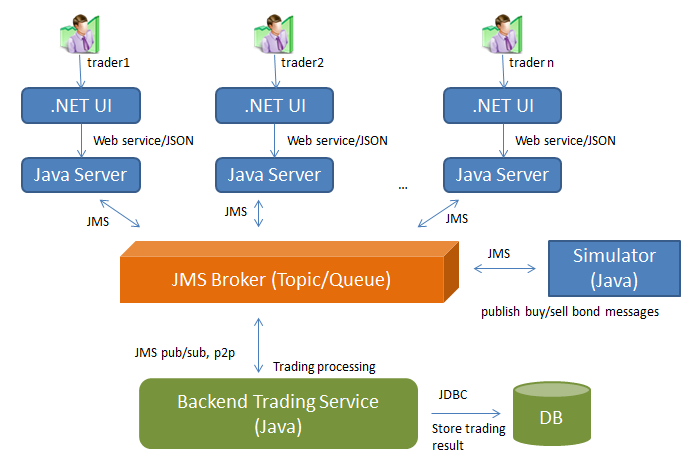


Figure 1: Architecture diagram

* Front-end .Net UI System
  + Traders can buy/sell bonds on this UI system.
* Front-end Java Server
  + This subsystem can handle all requests from UI side and talk with other subsystems via JMS messages.
* Simulator
  + It simulates a front end system that can generate some sell/buy requests.
  + It generates a number of sell/buy bond requests every minute.
* JMS Broker
  + Act as an intermediator between front end systems and back end systems.
  + Front-end systems do not allow talking with Back-end service directly. They can communicate by passing message via any messaging infrastructure. Currently, we are using Tibco EMS as the messaging infrastructure.
  + We created some topics and queues to serve different functions and clients.
  + The format of messages is based on JSON that we have listed in the below table.
* Back-end Bond Trading Service
  + It is playing an important role in this experimental application. It is similar to Shanghai Stock Exchange Corporation. It needs to handle all trade requests from front-end trading system, including buy/sell/bid/offer and compute price of each bond.

# Use Cases

1. Simulator publishes a message to sale bond in every 1 minute (The parameter is configurable).

We have listed some bond templates in below section. You can pick up a bond out of them randomly and broadcast the sale message to everyone via JMS broker.

(Tips: you can change some values of bond, so it looks like a true bond much)

After traders received the sale message, they can make a decision to buy or do nothing for this bond.

Once the bid messages that sent by traders reached to simulator, the simulator need to store the bid messages in memory till the bidding time is expired (for example, 20 seconds), and then pick up the highest price to deal, and the bidding price should be greater than Present Value (you need to compute the PV against the standard valuation format).

If no traders send the buy requests for this bond, this bond will be closed after 5 minutes (The parameter is configurable).

1. Simulator publishes message to buy a bond in every 1 minute (configurable).

Similar to #1, you can refer to the detailed message format in the next section.

1. In front-end side, you should show the bond requests that are being sale, buy and processing. (To see figure 3)
2. Trade statistic report for this trader. (figure 4)
3. To view trading history for this trader. (figure 5)

# Sequence Diagram

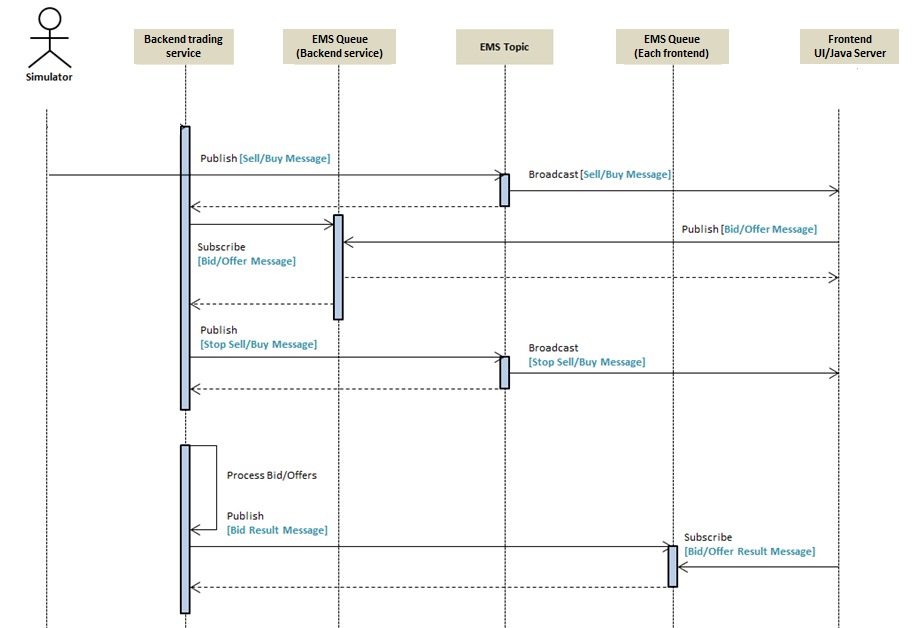


Figure 2: Trading work flow

The high level process is as below:

1. Back-end trading service (Simulator): Create a random bond message in every minute and then publish the new bond message to EMS topic. So Front-end systems (clients) can subscriber the messages and decide to buy or sell the bonds.
2. Bid: If there are multiple clients sent the buy messages, so simulator need to evaluate the price and choose a client who offers the maximum price to deal.
3. Back-end trading service (Simulator): Send trading result to every buyer/seller.

# JMS Message Format Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message Name | Destination | Publisher | Subscriber | Demo |
| Sell Message | [Topic]  EBOND.BOND.TRADING | Simulator | Client | {"action":"SELL","quantity":100,  "name":"APPLE 20 years bond", "cusip":"99006D9R0","fv":100,  "couponRate":0.07,"ytm":0.1,  "maturity":20,"rate":"AA-","expiredTimestamp":1425455547185,"tradeId":100001} |
| Buy Message | [Topic]  EBOND.BOND.TRADING | Simulator | Client | {"action":"BUY","quantity":100,  "name":"APPLE 20 years bond", "cusip":"99006D9R0","fv":100,  "couponRate":0.07,"ytm":0.1,  "maturity":20,"rate":"AA-","expiredTimestamp":1425455547185,"tradeId":100002,”price”:90} |
| Stop Sell/Buy Message | [Topic]  EBOND.BOND.TRADING | Simulator | Client | {“Result”:”CLOSED”,”TradeID”:”100001”}  {“Result”:”CLOSED”,”TradeID”:”100002”} |
| Bid Message | [Queue] EBOND.BOND.TRADEQUEUE | Client | Simulator | {“Account”:”DARREN”,”Action”:”BUY”,  ”TradeID”:”100001”,”Price”:”90”} |
| Offer Message | [Queue] EBOND.BOND.TRADEQUEUE | Client | Simulator | {“Account”:”DARREN”,”Action”:”SELL”,  ”TradeID”:”100001”} |
| Bid Result Message | [Queue]  EBOND.USER.ACCOUNT | Simulator | Client | {“Result”:”EXECUTED”,”TradeID”:”100001”} |
| Offer Result Message | [Queue]  EBOND.USER.ACCOUNT | Simulator | Client | {“Result”:”REJECTED”,”TradeID”:”100002”} |

Table 1: Trading message format

# Bond Template

Simulator need to generate random bonds every minute, you can refer to below bond samples.

CUSIP: identifier of each bond.

FV: face value.

Coupon Rate:

YTM Rate: Yield to maturity rate.

Maturity: 5 years/10 years/15 years/20 years/30 years.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bond CUSIP** | **Bond Type** | **BondName** | **Issued**  **Date** | **CCY** | **Maturity** | **Quantity** | **FV** | **Coupon Type** | **Coupon Rate** | **Expired Time** | **Credit Rating** | **YTM** |
| 99006C9A9 | Corp Bond | IBM 5 years bond | 20150102 | USD | 5 | 1000 | 100 | Fixed Coupon Rate | 6% | 20 | AAA | 10.00% |
| 99003X9B4 | US Treasury | IBM 10 years bond | 20150103 | USD | 10 | 1000 | 200 | Fixed Coupon Rate | 5% | 30 | AAA | 10.00% |
| 99006B9C7 | Corp Bond | IBM 20 years bond | 20150104 | USD | 20 | 1000 | 300 | Fixed Coupon Rate | 7% | 40 | AAA | 10.00% |
| 99000Q9D8 | US Treasury | IBM 30 years bond | 20150105 | USD | 30 | 1000 | 400 | Fixed Coupon Rate | 8% | 50 | AAA | 10.00% |
| 99004X9Z0 | Corp Bond | MSFT 5 years bond | 20150106 | USD | 5 | 1000 | 500 | Fixed Coupon Rate | 9% | 20 | AA+ | 10.00% |
| 99004T9L0 | Corp Bond | MSFT 10 years bond | 20150107 | USD | 10 | 1000 | 600 | Fixed Coupon Rate | 10% | 30 | AA+ | 10.00% |
| 99005B9C8 | Corp Bond | MSFT 20 years bond | 20150108 | USD | 20 | 1000 | 700 | Fixed Coupon Rate | 5% | 40 | AA+ | 10.00% |
| 99002M9U7 | US Treasury | MSFT 30 years bond | 20150109 | USD | 30 | 1000 | 800 | Fixed Coupon Rate | 6% | 50 | AA+ | 10.00% |
| 99006E9Y3 | Corp Bond | APPLE 5 years bond | 20150110 | USD | 5 | 1000 | 900 | Fixed Coupon Rate | 9% | 20 | AA- | 10.00% |
| 99005F9B1 | US Treasury | APPLE 10 years bond | 20150111 | USD | 10 | 1000 | ### | Fixed Coupon Rate | 8% | 30 | AA- | 10.00% |
| 99006D9R0 | Corp Bond | APPLE 20 years bond | 20150112 | USD | 20 | 1000 | 100 | Fixed Coupon Rate | 7% | 40 | AA- | 10.00% |
| 99006D9M1 | Corp Bond | APPLE 30 years bond | 20150113 | USD | 30 | 1000 | 200 | Fixed Coupon Rate | 10% | 50 | AA- | 10.00% |
| 99006G9T9 | US Treasury | GOOGLE 5 years bond | 20150114 | USD | 5 | 1000 | 300 | Fixed Coupon Rate | 5% | 20 | BBB | 10.00% |
| 99004M9W1 | Corp Bond | GOOGLE 10 years bond | 20150115 | USD | 10 | 1000 | 400 | Fixed Coupon Rate | 6% | 30 | BBB | 10.00% |
| 99004R9Y6 | Corp Bond | GOOGLE 20 years bond | 20150116 | USD | 20 | 1000 | 500 | Fixed Coupon Rate | 7% | 40 | BBB | 10.00% |
| 99006G9B8 | Corp Bond | GOOGLE 30 years bond | 20150117 | USD | 30 | 1000 | 600 | Fixed Coupon Rate | 8% | 50 | BBB | 10.00% |
| 99006E9V9 | Corp Bond | CITI 5 years bond | 20150118 | USD | 5 | 1000 | 700 | Fixed Coupon Rate | 9% | 20 | BB+ | 10.00% |
| 99004U9H6 | US Treasury | CITI 10 years bond | 20150119 | USD | 10 | 1000 | 800 | Fixed Coupon Rate | 6% | 30 | BB+ | 10.00% |
| 99006C9G6 | Corp Bond | CITI 20 years bond | 20150120 | USD | 20 | 1000 | 900 | Fixed Coupon Rate | 5% | 40 | BB+ | 10.00% |
| 99006F9Q7 | Corp Bond | CITI 30 years bond | 20150121 | USD | 30 | 1000 | ### | Fixed Coupon Rate | 10% | 50 | BB+ | 10.00% |

Table 2: Bond template

# Bond Present Value (current price)

1. Suppose all the Bonds are fixed coupon rate.
2. Credit Rating: AAA/AA+/AA-/BBB/BB, from high to low.
3. Bond price valuation: Follow the standard valuation formula.

<http://www.zenwealth.com/businessfinanceonline/BV/BondPrice.html>

# JMS Topic & Queue

|  |  |  |
| --- | --- | --- |
| **Destination Description** | **Type** | **Destination Name** |
| Simulator broadcast sell/buy requests | Topic |  |
| Backend trading service send trading result to front end systems  (Backend service -> Frontend Java Servers) | Queue | (Every front end system has a dedicated queue, the name is just like below format)  EBOND.ZhangSan  EBOND.LiSi  EBOND.WangWu  … |
| Front end systems send buy/sell request to back end trading service  (Frontend Java Server -> Backend Service) | Queue |  |

Table 3: Topic, Queue

# UI Mockup

Show all sell/buy bond requests from simulator and other clients. Be able to send sell/buy requests for selected bonds.

Show the processing trading status that you sent in recent 5 minutes.

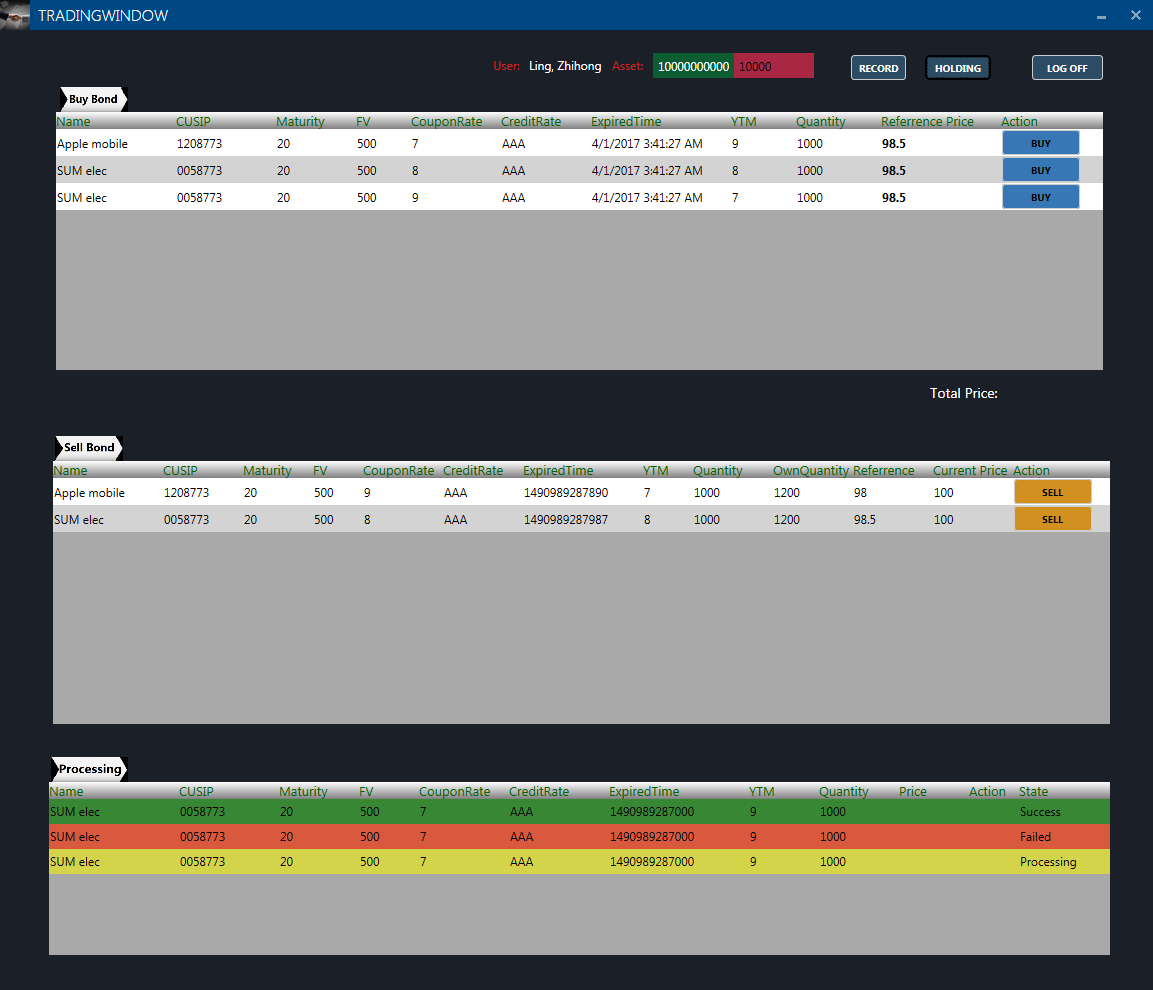


Figure 3: Show the bonds that you held (position information).

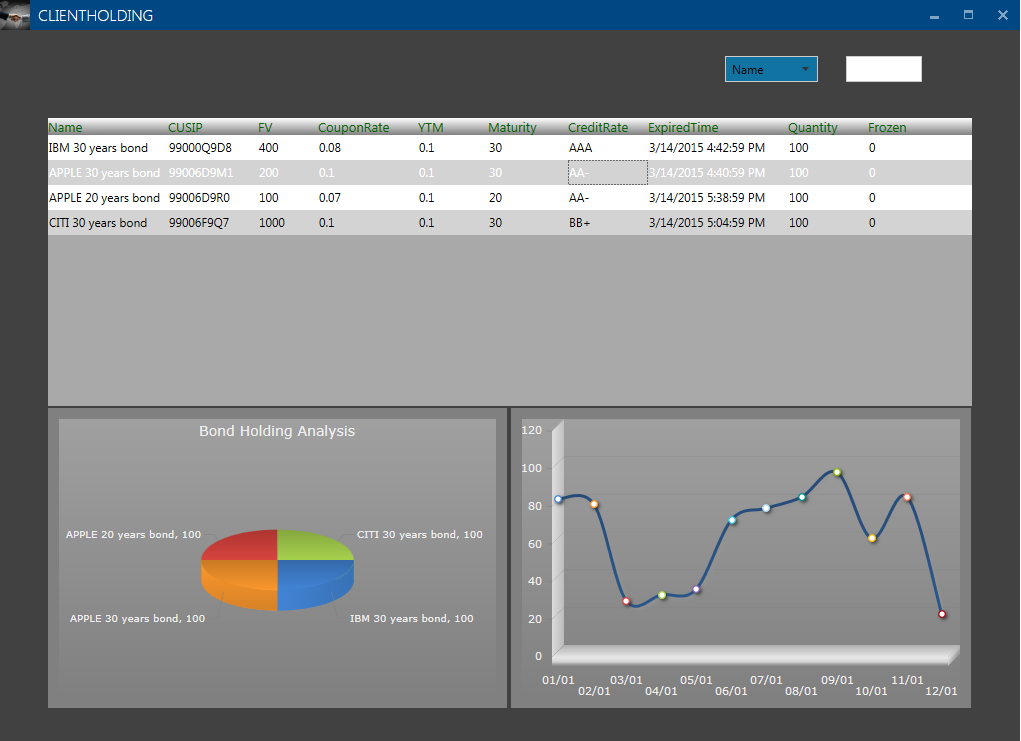


Figure 4: Trading statistics

Show trading history.

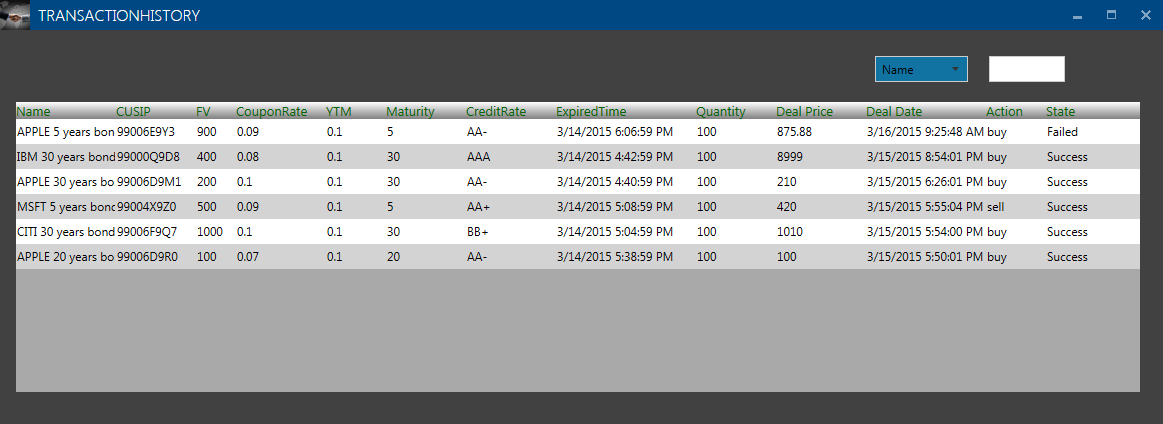


Figure 5: Trading history

# <End>