Instruction

**Code Pairing Assignment**

**Write a Java program with all the JUNIT cases. TDD approach will be preferred. Time to complete the below assignment is 2 to 3 hrs.**

**Problem Statement**

There is a scenario where thousands of trades are flowing into one store, assume any way of transmission of trades. We need to create a one trade store, which stores the trade in the following order

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Trade Id** | **Version** | **Counter-Party Id** | **Book-Id** | **Maturity Date** | **Created Date** | **Expired** |
| **T1** | 1 | CP-1 | B1 | 20/05/2020 | <today date> | N |
| **T2** | 2 | CP-2 | B1 | 20/05/2021 | <today date> | N |
| **T2** | 1 | CP-1 | B1 | 20/05/2021 | 14/03/2015 | N |
| **T3** | 3 | CP-3 | B2 | 20/05/2014 | <today date> | Y |

There are couples of validation, we need to provide in the above assignment

1. During transmission if the lower version is being received by the store it will reject the trade and throw an exception. If the version is same it will override the existing record.
2. Store should not allow the trade which has less maturity date then today date.
3. Store should automatically update expire flag if in a store the trade crosses the maturity date.

1. Project Overview

Provide support to process thousands of trade request. That can be processed using some rules.

In case of exception the user should know what is the error or cause for the failure.

2. Technical Design

Create an application that can process the thousands of the requests at a time. Provide details to the user in case of error. To achieve this, we can create rest api. We can create microservices. We can run multiple instances of the microservice to process the load of request better. Basis the load we can scale-up and scale-down the microservice instance, by this approach we will be able achieve better performance.

To mark the expiry true for past maturity date records a Cron job is created. This job will run everyday at 6AM to mark all the old trades (traded with maturity date less that current date) as expired

Code also covers the test cases

3. Assumptions

The user will send request one by one

The user expects the response for each request that has been sent – If we are not supposed to send the response back, we can go for Spring Cloud Bus where create task and publish the user request to kafka topic or rabbitmq and then corresponding task will process the request. This can be an alternate way to process the large volume of requests.

3. Alternate Approach

If we are getting this trade details in any csv or text file, with some delimiter and we are not supposed to send response for each record to the user,

we can also use Spring-Batch. In that case we can to store invalid data in separate file so that end user can know the faulty records.

\*\*\*Steps to Run the service

1. Copy the code

2. Run mvn clean install - In case you do not want to run the test cases then run mvn clean install -DskipTests

3. Go to target folder

4. Run tradeProcessor-0.0.1-SNAPSHOT.jar using command java -jar tradeProcessor-0.0.1-SNAPSHOT.jar

5. Swagger UI to explore the apis - http://localhost:8080/trades/swagger-ui.html

You can change the port by passing -Dserver.port=8083 in above command. In that case you have to use the same port in below swagger url.

6. api to open swagger -

http://localhost:8080/trades/swagger-ui.html

Sample json to insert the value

{

"bookId": "b1",

"counterPartyId": "cp-1",

"createdDate": "2021-05-23",

"expired": "N",

"maturityDate": "2021-05-25",

"tradeId": "T1",

"version": 3

}

\*\*\*Steps to Run the service

1. Copy the code

2. Run mvn clean install - In case you do not want to run the test cases then run mvn clean install -DskipTests

3. Go to target folder

4. Run tradeProcessor-0.0.1-SNAPSHOT.jar using command java -jar tradeProcessor-0.0.1-SNAPSHOT.jar

5. Swagger UI to explore the apis - http://localhost:8080/trades/swagger-ui.html

You can change the port by passing -Dserver.port=8083 in above command. In that case you have to use the same port in below swagger url.

6. api to open swagger -

http://localhost:8080/trades/swagger-ui.html

Sample json to insert the value

{

"bookId": "b1",

"counterPartyId": "cp-1",

"createdDate": "2021-05-23",

"expired": "N",

"maturityDate": "2021-05-25",

"tradeId": "T1",

"version": 3

}