Coding Challenge

Processing Customer Transaction

Objective:

Process loan transactions automatically at the specific time.

Conception:

Since it is an automated service, this can be achieved by scheduler services such as Flux, Quartz or Spring in-memory scheduler. To implement scheduler like Quartz, we need a database which makes more dependency for this prototype. To achieve portability and friendly deployment, this service has been implemented using spring scheduler. We used in-memory storage to process the transaction instead of database.

Technologies:

This task has been implemented in Java 1.8, maven and Spring Boot which includes Tomcat server.

Architecture:

This application is developed using Spring Boot. Please look at the class diagram.

Class Description:

\src\main\java\com\microservice\greaterbank\GreaterbankTaskApplication.java

> This class is the Spring Boot class which tells Spring framework to create context and inject @Component class in the class path.

src\main\java\com\microservice\greaterbank\job\CustomerLoanTransactionJob.java

> This is a job class which is annotated with @EnableScheduled and @Scheduled. This is responsible for calling the required service at specific cron expression mentioned on @Scheduled annotation.

\src\main\java\com\microservice\greaterbank\service\CustomerTransactionService.java

> This is a service class which handles all the business logics. This is annotated with @Service which will be wired by Spring IOC. This is responsible for posting the transaction, writing report and moving completed files.

\src\main\java\com\microservice\greaterbank\common\CommonHandler.java

> This is helper class used to process the external file and some common tasks. All the methods in this class are static.

\src\main\java\com\microservice\greaterbank\data\BatchTransactionData.java

This is a DTO class which holds list of Transaction Journal entries.

\src\main\java\com\microservice\greaterbank\data\TransactionJournalData.java

This is a DTO class which holds individual transaction journal data which is added with Batch Transaction Data

\src\main\java\com\microservice\greaterbank\data\PostingData.java

This is a DTO class which holds the accumulated credit and debit transaction amount.

Future Enhancement:

- We can integrate job schedulers like Flux or Quartz to schedule jobs.
- We can use database to keep each accounts transactions.
- Also we can add provisions to upload the file into the core system via UI and store the transactions into the database and the end user or financial department can verify it and initiate the transactions for accounting entries from the UI.
- Alternatively we can write Java stored procedure and run this application inside oracle server. Since it runs inside database, which will give better performance. We can trigger oracle objects from scheduler.

Pre-Requests:

- This application runs on the port number 8080. So please ensure it is not used already.
- Create a directory on the system where you wanted to run this application and create three sub-folders called pending, processed and reports. Keep your transaction file on pending folder. The program automatically moves processed files into processed folder and reports are generated on reports folder.
- Create environment variable for TRANSACTION_PROCESSING and set the root directory as value.
- The trigger time pre-set to run twice a day which is 06.02 and 21:02. Make sure the application is started and transaction file placed before those times. Once you started the server, the application will be running until close the command prompt.

Deployment:

This project is bundled into greaterbank-task-0.0.1-SNAPSHOT.jar. This can be executed on the machine which has Java 8. Run the below command on the command prompt which will start the server.

```
java –jar greaterbank-task-0.0.1-SNAPSHOT.jar
```