Bank Application

Author : Chetan Pawar

Entity :

Data will be persisted in 3 different tables.

1. Account :

One row for each account reference

Methods :

I. performTransaction(BigDecimal transactionAmount) : - This method performs a transaction on an account.. It first checks if the transaction will lead to negative balance. Successfull transactions will be carried out.

II. *getBalance() :* Returns the balance in the account.

2. AccountTransaction :

One row for each transaction reference

Methods :

I. toTransactioValueObject () : This method converts accountTransaction object on which it is called to the value of Transaction.

3. AccountTransactionLeg :

One row for each transaction_leg reference

Package Structure :

1. launcher : contains spring boot application startup class "BankApplication.java".
2. exception : contains newly added exceptions
3. model : contains all entities that we want to persist.
4. repository : contains all jpa repositories.
5. service : contains implementation of all service classes
6. util : contains utility class
7. validator : contains Account and Transaction validator classes.

Technology Stack :

1. Java 8
2. Spring Boot 1.5.9
3. Spring Data JPA :
4. Database : H2
5. Junit
6. SpringBootTest
7. Apache Common Langs
7. Maven 3.x

Technology Choices:-

1. Spring Boot :

Spring boot has been used for dependency injection. The code is much cleaner using spring and all the boiler plate code is managed internally by spring including transaction management. In addition to this Spring boot allows us to configure application by using application.properties file and ready to use annotations.

2. Spring Data JPA :

In assignment spring data JPA has been used with spring boot which makes it easy to easily implement JPA based repositories.It deals with enhanced support for JPA based data access layer. Spring data JPA significantly improve the implementation of data access layer by reducing the effort to the amount thats actually needed.

3. H2 In memory Database :

Datastore to persist account and transaction details.

4. Junit :

Junit is sued for writing tests.

5. Apache Commons Langs :

open source library for Utility methods

6. Multi-threading and transaction management :

When we are performing a multi-legged transaction on 2 or more accounts, transaction update will be carried out with Isolation level **REPEATABLE_READ.** It guarantees that any data read was committed at the moment it is read. It also guarantees that any data that was can not change. If the transaction reads the same again, it will find previously read data in place, unchanged and available to read. It will rollback the transaction if any exception is thrown in this block.

7. Maven :-

used for the purpose of project build, dependency management.

8. Error Handling :

In addition to this given exceptions we have added one extra Exception "AccountAlreadyExistException" which extends Business exception.

9. Changes done in pom.xml :

I have created TestSuite to represent/run all the tests. Please find TestSuite.java file to find all tests.

```
<plugins>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.16</version>
        <dependencies>
            <dependency>
                <groupId>org.apache.maven.surefire</groupId>
                <artifactId>surefire-junit47</artifactId>
                <version>2.16</version>
            </dependency>
        </dependencies>
        <configuration>
            <includes>
                <include>**/TestSuite.java</include>
            </includes>
        </configuration>
    </plugin>
</plugins>
```

I have added `<include>**/TestSuite.java</include>` to run TestSuite as a part of maven-surefire-plugin.
Command to run TestSuite :

```
> mvn clean test
```

Design Decisions :

1. Why we created Transaction and TransactionLeg entities again?

We have created Three different Entity classes to persist account and transaction related data. Even though we have already given Transaction and TransactionLeg entities but I wanted to persist it using spring and spring data jpa using annotations so that it will be more clear , so I created two extra entities apart from Account entity.

2. Why we created Test Suite ?

I wanted to run BankFunctionTest.java test but as We have used spring boot and jpa , passing the application context to BankFunctionalTest was little bit tricky so to make it clear and concise We created TestSuite and by using TestSuite we are running all tests from a single point.

Test Coverage :

- Approximate 90%