Service Specification

Transaction Management

Client Controller
Transaction Controller

Author : Madhusudan Byregowda

Date : 30-9-2018

DOCUMENT REFERENCE INFORMATION

Name of the Document	Transaction Management
File name	TransactionManagement_01.00.00.docx

KEY PLAYERS

Key player	Name
Solution Architect / Development	Madhusudan

VERSION HISTORY

Nr	Description (of update)	Date	Version
1	Initial creation	28-09-2018	0.1
2	Final version	30-09-2018	1.0

REFERENCES

Nr.	Links
1	https://maven.apache.org
2	https://spring.io/projects/spring-boot
3	https://projects.spring.io/spring-data-jpa/
4	https://spring.io/guides/gs/rest-service/
5	https://www.apache.org/
6	https://www.oracle.com/technetwork/java/javase/overview/java8- 2100321.html
7	http://www.springboottutorial.com/spring-boot-and-h2-in-memory- database

CONTENTS

Contents 3
1 Overview of Transaction Management 4
1.1 CLIENT CONTROLLER 4
1.2 Transaction Controller 4
2 How to Get the App Up 5
2.1 Pre-requisites to run the App 5
2.2 Getting Started 5
3 Connect & reconnect 6
3.1 CLIENT CONTROLLER Example data 6
3.2 Transaction CONTROLLER Example data 8
4 Logical Data Model 9
5 Technology Stack 10
5.1 Java 8 10
5.2 Spring / Spring Boot 10
5.3 Maven 10
5.4 H2 in memory database 10
5.5 JPS 11
5.6 RESTfull Web Service 11
5.7 Apache embedded tomcat server 11

1 OVERVIEW OF TRANSACTION MANAGEMENT

The purpose of this document is to describe the application and its operation(s).

The application Transaction Management enables the user to manage client creation and provides REST API to transact between two accounts.

Transaction Management Application provides below REST full Controllers:

1. Client Controller
2. Transaction Controller

The following sections list the operations of a service and a short description of what the operation offers to the user. More details can be found in following sections.

1.1 CLIENT CONTROLLER

Client controller operations enables the user to create new clients and accounts for clients

Protocol : REST/HTTP

Interaction pattern : Request-Response

1.2 TRANSACTION CONTROLLER

Transaction controller operations provides the user API to execute transaction between two accounts

Protocol : SOAP/HTTP

Interaction pattern : Request-Response

2 How to Get the App Up

2.1 Pre-requisites to run the App

maven 3 or higher

java 8 or higher

2.2 Getting Started

Below steps needs to be followed in order to run the application.

1.Clone the source code from git in local directory

$ git clone https://github.com/mbyregow/TransactApp.git

2.Move to the root directory of the TSApp project into any local folder and build the source code using maven command

$ mvn clean install

3.To start the application below command can be used

$ java -jar target/TS-0.0.1-SNAPSHOT.jar

3 CONNECT & RECONNECT

3.1 CLIENT CONTROLLER EXAMPLE DATA

URI : /clients/{clientId}
Description : To get a specific client
HTTP Method : GET
Request : NA
Response : Status : 200


```
{
        "id": 1,
        "firstName": "Test", "lastName": "Test",
        "emailId": "test.test@mail.com", "accounts": [
                {
                        "accountId": 1,
                        "accountNumber": "100000",
                        "accountType": "SAVING", "balance": 1000

                }
        ],
}
```


URI : /clients
Description : To get all the clients
HTTP Method : GET
Request : NA
Response : Status : 200


```
[{
        "id": 1,
        "firstName": "test",
        "lastName": "test",
        "emailId": "test.test@mail.com", "accounts": [
                {
                        "accountId": 1,
                        "accountNumber": "100000",
                        "accountType": "SAVING",
                        "balance": 1000
                }
        ],
},
{
        "id": 2,
```

```
	"firstName": "test2",
	"lastName": "test2",
	"emailId":"test2.test2@mail.com", "accounts": [
		{
			"accountId": 2,
			"accountNumber": "100000",
			"accountType": "SAVING",
			"balance": 2000
		}
	],
}]
```


URI : /clients
Description : Create a new client and account
HTTP Method : POST
Request :


```
{
	"firstName":"test",
	"lastName":"test",
	"emailId":"test.test@mail.com", "accounts" :[
		{
			"accountNumber" : "1000000",
			"accountType" : "SAVING",
			"balance" : "2000"
		}
	]
}
```


Response : Status : 200


```
{
	"id": 1,
	"firstName":"test",
	"lastName":"test", "emailId":"test.test@mail.com", "accounts" :[
		{
			"accountNumber" : "1000000",
			"accountType" : "SAVING",
			"balance" : "2000"
		}
	]
}
```

3.2 TRANSACTION CONTROLLER EXAMPLE DATA

URI : /transact

Description : transact between two accounts

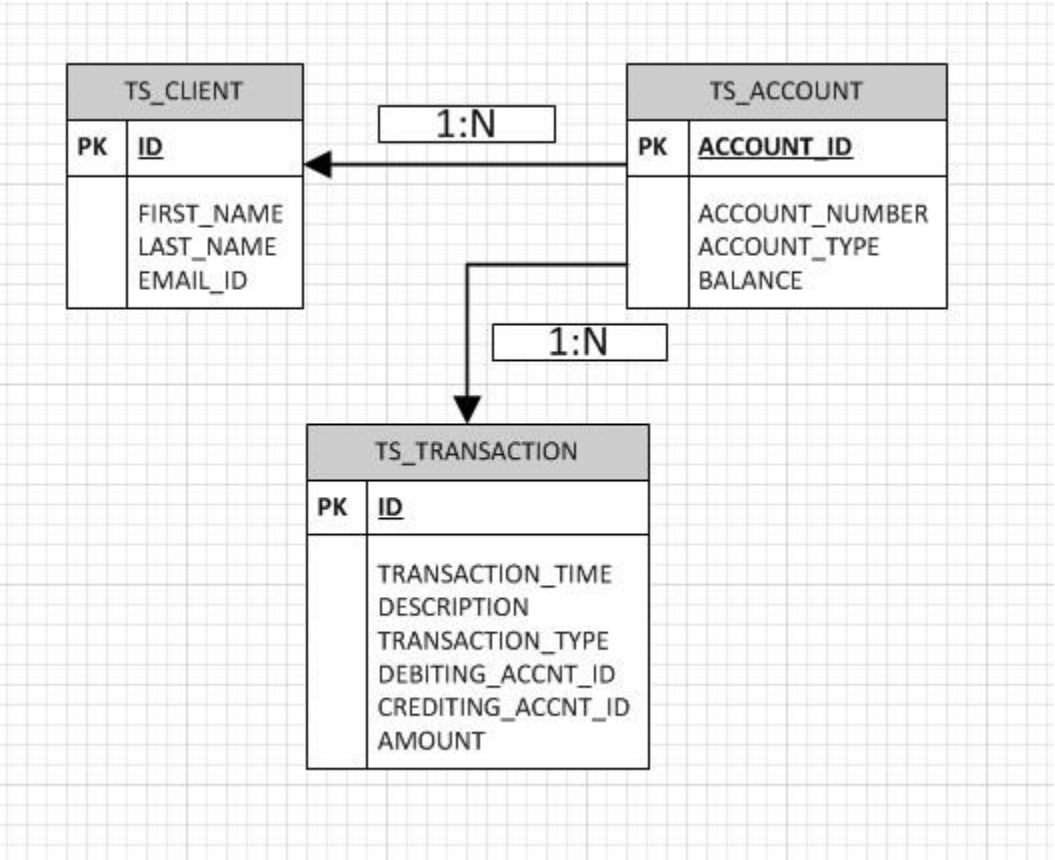
HTTP Method : POST

Request :

```
{
        " description" : "test",
        " debitingAccount" :{
                "accountNumber" : "1"
        },
        " creditingAccount" :{
                "accountNumber" : "2"
        },
        "amount" : "100"
}
```

Response : Status : 200

4 LOGICAL DATA MODEL



5 TECHNOLOGY STACK

Major technology stack used for building Transfer management services have been outlined below along with the advantages and limitations :

1. Java 8
2. Spring REST Controller, Spring Boot
3. Maven
4. H2 in memory database
5. Spring -JPA
6. RESTful Web Service
7. Apache embedded tomcat server

5.1 JAVA 8

Java 8 is a revolutionary release of the java development platform. It includes a huge upgrade to the Java programming model and a coordinated evolution of the JVM, Java language, and libraries. Java 8 includes features for productivity, ease of use, improved polyglot programming, security and improved performance.

Advantages :

- Improved performance.
- Better memory management.
- Improved productivity.

5.2 SPRING / SPRING BOOT

Spring boot is used to reduce the initial setup work needed and spring boot provides an easier way to initialize data source and create a executable jar. Dependency management is much easier with spring boot in place.

Advantages :

- Spring Boot makes it easy to create spring-powered, production ready application and services.
- Enables radically faster and widely accessible experience for all Spring development
- Provide a range of non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration)
- Absolutely no code generation and no requirement for XML configuration

Limitations :

- Spring Boot is limited to work with Spring based applications (Java/groovy).

5.3 MAVEN

Maven is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software: first, it describes how software is built, and second, it describes its dependencies.

5.4 H2 IN MEMORY DATABASE

Typical databases involve a lots of setup like install the database, setup schema, setup the tables, application to database connection. Therefore for the TMS in memory H2 database has been used which is provided by spring boot.

Advantages

- Zero setup or infrastructure
- No extensive Configuration or maintenance
- Easy to use for Learning, POCs and Unit Tests
- Spring Boot provides Simple Configuration to switch between a real database and an in memory database like H2

Limitations

- Poor performance.
- Concurrency issues.

5.5 JPS

To implement the data access layer the JPS spring -jpa has been utilized in the application. It helps in reducing boiler-plate code and reduces the effort and amount needed to maintain data access layer.

Advantages

- Support to build repositories based on Spring and JPA
- Easy migration to other JPAs for data access layer.
- Input validation support for entity and objects.

5.6 RESTFULL WEB SERVICE

REST or RESTful API design (Representational State Transfer) is designed to take advantage of existing protocols. While REST can be used over nearly any protocol, it usually takes advantage of HTTP when used for Web APIs.

Advantages

- Resource identification using URI.
- Uniform interface.
- Self-descriptive messages
- Stateless interactions through hyperlinks

Limitations

- Only works on top of the HTTP protocol.
- Hard to enforce authorization and security on top of it.
- Hard to enforce a strict contract between client and server.

5.7 APACHE EMBEDDED TOMCAT SERVER

Embedded server implies that our deployable unit contains the binaries for the server (example, tomcat.jar).

Advantages

- Incredibly lightweight, Highly flexible and Open-source.
- Offers additional security.

Limitations

- Less robust than paid servers.
- Support for server and Configuration challenges