**Spring Boot Rest Service**

I have built  **RESTFul APIs**for a Simple account opening service. I have used User and Account entity Model

**Table of Contents**

1. What I’have build?
2. Tools and Technologies Used
3. Database
4. Testing REST APIs via Postman Client
5. Test Client
6. Actuator
7. Log File

**1. What we i’ve built and start application**

I have built  **RESTFul APIs**for a Simple account opening service using Spring Boot 2 JPA and H2 embeded database.

I have also used Spring Actuator module and write junit unit test.

There is also log file which name is finance.log

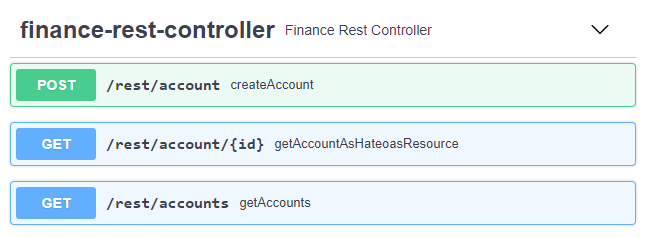
You can download source code and execute in spring-tool-suite or

**There is finance-0.0.1-SNAPSHOT.jar in Project.**

**You can start application by command line. You should write below command.**

**java -jar finance-0.0.1-SNAPSHOT.jar**

**Swagger**



**2. Tools and Technologies Used**

* *Spring Boot* - 2.0.0.RELEASE
* *H2* – Embeded Database
* *JDK*- 1.8 or later
* *JPA -* Hibernate
* *IDE*- Eclipse or Spring Tool Suite (STS)

. **3.Database**

I have used H2 embeded database. While application is starting, scheme.sql and data.sql executes automatically.Those files locates in resorces folder in Project.

Scheme.sql creates tables,keys,and sequences

Data.sql insert new records

I have two tables.

1. T\_user => It stores user data
2. T\_account => it stotes accound data

You can see table details and data which will be inserted in application start at the below.

CREATE TABLE PUBLIC.T\_USER(

ID BIGINT NOT NULL,

FIRST\_NAME VARCHAR(255) not null,

LAST\_NAME VARCHAR(255) not null

);

ALTER TABLE PUBLIC.T\_USER ADD CONSTRAINT PUBLIC.CONSTRAINT\_1 PRIMARY KEY(ID);

CREATE TABLE PUBLIC.T\_ACCOUNT(

ID BIGINT NOT NULL,

NAME VARCHAR(255),

BALANCE DECIMAL(20, 2),

USER\_ID BIGINT

);

ALTER TABLE PUBLIC.T\_ACCOUNT ADD CONSTRAINT PUBLIC.CONSTRAINT\_2 PRIMARY KEY(ID);

CREATE SEQUENCE PUBLIC.FINANCE\_SEQUENCE START WITH 100;

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (1, 'Alexandru', 'Ferit');

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (2, 'Adrian', 'Dal');

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (3, 'Andrei', 'Rize');

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (4, 'Mihai', 'Duru');

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (5, 'Ionuţ', 'Mus');

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (6, 'Ana-Maria', 'Su');

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (7, 'Mihaela', 'Zor');

INSERT INTO t\_user (id,first\_name,last\_name) VALUES (8, 'Andreea', 'Eski');

INSERT INTO t\_account (id,name,balance,user\_id) VALUES (1, 'Family', 0.00, 1);

INSERT INTO t\_account (id,name,balance,user\_id) VALUES (2, 'Retired', 178.34, 3);

INSERT INTO t\_account (id,name,balance,user\_id) VALUES (3, 'MyAccount', 12300.65, 5);

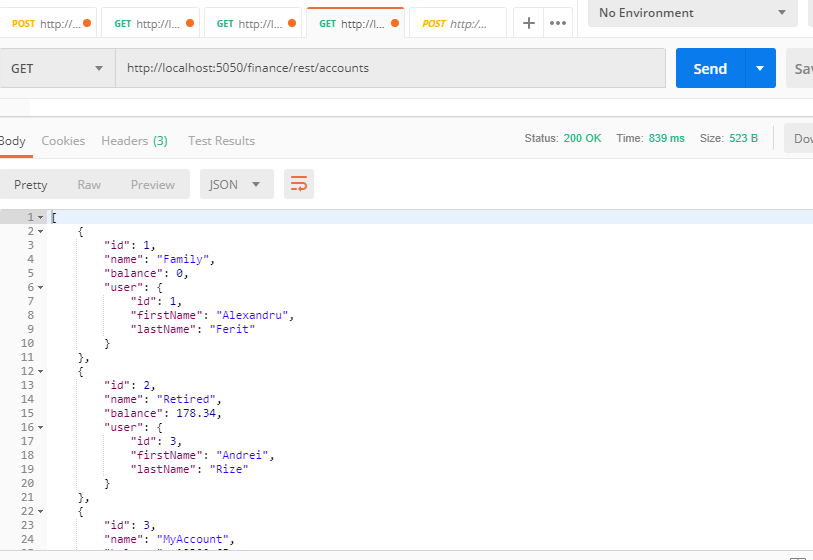
INSERT INTO t\_account (id,name,balance,user\_id) VALUES (4, 'Other', 0.00, 7);

**4. Testing REST APIs via Postman Client**

**Get All Accounts**

**HTTP Method**: GET

**Request URL**: **http://localhost:5050/finance/rest/accounts**

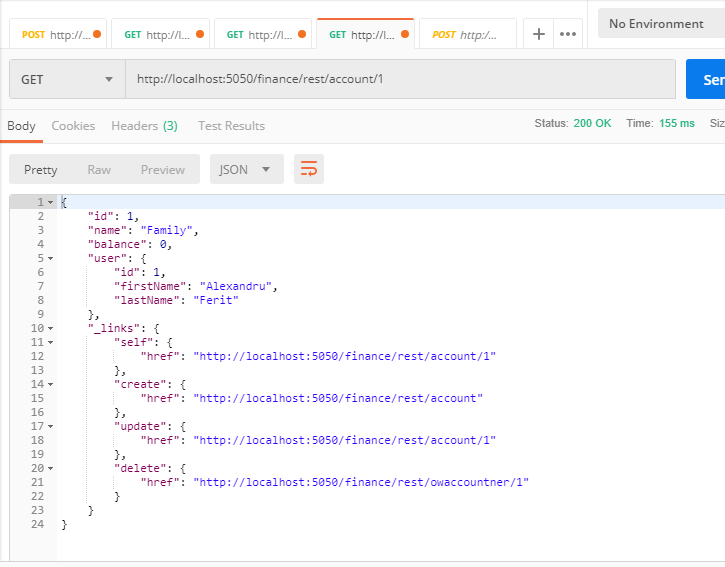


**Get Account by ID REST API**

I have used **HATEOAS**, This architectural style lets you use hypermedia links in the response contents so that the client can dynamically navigate to the appropriate resource by traversing the hypermedia links.

**HTTP Method**: GET

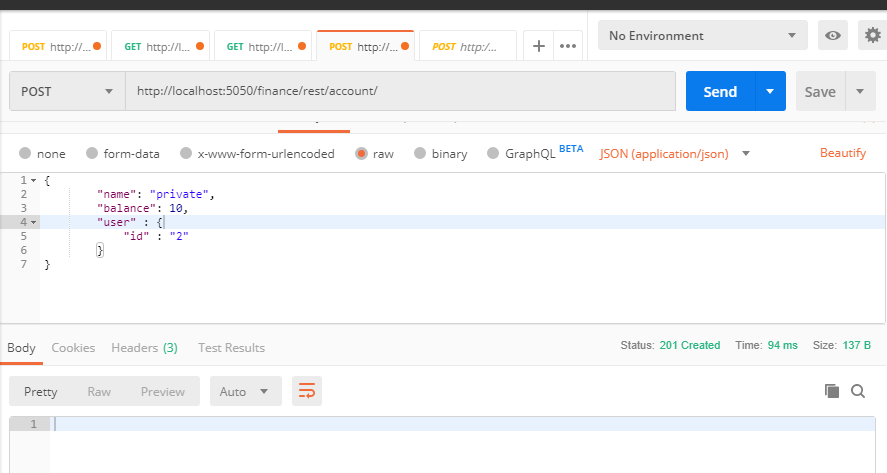
**Request URL**: **http://localhost:5050/finance/rest/account/1**



**Create Account REST API**

**HTTP Method**: POST

**Request URL**: **http://localhost:5050/finance/rest/account/**



If account opening time is not between 08:00 and 17:00 service returns” Accout may open between 08:00 and 17:00” error message.

If userid does not exists in system,service returns “There is not any user with userid error” erorr message

If user has already account,service returns “User has already account” error mesesage

You can test create service as below json. You can modify user:id value for getting error message.

{

"name": "private",

"balance": 10,

"user" : {

"id" : "434"

}

}

**5. Test Client**

I have written a test client for testing account services.

When you first start application, you may execute Junit test client.

You should get success from unit test.

@Test

**public** **void** testGetAccounts() {

ResponseEntity<List> response = restTemplate.getForEntity("http://localhost:5050/finance/rest/accounts", List.**class**);

List<Map<String,String>> body = response.getBody();

MatcherAssert.*assertThat*(response.getStatusCodeValue(), Matchers.*equalTo*(200));

List<String> firstNames = body.stream().map(e->e.get("name")).collect(Collectors.*toList*());

MatcherAssert.*assertThat*(firstNames, Matchers.*containsInAnyOrder*("Family", "Retired", "MyAccount", "Other"));

}

**6. Actuator**

You can see application status from actuator services.

<http://localhost:5050/finance/actuator>

**7. Log File**

You can see application logs at finance.log file.

Log level may changed from application.properties file which locates in Project=>Resources.

