The Transefer Service is implemeneted as REST API. The application is developed using open source artifacts. The brief of the technology stack for the implementation of the solution is as follows

* **Spring Boot (1.5.9.RELEASE) 🡺 main J2EE framewrok**
* **Spring Data JPA (1.5.9.RELEASE) 🡺 ORM abstraction, hinernate is the inbuild persitance provider**
* **h2database (1.4.196) 🡺 for Datastore (In memory database)**
* **LogBack 🡺 For Application logging**
* **Java 1.8 🡺 Compile/Runtime**
* **Maven 🡺 Build tool**
* **Mockito 🡺 Junit framework**
* **Swagger (2.7.0) 🡺 API Documentation**

**Prerquisite to run the application:**

* Maven
* Java 1.8

**Steps to deploy/run the application.**

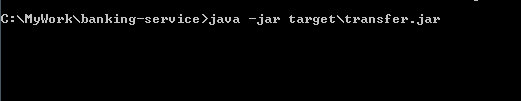
Clone/Download the application from the Github

1. Extratract (if downloaed as zip bundle) and go the the project root directory (e.g. C:\MyWork\banking-service)
2. Run mvn clean install



1. Once build is sucessful, run the executable jar as follwos:

java -jar target\transfer.jar



1. Now the application should start and you should be able to see the logs as follows:



4a. The Application log is generated in the root directory . The name of the file is ***restapi***

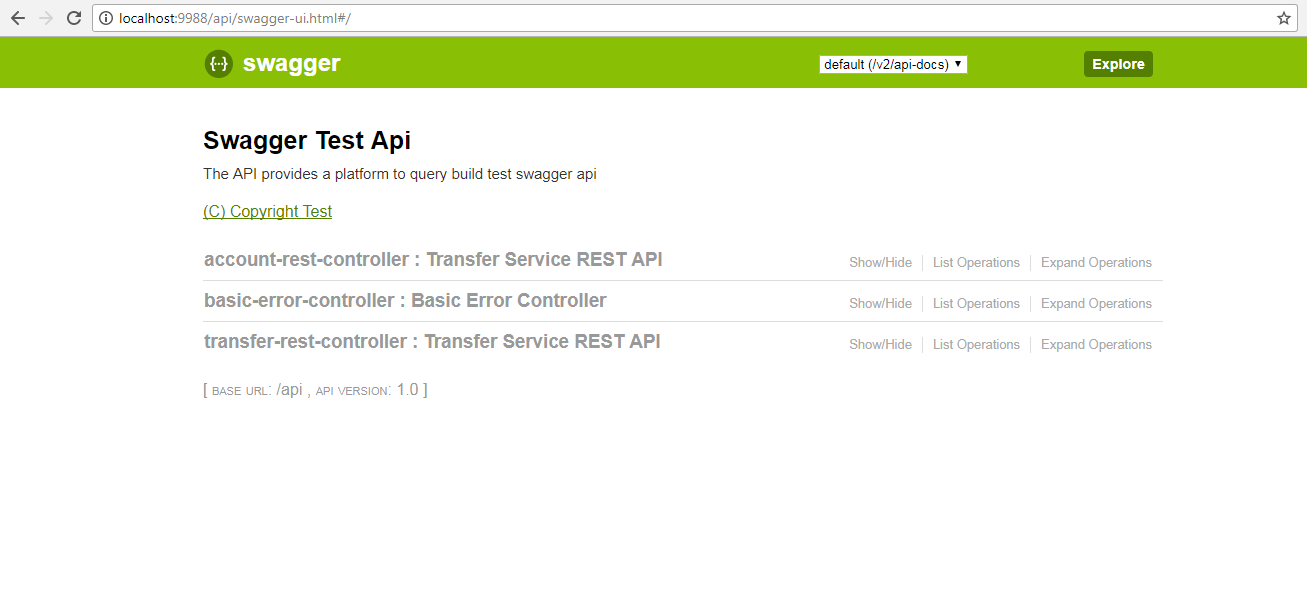
1. The application is running on port 9988 and ContextPath=api

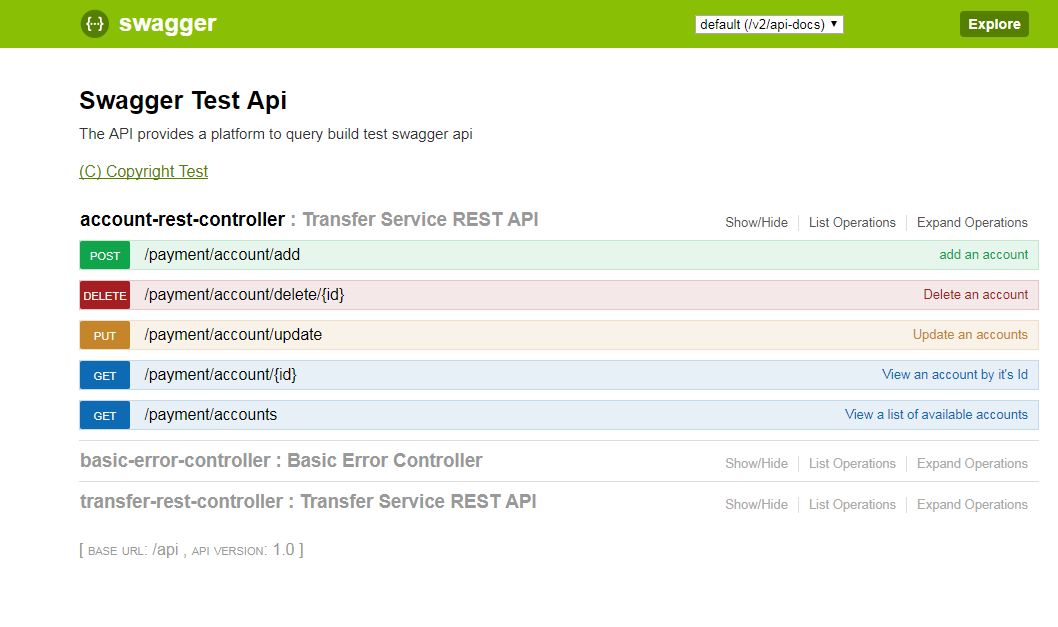
The config can be found in application.properties file under resources directory

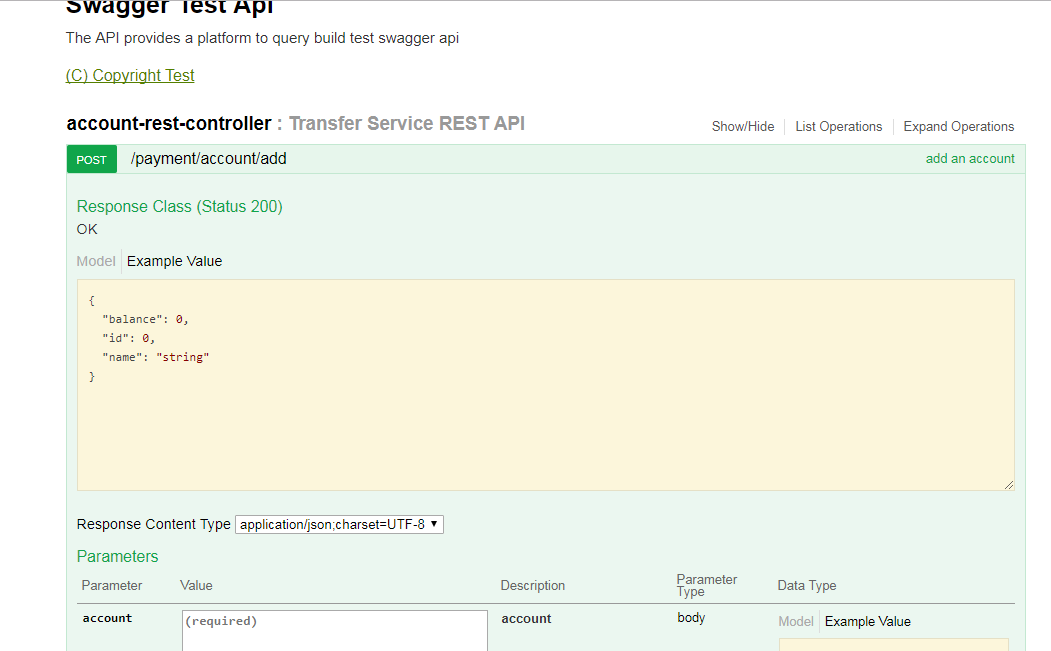
1. The REST API documentation can be found in the following link

<http://localhost:9988/api/swagger-ui.html>

Following are fe sample screens:







1. Now, You go ahead and add 2 accounts (for example) using following link and sample request

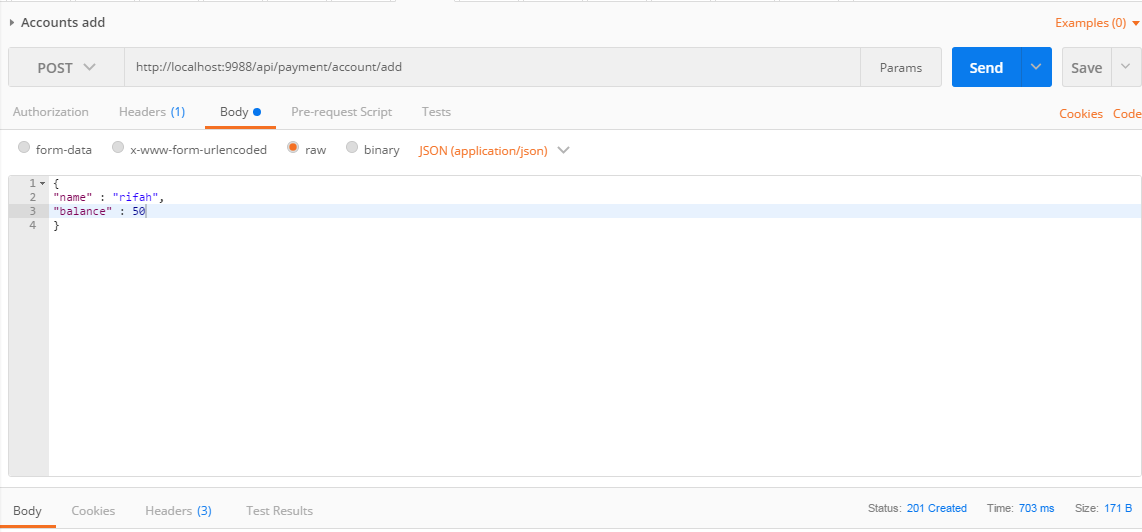
<http://localhost:9988/api/payment/account/add>

**{**

**"name" : "rifah",**

**"balance" : 50**

**}**

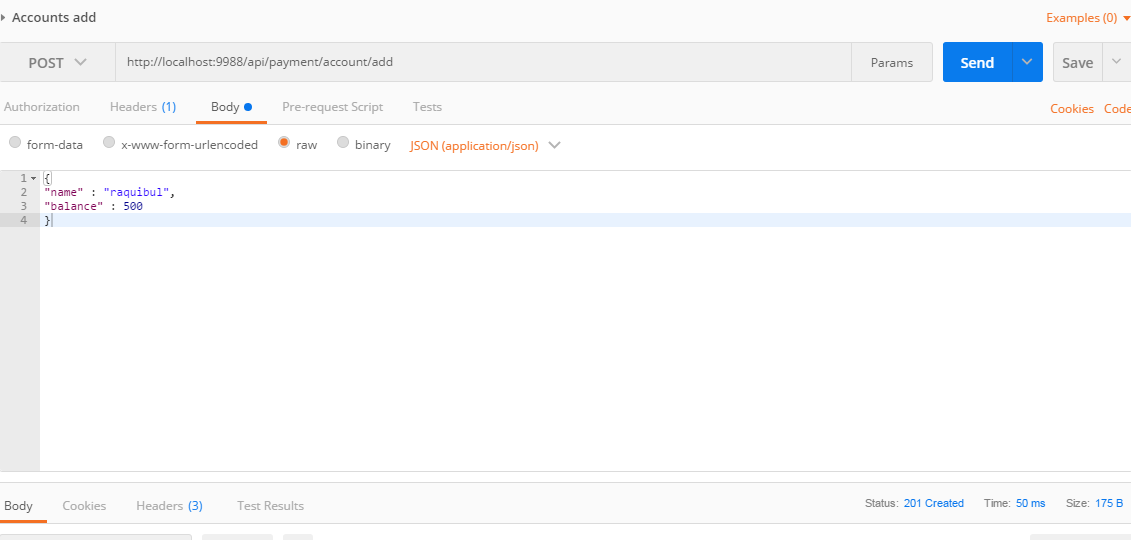


{

"name" : "raquibul",

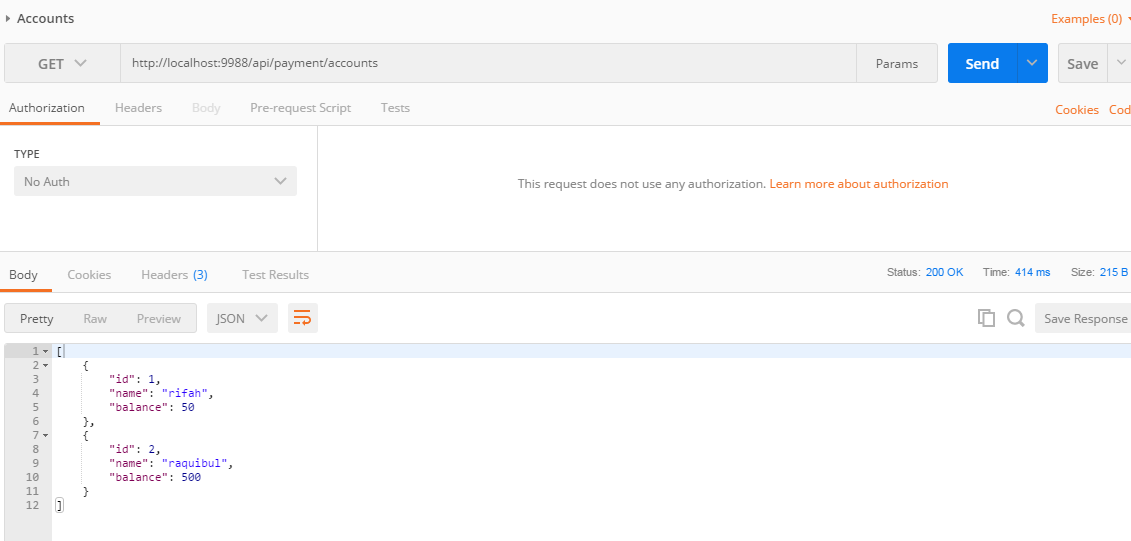
"balance" : 500

}



1. Go Ahead and check that the above added accounts are listed in the following URL

<http://localhost:9988/api/payment/accounts>



1. Now, Transfer/Transact money using following URI

<http://localhost:9988/api/payment/transaction/add>

Here is an example request which transfers 150 EUR from Raquibul to rifah

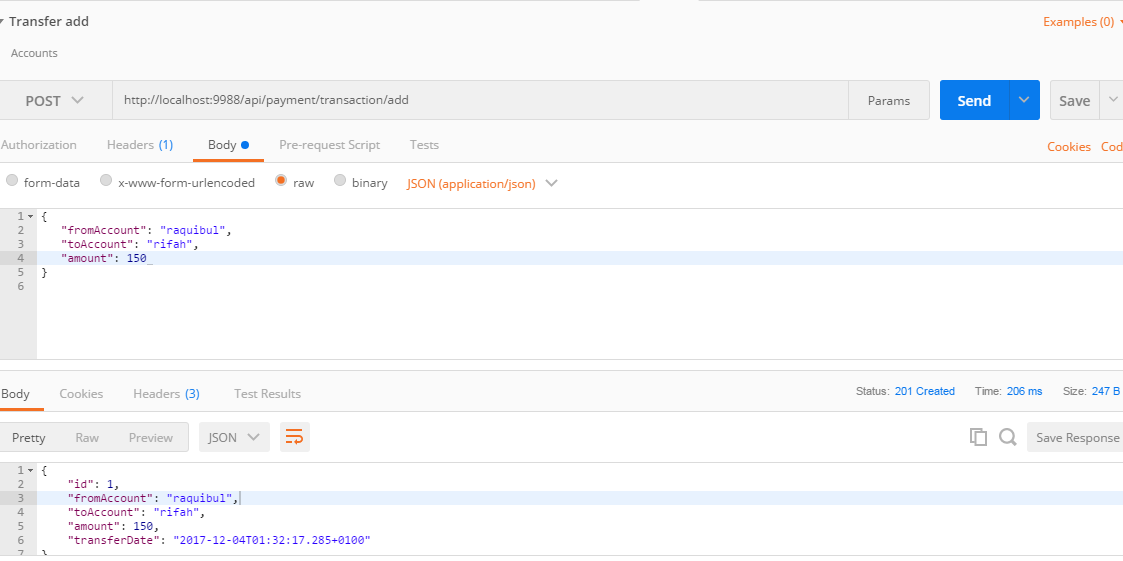
{

"fromAccount": "raquibul",

"toAccount": "rifah",

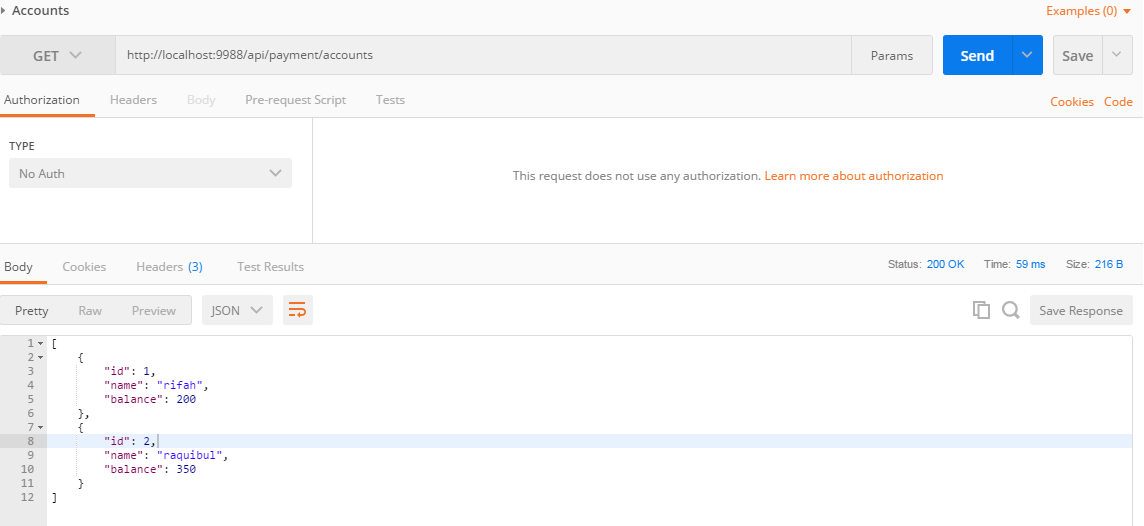
"amount": 150

}



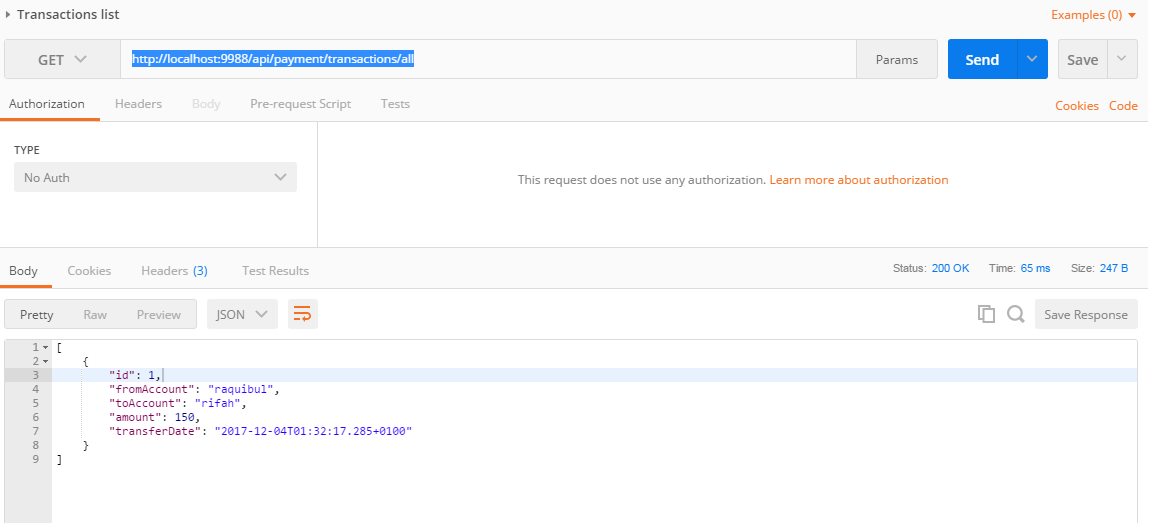
1. After above transfer/transaction check balance of both the account (Raquibul & rifah) . They should be updated with the transfer as follows:

<http://localhost:9988/api/payment/accounts>



You can list all the transactions through following URI

<http://localhost:9988/api/payment/transactions/all>



**Easy Customization options:**

Though the application by default is packaged as jar it can also be pakacged as war. The POM file has all necessary configuration. Only the war profile needs to be enabled. It can be in the following POM snippet:

<profiles>

<profile>

<id>jar</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<packaging.type>jar</packaging.type>

</properties>

</profile>

<profile>

<id>war</id>

<activation>

<activeByDefault>false</activeByDefault>

</activation>

<properties>

<packaging.type>war</packaging.type>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

</dependencies>

</profile>

</profiles>

Once it’s packaged as above war it can be deployed to any web container (Tomcat, JBOSS, Websphere etc.)

The Datasource can easily be updated to integrate with the external Database. Application.properties needs to be updated and necessary driver dependency needs to be added to the POM file.

**Out of Scope/Limitation:**

Security feature is not incorporated as of now. It can also be done using Spring Security module/Spring Tocken

Currency feature is not incorporated as of now. There is no distinction in terms of Currency. All transactions/accounts are using same Currency