The “money-transfer” application is an implementation of the Revolut test task on a position of a Java/Scala developer. The application is written in Java, REST API is implemented with the means of the JAX RS library (now pretty well forgotten because of the Spring).

# **Model**

The application’s model includes classes Account and MoneyTransfer.

## **Account**:

Defines the following properties of an account:

* Id;
* Name;
* Balance;

For example, we can have an account with properties: “id”=” f4782b33-5f2c-4d3f-b8f5-9ef4578e9b13”, “name”=”vasya” and balance = “0”. All fields are non-mandatory; for example, an Account might have an empty “id” value before saving into the datastore.

## **MoneyTransfer**:

Defines the following properties of a money transfer:

* Amount of money;
* “From” account id;
* “To” account id;

All fields are non-mandatory – for such operations as “payroll” and “withdraw” money to or from an account, a MoneyTransfer object can have only the “amount” field filled.

# **Controller and DAO**

AccountRestService is a class that implements a REST API controller. AccountDao is a singleton class that performs operations on accounts. These classes work tightly – AccountRestService get REST requests from the outside world and delegates them to AccountDAO.

The following operations are supported:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation** | **HTTP Method** | **Input** | **Output** | **Exception** |
| Create an account | POST | Account | Account (with assigned id) | - |
| Get all accounts | GET | - | List<Account> | - |
| Delete all accounts | DELETE | - | - | - |
| Get account by id | GET | Account id | Account | AccountNotFound |
| Payroll | POST | Account id, MoneyTransfer object | Account | InsufficientFunds |
| Withdraw | POST | Account id, MoneyTransfer object | Account | AccountNotFound, InsufficientFunds |
| Transfer | POST | MoneyTransfer object | - | AccountNotFound, InsufficientFunds |

# **Build and Deployment**

“money-transfer” is a Maven based application. To build, one must run the “mvn clean package” command – it will clear up, compile, run tests and create an executable jar file in target/ directory.

To run, one must run the following command after the build:

java -jar target/money-transfer-0.0.1-SNAPSHOT.jar

The jetty server will be launched on port 8090; REST API would be available on <http://localhost:8090> endpoint. A [postman](https://github.com/postmanlabs/postman-app-support/wiki) collection is included into the docs/ folder of the project: <https://github.com/edwardz10/money-transfer/blob/master/docs/Money%20Transfer.postman_collection.json>

It includes all REST API operations discussed earlier. One can play with all of operations of accounts and money transfers provided the application is deployed.

# **Tests**

AccountRestServiceTest covers API of the AccountRestService class: creating & deleting accounts, getting account info by and id, payrolling and money transferring between accounts.