This document describes structure and architecture of Currency Aggregator service.

**Overview**

Currency Aggregator service loads and aggregates currency rates from files into one storage and provide interface that allows retrieving, updating and deleting of currency rates. Service supports next file formats: csv, json, xml. Service able to generate report with best propositions of buying and selling currencies. Currency Aggregator provides RESTful API.

Technology stack:

* Java 8;
* Maven 3.5.3;
* Tomcat 9.0.8;
* Spring Framework 5.0.4;
* Jackson 2.9.5;
* OpenCSV 4.1;
* Junit 4.12;
* Mockito 2.18.3;
* Log4J 2.11;
* Moneta 1.2.1.

**High-Level Architecture**

Main components of a Currency Aggregator service are:

* Controller (CurrencyController class)

Controller responsible for accepting http requests from users and responding with requested resource. Controller calls service methods.

* Service (CurrencyInMemoryService class)

Service responsible for providing methods for manipulating currencies. Service delegates method calls both to internal storage and file providers, keeping them in consistency.

* Storage (CurrencyRatesStorage class)

Storage responsible for keeping aggregated currency rates and performing CRUD operations that affects only internal data structures.

* File providers (CsvCurrencyProvider, JsonCurrencyProvider, XmlCurrencyProvider classes)

File providers perform low-level CRUD operations on files.

* Model (CurrencyRate class)

Represents main entity of application – currency that has code, buy and sell price properties.

**API overview**

|  |  |  |  |
| --- | --- | --- | --- |
| **HTTP method** | **URI** | **Query parameters** | **Operation** |
| GET | / | none | Get all rates for all codes and banks |
| GET | /{code} | none | Get all rates for code from all banks |
| GET | /{code}/{operation} | none | Get all prices for operation (buy or sell) from all banks |
| GET | /report | none | Get report with the most profitable prices |
| PUT | /{code}/{operation} | **value, bank** | Update operation price (buy or sell) for bank |
| DELETE | / | **bank** | Delete bank |

For PUT and DELETE requests, service accept query parameters (e.g. localhost:8080/USD/sell?**bank**=mega&**value**=2.2) and response with empty body with either «200 OK» if operation succeed or «422 Unprocessable Entity» if bank in parameter does not exist. If one or both query parameters are not present service responds with «400 Bad Request».

For GET requests, service responses with queried currencies in body with «200 OK» status code and header Content-Type: application/json

For request with HTTP methods that are not present in the table above, service responds with «405 Method Not Allowed» status code.

**UML class diagram**

