**JP Morgan Super Simple Stocks**

The developed application is supposed to manage trade operations on the specified set of stocks. And also calculate several statistics based on stocks and there trade operations.

**1. Basic Requirements**

1. For a given stock:

1.1 Calculate the dividend yield.

1.2 Calculate the P/E Ratio.

1.3 Record a trade, with timestamp, quantity of shares, buy or sell indicator and price.

1.4 Calculate Stock Price based on trades recorded in past 15 minutes.

2. Calculate the GBCE All Share Index using the geometric mean of prices for all stocks

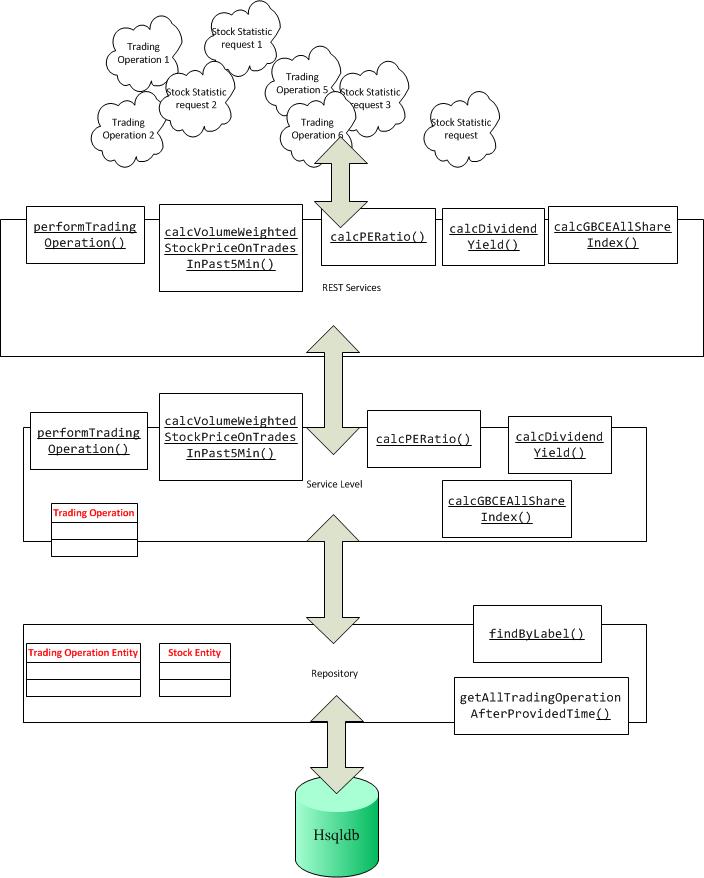
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stock Symbol** | **Type** | **Last Dividend** | **Fixed Dividend** | **Par Value** |
| TEA | Common | 0 |  | 100 |
| POP | Common | 8 |  | 100 |
| ALE | Common | 23 |  | 60 |
| GIN | Preferred | 8 | 2% | 100 |
| JOE | Common | 13 |  | 250 |

**2. Technologies Used in the application.**

1. Maven as a building tool
2. Spring 4 as IoC
3. Spring MVC – to expose Stocks services as REST Services
4. Hsqldb – in memory DB (can be changed to a real DB as Mysql, Oracle, PostgreSQL)
5. Hibernate 5 + JPA Annotations – ORM (maps model to DB tables)
6. Jetty as Web Server (Integrated with Maven)
7. Junit 4 – Testing engine
8. JTA –Support of transaction API
9. Lombok – POJO Helper, source code simplifier
10. Log4J – Logging frmw
11. Spring Data JPA - support of Repository Pattern

**3. System Architecture**

Current application structure is divided into three logical parts: Repository, Service, Rest Services. I will describe them in more details

****

**3.1 Repository**

To support object collecting and saving data in the application I desired not to use simple Collections frwm. To support DB work simulation and activate transactions I desired plug an In-Memory DB (Hsqldb).

This level of application use ORM pattern based on Hibernate frmw with the help of JPA Annotations. Application does not work with DB instances directly, it works only with Entities.

In the Current Application there two Entities with one-to-many relations.

1. StockEntity
2. TradingOperationEntity

Work with DB object is made with the help of Pattern Repository.

**3.2 Service**

This level performs all service operations which were mark in technical requirements. Application retrieves saved information from DB and based on it makes different calculations. This part can be used as an embedded module in another application or can be used by different services with the help of REST or SOAP frmws.

3.3 **Rest Services**

On this level I exposed all the services to WEB.

1. **How to check the Application**

My application is fully covered with Junit-integration tests. To try it, you can do:

1. unpack an \*.jar file with source code.
2. Go to the root of application
3. Build it with command line (maven clean install)
4. Check the results of units tests

Another way it to start up an embedded web server and try REST services manually.

1. unpack an \*.jar file with source code.
2. Go to the root of application
3. Build it with command line (maven clean install)
4. Run with command line (maven jetty:run)
5. When The server successfully started you can check it with links
6. **Check performTradingOperation**

http://localhost:8080/app/stocks/performTradingOperation/GIN/BUY/10110/121.0

Pattern is like this: /performTradingOperation/{stockSymbol}/{tradingOperationType}/{stockQuantity}/{price}

1. **Check calcDividendYield**

http://localhost:8080/app/stocks/calcDividendYield /GIN/100.1

Pattern is like this: /calcDividendYield/{stockSymbol}/{price}

1. **Check calcPERatio**

http://localhost:8080/app/stocks/calcPERatio/GIN/100.1

Pattern is like this: /calcPERatio/{stockSymbol}/{price}

1. **Check calcPERatio calcVolumeWeightedStockPriceOnTradesInPast5Min**

http://localhost:8080/app/stocks/ calcVolumeWeightedStockPriceOnTradesInPast5Min /GIN

Pattern is like this: /calcVolumeWeightedStockPriceOnTradesInPast5Min/{stockSymbol}

1. **Check calcGBCEAllShareIndex**

http://localhost:8080/app/stocks/calcGBCEAllShareIndex

Pattern is like this: /calcGBCEAllShareIndex