**A. Run services:**

java -jar stockPriceService.jar

**B. APIs:**

Codes use Embedded Jersey for server.

Implemented a singleton Map cache (Max 20000 items) to hold data JSON file (GE.json) when starting server.

The Map cache is in structure: [Symbol, ([dateClose1, price1], [dateClose2, price2])] (key: symbol, value: map of date close and price).

**1) Close Price for a ticker symbol for a range of dates (start date and end date)**

GET: <http://localhost:8080/api/v2/GE/closePrice?startDate=2016-06-10&endDate=2016-06-11>

Just simple get values from Map cache with key = symbol.

Each date again is a key of map [DateClose, Price]

It may return close prices of a symbol or one of error messages ("Symbol is not found", "Start Date or End Date is invalid", "Start Date must before or equals to End Date")

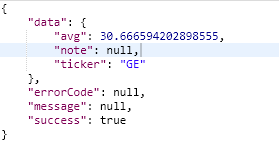


**2) The 200-day moving average price for a ticker symbol beginning with a start date**

GET: <http://localhost:8080/api/v2/GE/200dma?startDate=2016-06-10>

Similar to #1, Get close prices from the cache map and calculate average price.

It may return average prices of a symbol or one of error messages ("Symbol is not found", " Start Date is empty or invalid ")



Not implement: If there is no data for the start date, the first possible start date is suggested in the error message.

**3) Single request for the 200-day moving average price for a up to 1000 ticker symbols beginning with a start date**

POST: <http://localhost:8080/api/v2/200dmas>

BODY:

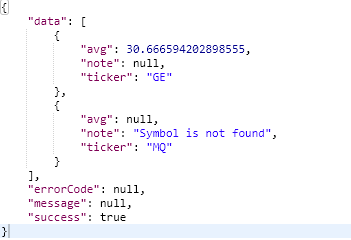
{

"tickers": ["GE","MQ"],

"startDate":"2016-06-10"

}

HTTP has limit on the length of a URI and it’s not recommend for App to bind too long parameters. So, I use POST for this problem instead of GET



Similar to #2, just get average price for each key symbol.

Not implement: If there is no data for a ticker symbol with the start date provided, data for the first possible start date is provided back to the client.

**4) Cache the Close Dates for the 10000 Most Frequently Used ticker symbols so that I will reduce latency on all User Stories**

There are some techniques to cache most frequently used item: Redis, EhCache. For this assignment, I build a simple singleton memory cache with max 20000 items (symbols).

For any request, service will get item from cache, if found then process on the returned data, otherwise, query from database and put the symbol into cache (key: symbol, value: map key - close date, value - price)

Each item in cache has a configured living time. The memory cache itself has a thread (configured timer) to clean up those items which last access from now > living time.