RESTful Notes API Guide

Depanker Sharma

Table of Contents

Overview

These are the API's for importing financial data and outputing the report accordingly.

Chapter 1. General

1.1. HTTP verbs

Verb	Usage
`GET`	Used to retrieve a resource (idempotent)
`POST`	Used to create a new resource
`PUT`	Used to update an existing resource (idempotent)
`DELETE`	Used to delete an existing resource

Chapter 2. HTTP status codes

RESTful notes tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP status codes.

Status code	Usage
`200 OK`	The request completed successfully
`202 Accepted`	The request has been accepted for processing, but the processing has not been completed. The request might or might not be eventually acted upon, and may be disallowed when processing occurs.
`204 No Content`	An update to an existing resource has been applied successfully
`400 Bad Request`	The request was malformed. The response body will include an error providing further information
`404 Not Found`	The requested resource did not exist
`409 Not Found`	Attempt to create a duplicate resource
`500 Internal server error`	Any Unhandled ----

Resources

Chapter 3. Submit-Ticks

This api accepts the tiks and puts them into a quere which is then consumed by a consumer. In case of application shutdown, an attempt is made to save the data in queue onto a file which is checked by the application on startup and loaded if found.

Request

```
$ curl 'http://localhost:8080/submit-tick' -i -X POST \
    -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkZXBhbmtlciJ9' \
    -d 'TIMESTAMP=123123|PRICE=5.24|CLOSE_PRICE=|CURRENCY=EUR|RIC=AAPL.OQ
TIMESTAMP=4564564|PRICE=5.24|CLOSE_PRICE=|CURRENCY=EUR|RIC=IBM.N
TIMESTAMP=6786754|PRICE=|CLOSE_PRICE=7.5|CURRENCY=EUR|RIC=AAPL.OQ'
```

Response

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 56

{
  "message" : "Accepted ticks of size 3, to persist"
}
```

3.3. Error

Malformed request

```
$ curl 'http://localhost:8080/submit-tick' -i -X POST \
    -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkZXBhbmtlciJ9' \
    -d 'DUMMY-DATA'
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: 44

{
  "message" : "Unable to parse the data"
}
```

Request when required fields are missing

```
$ curl 'http://localhost:8080/submit-tick' -i -X POST \
    -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkZXBhbmtlciJ9' \
    -d 'TIMESTAMP=|PRICE=5.24|CLOSE_PRICE=|CURRENCY=|RIC=
TIMESTAMP=4564564|PRICE=5.24|CLOSE_PRICE=|CURRENCY=EUR|RIC=IBM.N
TIMESTAMP=6786754|PRICE=|CLOSE_PRICE=7.5|CURRENCY=EUR|RIC=AAPL.OQ'
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: 287

[ {
  "field" : "acceptTicks.tickers.tickers[0].timestamp",
  "message" : "Timestamp cannot be null"
}, {
  "field" : "acceptTicks.tickers.tickers[0].currency",
  "message" : "must not be blank"
}, {
  "field" : "acceptTicks.tickers.tickers[0].ric",
  "message" : "must not be blank"
} ]
```

Chapter 4. Lookup-Ticks

4.1. About API

4.2. This API expects a SQL query to lookup the data form exported csv. As mentioned in the requirements NO DATABASE is being used.

SQL syntax supports selecting all columns using

`SELECT *` or specific columns which can have following values `TIMESTAMP,PRICE,CLOSE_PRICE,CURRENCY,RIC`.

`FROM` needs to be value or RIC in exported CSV example "AAPL.OQ".

`WHERE` clause only support's AND operator for now.

`ORDER BY` clause is supported with multiple columns.

A Sample query could be

```
select PRICE, CLOSE_PRICE from AAPL.OQ where PRICE > 4 order by timestamp asc, price desc
```

For columns containing numerical values following operators are valid. `'=', '<', '<=', '>', '>=', '!='`

For columns containing string values following operators are valid `'=', '!='`

Request

```
$ curl
'http://localhost:8080/lookup?query=select%20*%20from%20AAPL.OQ%20where%20close_price%
20!%3Dnull%20order%20by%20timestamp%20desc' -i -X GET \
    -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkZXBhbmtlciJ9'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 762

[ {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
}, {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
}, {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
}, {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
}, {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
}, {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
}, {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
}, {
  "TIMESTAMP" : 6786754,
  "CLOSE_PRICE" : 7.5,
  "CURRENCY" : "EUR",
  "RIC" : "AAPL.OQ"
} ]
```

4.2.3. Error

Request when query field is blank

```
$ curl 'http://localhost:8080/lookup?query=' -i -X GET \
    -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkZXBhbmtlciJ9'
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: 70

[ {
  "field" : "getData.query",
  "message" : "must not be blank"
} ]
```

Request csv file does not exist

```
$ curl
'http://localhost:8080/lookup?query=select%20*%20from%20ABCD%20where%20close_price%20!
%3Dnull%20order%20by%20timestamp%20desc' -i -X GET \
    -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkZXBhbmtlciJ9'
```

Response

```
HTTP/1.1 204 No Content
Content-Type: application/json
Content-Length: 42

{
  "message" : "No data found for ABCD"
}
```

Chapter 5. Register user

This is a dummy API of registering the username along with password ====== Request

```
$ curl 'http://localhost:8080/register-user' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "username" : "depanker",
  "password" : "depanker"
}'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 51

{
  "message" : "User depanker has been register"
}
```

5.2. Error

Request when required fields are missing

```
$ curl 'http://localhost:8080/register-user' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "username" : "",
  "password" : ""
}'
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: 128

[ {
  "field" : "username",
  "message" : "must not be blank"
}, {
  "field" : "password",
  "message" : "must not be blank"
} ]
```

Chapter 6. Token fetching

This api will fetch JWT token for the registered user ====== Request

```
$ curl 'http://localhost:8080/token' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "username" : "depanker",
  "password" : "depanker"
}'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 51
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkZXBhbmtlciJ9.L3A9pzvMXgpOgktubpFO6RzHOw-CTDb-
PDna019UFG9sFvwoBn5Pm9zNUza6vIgiCZWHsIZcyspr_jKT0rPnHw
```

6.2. Error

Request when user is not registered or credentials are in-correct

```
$ curl 'http://localhost:8080/register-user' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "username" : "abc",
  "password" : "123"
}'
```

Response

```
HTTP/1.1 403 Bad Request
Content-Type: application/json
```

How to run the software

To run the software, use the terminal, `cd` to `ticker-flow` type `./mvnw clean build`. After the process gets successfully competed a jar `ticker-flow-0.0.1-SNAPSHOT.jar` will get created under target/ directory Now there are two ways to run the jar

1. Execute `java -jar target/ticker-flow-0.0.1-SNAPSHOT.jar`. Once the jar is up, below mentioned curl request's could be used as test cases.

2. Alternatively if docker is installed on the system following command can start the application as Docker container

2.1 Build the docker image `docker build -t depanker/demo-ticker-flow .` or pull the image from dockerhub using `docker pull depanker/demo-ticker-flow`

2.2 Once the above command has been successfully installed, execute the following `docker run -p 8080:8080 -t depanker/demo-ticker-flow`