Assignment

How we will run your program

1. Clone your repository
2. npm install (or pip install, if your language does not have a installer please provide a bash script called install.sh)
3. npm start (another way to run the app)
4. EOF to quit the REPL

Thank you for your continued interest in Rooms to Go!

This at-home coding challenge is an opportunity for you to write some clean code that shows us how you use data structures to solve algorithmic problems.

- Write code as if you were shipping it: Assume you are going to be code reviewed, articulate a test plan, etc.
- We expect this solution to be written in nodejs, but you are welcome to use another language.
- Use your preferred IDE or editor and whatever tooling you're comfortable with.
- Feel free to use whatever references you'd like, including Google.
- Your solution does not have to persist data between runs.
- When you’re finished, make sure your code is committed to the repo with instructions on how to run it.
- Your solution should be self-contained and not require additional software to run it.
- Got an idea for a cool feature to add? Do it! We love seeing your creative side

Problem Statement

Create a command line REPL to manage product inventory.

Managing product inventory requires adding products to a product catalog and adding warehouses to store the products.

We will use 7 commands to enable this functionality:

1. ADD PRODUCT
2. ADD WAREHOUSE
3. STOCK
4. UNSTOCK

5. LIST PRODUCTS
6. LIST WAREHOUSES
7. LIST WAREHOUSE

Details

- Our application will take in user input one line at a time.

Bold text denotes text that will be entered as-is, italics denote arguments that will be replaced by a value. Optional arguments are surrounded by square brackets ([]).

1. **ADD PRODUCT** "*PRODUCT NAME*" *SKU*

- This command adds a new product to our product catalog.
- We can have products with the same names but not the same SKU.
- "PRODUCT NAME" - STRING (Do not store the enclosing quotes they are there for us to be able to pass in space seperated product names.
- SKU - Unique Identifier.

2. **ADD WAREHOUSE** *WAREHOUSE# [STOCK_LIMIT]*

- Creates a new warehouse where we can stock products.
- We assume that our warehouses can store infinitely many products if an optional stock limit argument is not specified.
- WAREHOUSE# - INTEGER
- STOCK_LIMIT - Optional, INTEGER

3. **STOCK** *SKU WAREHOUSE# QTY*

- Stocks QTY amount of product with SKU in WAREHOUSE# warehouse.
- SKU - Unique Identifier, must be a valid sku (is in product catalog).
- Warehouse# - INTEGER, must be a valid warehouse number
- QTY - Integer
- If a store has a stock limit that will be exceeded by this shipment, ship enough product so that the Stock Limit is fulfilled.

4. **UNSTOCK** *SKU WAREHOUSE# QTY*

- Unstocks QTY amount of product with SKU in WAREHOUSE# warehouse.
- SKU - Unique Identifier, must be a valid sku (is in product catalog).
- Warehouse# - INTEGER, must be a valid warehouse number

- QTY - Integer
- If a store has a stock that will go below 0 for this shipment only unstock enough products so stock stays at 0.

5. **LIST PRODUCTS**

- List all produts in the product catalog.

6. **LIST WAREHOUSES**

- List all warehouses.

7. **LIST WAREHOUSE** *WAREHOUSE#**

- List information about the warehouse with the given warehouse# along with a listing of all product stocked in the warehouse.

COMMAND HISTORY

We like to keep a log of commands issued in the product management software so we can debug issues that arose during manual data entry. While the REPL is active, asynchrously stream history to a file in batches of 2. If a user types in two commands we want those two to be in the same batch, if they type 3 commands stream the first two and wait for the fourth command. Do not stream more than one batch at a time.

EXAMPLE SESSION

Here is an example session to show you what a run of your program should look like.

- Example Input is prepended with >
- Example output is not prepended with >.

```
> ADD WAREHOUSE 970
> ADD WAREHOUSE 45
> ADD WAREHOUSE 2

> LIST WAREHOUSES
WAREHOUSES
970
45
2

> ADD PRODUCT "Sofia Vegara 5 Piece Living Room Set" 38538505-0767-453f-89af-
d11c809ebb3b
> ADD PRODUCT "BED" 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70
> ADD PRODUCT "TRUNK" 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70
```

```
ERROR ADDING PRODUCT PRODUCT with SKU 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70 ALREADY
EXISTS

> LIST PRODUCTS
Sofia Vegara 5 Piece Living Room Set 38538505-0767-453f-89af-d11c809ebb3b
BED 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70

> STOCK 38538505-0767-453f-89af-d11c809ebb3b 970 1000
> LIST WAREHOUSE 970
ITEM NAME                              ITEM_SKU                               QTY
Sofia Vegara 5 Piece Living Room Set    38538505-0767-453f-89af-d11c809ebb3b  1000
```

SUBMISSION

- You can assume that stocking happens without delay and as soon as the command is issued.
- Add documentation needed to install any packages, run the tests and code to this README/ replace this README with one of your own with information on how to install and run code and tests.
- Add your code to a github repository and share it with us.

Good luck!

Steps in approach ->

Introduction

The original requirement is to build a very simple backend system to fetch data through command line. Adding to the original requirement, I would also like to build a small full-stack system as POC of Microservice and AWS, also I would like to experiment mulit database(SQL/noSQL) access through Springdata. I would like to challenge myself as a one day's job to finish it, some might not be reachable for current approached, which will be continue to implemented later once getting more time.

Design:

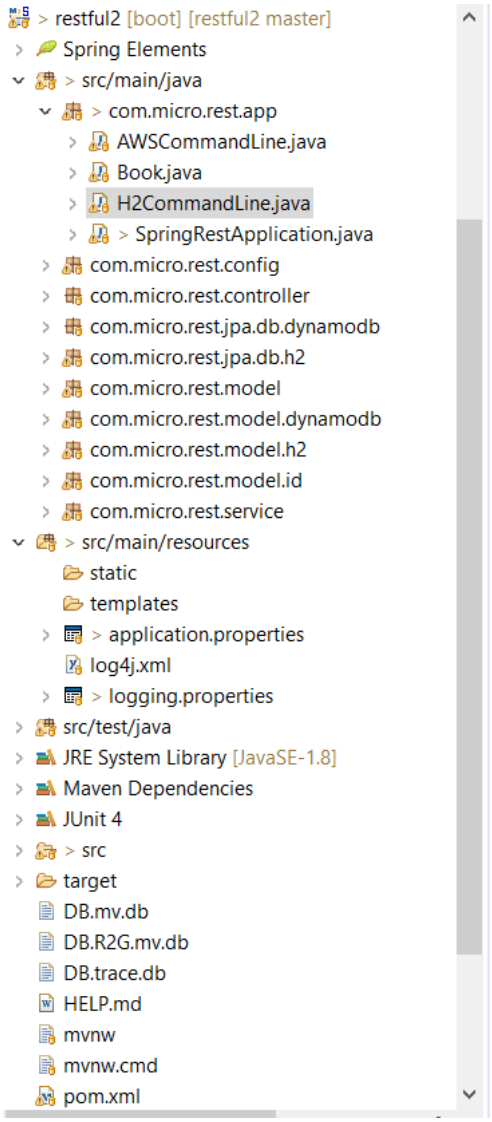
System is implemented on SpringBoot's Microservices/Restful API. User can operate CRUD function through shell client application or web. The system is also designed on multiple database sources: H2 and DynamoDB to compare the SpringBoot implementation on SQL/noSQL, also, I would like to deploy the application to AWS later, which DynamoDB would be ideal economically.

Accomplishment
Application is successfully up running, Full fill the requirement on server side, directly service call from standalone application.
Rest Service is operated correctly on top of SpringBoot

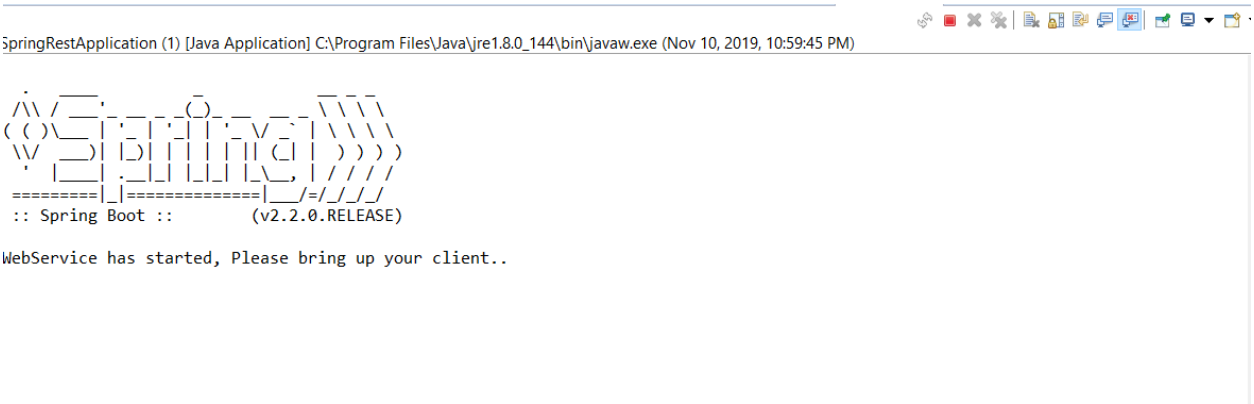
CRUD function is operated correctly on H2
Docker and SAM local have successfully tested on both DynamoDB and H2
Service side implementation on all functions
Add Spring Security on top of microservice, to access remotely through microservice, user criterial needed (client/client)
Implement Client Side function that support the commands, calling through microservice on H2, add microservice, the user still update and fetch data locally, but through microservice.
Add Logs on both Client and Server Side
Build deployment package

Todo
Add logic into stock, the warehouse capacity limitation
Add logic into unstock, should be able to update the current number of products in stock
Enhance on error/exception handler and log mechanism
Enhance on end to end testing
Java9 JShell Implementation
Implement ORM on DynamoDB
Add Web Client calling Microservice
Deploy on AMS
Adding Lambada and API Gateway

Screen Shot of excution, that calling to microservice and back with the info as asked.



Start Server



Run the Command Shell

```
H2CommandLine (1) [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Nov 10, 2019, 11:01:34 PM)
>
ADD WAREHOUSE 970
>ADD WAREHOUSE 45
>ADD WAREHOUSE 2
>LIST WAREHOUSES
970
45
2
>ADD PRODUCT "Sofia Vegara 5 Piece Living Room Set" 38538505-0767-453f-89af-d11c809ebb3b
>ADD PRODUCT "BED" 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70
>ADD PRODUCT "TRUNK" 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70
ERROR ADDING PRODUCT PRODUCT with SKU 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70 ALREADY EXISTS
>LIST PRODUCTS
Sofia Vegara 5 Piece Living Room Set 38538505-0767-453f-89af-d11c809ebb3b
BED 5ce956fa-a71e-4bfb-b6ae-5eeaa5eb0a70
>STOCK 38538505-0767-453f-89af-d11c809ebb3b 970 1000
38538505-0767-453f-89af-d11c809ebb3b
>LIST WAREHOUSE 970
Sofia Vegara 5 Piece Living Room Set 38538505-0767-453f-89af-d11c809ebb3b 1000
>UNSTOCK 38538505-0767-453f-89af-d11c809ebb3b 970 1000
>LIST WAREHOUSE 970
>STOCK 38538505-0767-453f-89af-d11c809ebb3b 970 1000
38538505-0767-453f-89af-d11c809ebb3b
>LIST WAREHOUSE 970
Sofia Vegara 5 Piece Living Room Set 38538505-0767-453f-89af-d11c809ebb3b 1000
>
```

Log File Recorded in R2GFile.log:

```
23:03:43.229 [main] DEBUG o.s.web.client.RestTemplate - Response 200 OK
23:03:43.229 [main] DEBUG o.s.web.client.RestTemplate - Reading to [java.lang.String] as "application/json"
23:03:43.230 [http-nio-8888-exec-5] DEBUG o.s.o.j.s.OpenEntityManagerInViewInterceptor - Closing JPA EntityManager in
OpenEntityManagerInViewInterceptor
23:03:43.230 [http-nio-8888-exec-5] DEBUG o.s.web.servlet.DispatcherServlet - Completed 200 OK
23:03:43.230 [http-nio-8888-exec-5] DEBUG o.s.s.w.a.ExceptionTranslationFilter - Chain processed normally
23:03:43.230 [http-nio-8888-exec-5] DEBUG o.s.s.w.c.SecurityContextPersistenceFilter - SecurityContextHolder now cleared, as request processing
completed
23:03:49.918 [main] INFO  com.micro.rest.app.H2CommandLine - LIST WAREHOUSE 970
23:03:49.919 [main] DEBUG o.s.web.client.RestTemplate - HTTP GET http://localhost:8888/h2/warehouses/970
23:03:49.919 [main] DEBUG o.s.web.client.RestTemplate - Accept=[text/plain, application/json, application/cbor, application/*+json, */*]
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.security.web.FilterChainProxy - /h2/warehouses/970 at position 1 of 9 in additional filter chain;
firing Filter: 'WebAsyncManagerIntegrationFilter'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.security.web.FilterChainProxy - /h2/warehouses/970 at position 2 of 9 in additional filter chain;
firing Filter: 'SecurityContextPersistenceFilter'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.c.HttpSessionSecurityContextRepository - No HttpSession currently exists
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.c.HttpSessionSecurityContextRepository - No SecurityContext was available from the HttpSession:
null. A new one will be created.
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.security.web.FilterChainProxy - /h2/warehouses/970 at position 3 of 9 in additional filter chain;
firing Filter: 'HeaderWriterFilter'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.security.web.FilterChainProxy - /h2/warehouses/970 at position 4 of 9 in additional filter chain;
firing Filter: 'LogoutFilter'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.matcher.OrRequestMatcher - Trying to match using Ant [pattern='/logout', GET]
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.m.AntPathRequestMatcher - Checking match of request : '/h2/warehouses/970'; against '/logout'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.matcher.OrRequestMatcher - Trying to match using Ant [pattern='/logout', POST]
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.m.AntPathRequestMatcher - Request 'GET /h2/warehouses/970' doesn't match 'POST /logout'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.matcher.OrRequestMatcher - Trying to match using Ant [pattern='/logout', PUT]
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.m.AntPathRequestMatcher - Request 'GET /h2/warehouses/970' doesn't match 'PUT /logout'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.matcher.OrRequestMatcher - Trying to match using Ant [pattern='/logout', DELETE]
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.m.AntPathRequestMatcher - Request 'GET /h2/warehouses/970' doesn't match 'DELETE /logout'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.u.matcher.OrRequestMatcher - No matches found
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.security.web.FilterChainProxy - /h2/warehouses/970 at position 5 of 9 in additional filter chain;
firing Filter: 'RequestCacheAwareFilter'
23:03:49.921 [http-nio-8888-exec-7] DEBUG o.s.s.w.s.HttpSessionRequestCache - saved request doesn't match
```

Further Future to do list

(1) Let user choice log level, info, debug or use sam log
(2) Deploy to Docker/Kubernetes
(3) Running local on SAM
(4) Deploy to AWS, Lambada / API Gateway
(5) Testing

1. clone aws-sam-local
2. cd samples/java
3. mvn clean install package
4. echo '{ "some": "input" }' | sam local invoke GetHelloWorld