System Requirements

1. Java 8 or above
2. Maven 3.8 or above
3. Required ENV vars:
   JAVA_HOME - location of a JDK home directory

Run Application

1. Run existing unit test:

```
$ mvn clean test
```

2. Run as a standalone application

```
$ mvn clean install
$ java -jar target/SpotPrice-0.0.1-SNAPSHOT.jar
```

Or

```
$ mvn spring-boot:run
```

The application is listening port 5000. It can be changed by editing application.properties.
src/main/resources/application.properties

```
# web port
server.port=5000
```

Loading Data When the Application Start

1. Data for a standalone application

The application tries to load data from file “data.json” when it starts. It searches the file in the current folder. If the file does not exist, it tries to search the file in classpath.

2. Data for unit test

Each test case will clear and reload test data before it runs. The test data file name is “data_test.json”. Same as loading data for a standalone application, it searches file in the current folder and classpath.

Note:

a. The existing testcases need at least two rates to run. The endpoint *rates* can pass new input data.
b. The test cases for endpoint *price* were hardcode in java file. Please see the cases below.

```
{"2015-07-01T07:00:00-05:00", "2015-07-01T12:00:00-05:00", "true"},
{"2015-07-01T12:00:00-05:00", "2015-07-01T07:00:00-05:00", "false"}, // end < start
{"2015-07-04T15:00:00+00:00", "2015-07-04T20:00:00+00:00", "true"},
{"2015-07-04T07:00:00+05:00", "2015-07-04T20:00:00+05:00", "false"} // in two days
```

Endpoints

Based on the requirements of taking home tests, there are two endpoints that have been implemented. Also, I implemented another version which can be used easier to access a single rate. In the requirements, the application returns Json object or String object based on the results. Version 2 is trying to always use Json object as response body. For example, if there is no price found, the requirement returns a String object "unavailable". Version 2 returns a Json data {"message":"unavailable"}.

1. Endpoint *rates*

Endpoint *rates* support two HTTP methods: GET and PUT. GET method can get all rates stored on server. PUT method uploads new rates data for updating data on server.

- GET URLs

```
http://localhost:5000/rates

http://localhost:5000/v2/rates
```

Those two versions' URLs will return all rates stored on server. The difference is that the version 2 has an id attribute for a rate. In version 2, there is another URL to access a single rate with id in the URL as below.

```
http://localhost:5000/v2/rates/{id}
```

Here is an example:

```
{
  "id":2,
  "days":"fri,sat,sun",
  "times":"0900-2100",
  "tz":"America/Chicago",
   "price":12345
}
```

PUT URLs

```
http://localhost:5000/rates

http://localhost:5000/v2/rates
```

Those two versions' URLs have the same behaviors at server side. Since it is hard to identify which attribute can be a key of a rate, the application will place the whole stored data which the json data in request body. It replaces the data on the server.
Version 2 gives a chance to update a single rate. Here is version 2 URL for updating a single rate.

```
http://localhost:5000/v2/rates/{id}
```

2. Endpoint *price*

This endpoint only supports HTTP GET method. Example URLs:

http://localhost:5000/price?start=2015-07-01T07:00:00-05:00&end=2015-07-01T12:00:00-05:00

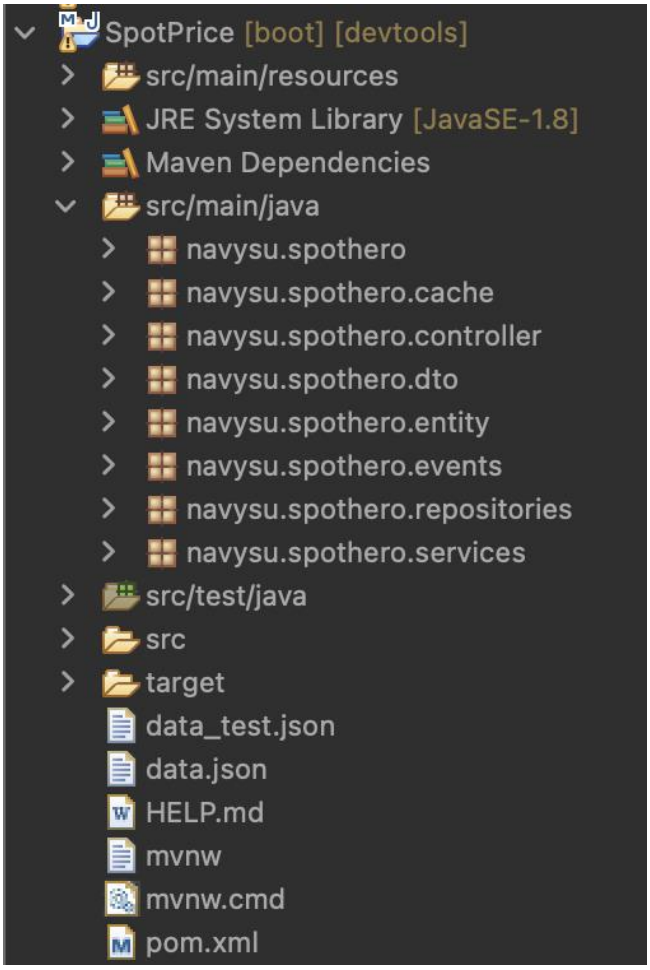
http://localhost:5000/v2/price?start=2015-07-01T07:00:00-05:00&end=2015-07-01T12:00:00-05:00

Those two versions' URLs have same behavers at the server side to find an available price.

Security

The endpoint *price* can be opened to the public, but the endpoint *rates* should be protected with JWT. The security fix has not been implemented in this submission.

File Structure of the application



Java source codes are in *src/main/java*. Test codes are in src/test/java
The java package name is based on the classes function or role. Such as Controller is the restful controller classes, entity is the class entity mapping to database, and services supports business process and get/save data into databases.