Weather Service

Summary

This document provides details of "weather web application", intended to serve weather reports to its customers via REST endpoints. The REST endpoints provide the ability to add, query and delete weather data. The application has been developed using Spring Boot and JDK 1.8.

Steps to Execute

1. Ensure maven and java 1.8 is installed
2. There are 2 ways to get the source code:
    a. Unzip the attachment from the email - **weather-service-master.zip**
    b. Download code from this link - https://github.com/pallavisoni03/weather-service/archive/refs/heads/master.zip
    c. Or from command line run (this requires git installed in the system) -
        git clone git@github.com:pallavisoni03/weather-service.git
3. From command line, go to weather-service directory
4. To run the tests (Integration and Unit tests) and to compile, run "**mvn clean install**" from the command line
5. To run spring boot app, run "**mvn spring-boot:run**" from the command line. This should start the server at the port 8080, which can be confirmed by seeing the logs on the console. Sample log like this
    Initializing Servlet 'dispatcherServlet'
    Completed initialization in 2 ms
6. Sending requests from command line

To create weather data in ASC order by ID	curl --location --request POST 'http://localhost:8080/weather' \
	--header 'Content-Type: application/json' \
	--data-raw '{
	"id": 1,
	"date": "1986-01-02",
	"location": {
	"lat": 36.1189,
	"lon": -86.6892,
	"city": "Palo Alto",
	"state": "California"
	},
	"temperature":
	[37.3,36.8,36.4,36.0,35.6,35.3,35.0,34.9,35.8,38.0,40.2,42.3,43.8,44.9,45.5,45.7,44.9,43.0,41.7,40.8,39.9,39.2,38.6,38.1]
	}'

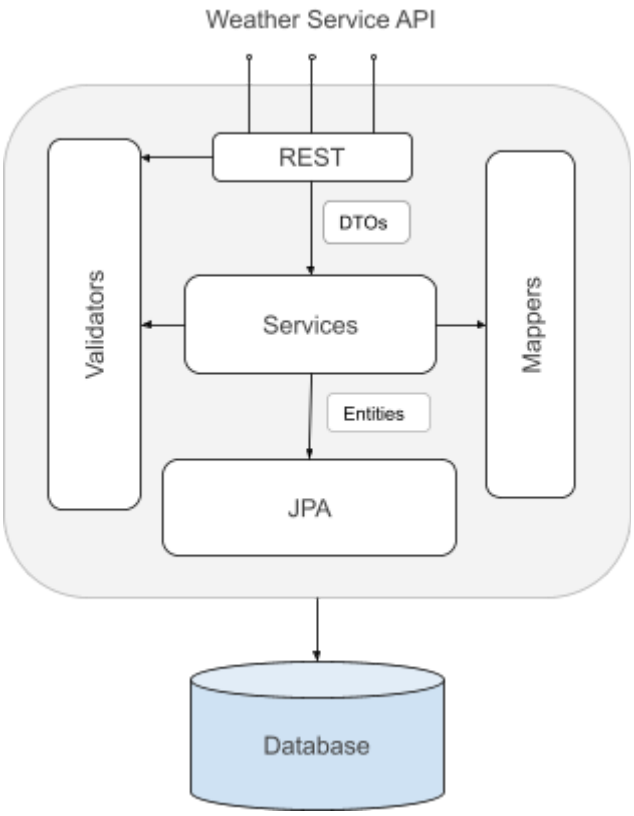
To get all weather data	curl --location --request GET 'http://localhost:8080/weather'
Search weather data by date	curl --location --request GET 'http://localhost:8080/weather?date=1986-01-02'
Get data by id	curl --location --request GET 'http://localhost:8080/weather/1'
Erase all data	curl --location --request DELETE 'http://localhost:8080/erase'

7. Sending requests from any client(ex. Postman):
    a. To create weather data (Sample data below) - POST http://localhost:8080/weather


```
{
  "id": 1,
  "date": "1986-01-02",
  "location": {
    "lat": 36.1189,
    "lon": -86.6892,
    "city": "Palo Alto",
    "state": "California"
    },
    "temperature":
[37.3,36.8,36.4,36.0,35.6,35.3,35.0,34.9,35.8,38.0,40.2,42.3,43.8,44.9,45.5,45.7,44.9,43.0,41.7,40.8,39.9,39.2,38.6,
38.1]
}
```


    b. To get all weather data - GET http://localhost:8080/weather
    c. Search weather data by date - GET http://localhost:8080/weather?date=1986-01-02
    d. Erase all data - DELETE - http://localhost:8080/erase

Design



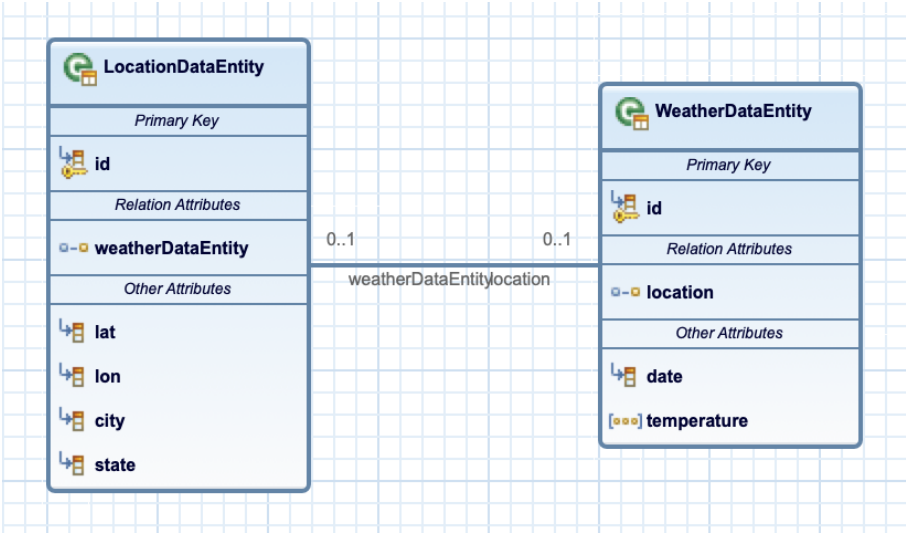
Tech Stack / Dependencies

REST	Spring controller
Persistence Layer	Spring JPA
Database	H2 Database
Validation	Apache Commons Validator
Unit Testing	JUnit
Logging	SLF4J
REST Client	Postman

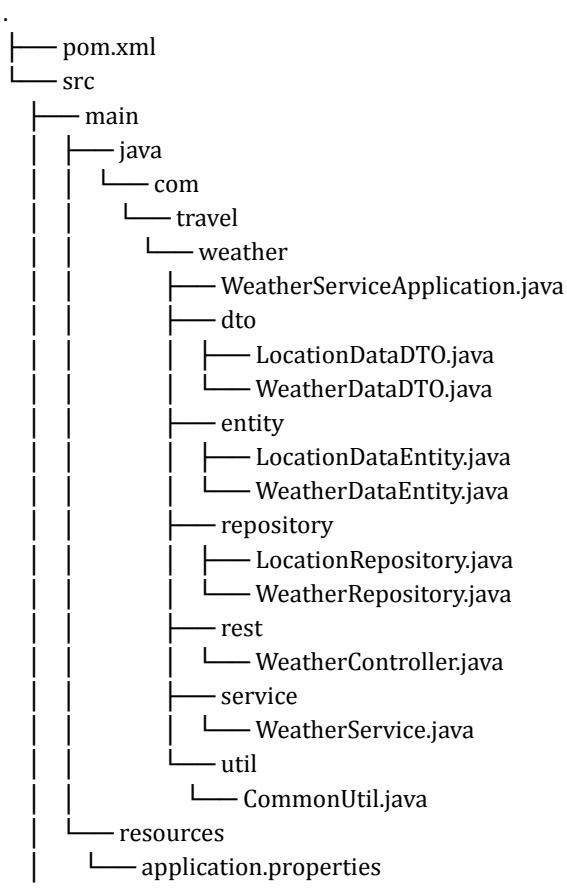
REST Endpoints

Method	Functionality	Endpoints	Response	Validations
POST	Adding new weather data	<host>/weather	201 OK Header: location = <host>/weather/{id}	400 Id already exixts Check for valid values ID, City, Date, Lat, Lon
GET	Returning all the weather data	<host>/weather	200 OK In ASC order of ID	
GET	Returning the weather data filtered by date	<host>/weather?date={date}	200 OK	Check for valid value of Date
DELETE	Erasing all the weather data	<host>/erase	200 OK	

ER Diagram



Folder Structure



References

1. https://spring.io/guides/tutorials/rest/
2. https://commons.apache.org/proper/commons-validator/
3. http://modelmapper.org/getting-started/
4. https://www.baeldung.com/jpa-one-to-one