

User Guide for Weather Report APIs

1. Introduction

This document is a guide to build and run the Weather Report API application once the source code has been downloaded. It also shows some testings at the end of the document for references.

Note: The source code is assumed to be downloaded into the directory C:\temp

2. Build Executable Jar File

Install the Maven application in your PC and then execute the following command to build the Jar file in a Command Prompt window

```
C:\temp\weather-app-main\Weather-App> mvn install package
```

3. Run The Application

Before running the application, please be sure that the JDK 17 is already installed.

Execute the following commands to start the application Weather App.

```
C:\temp\weather-app-main\Weather-App>cd target
```

```
C:\temp\weather-app-main\Weather-App\target>java -jar weather-app-1.0.jar
```

The result should look like so:

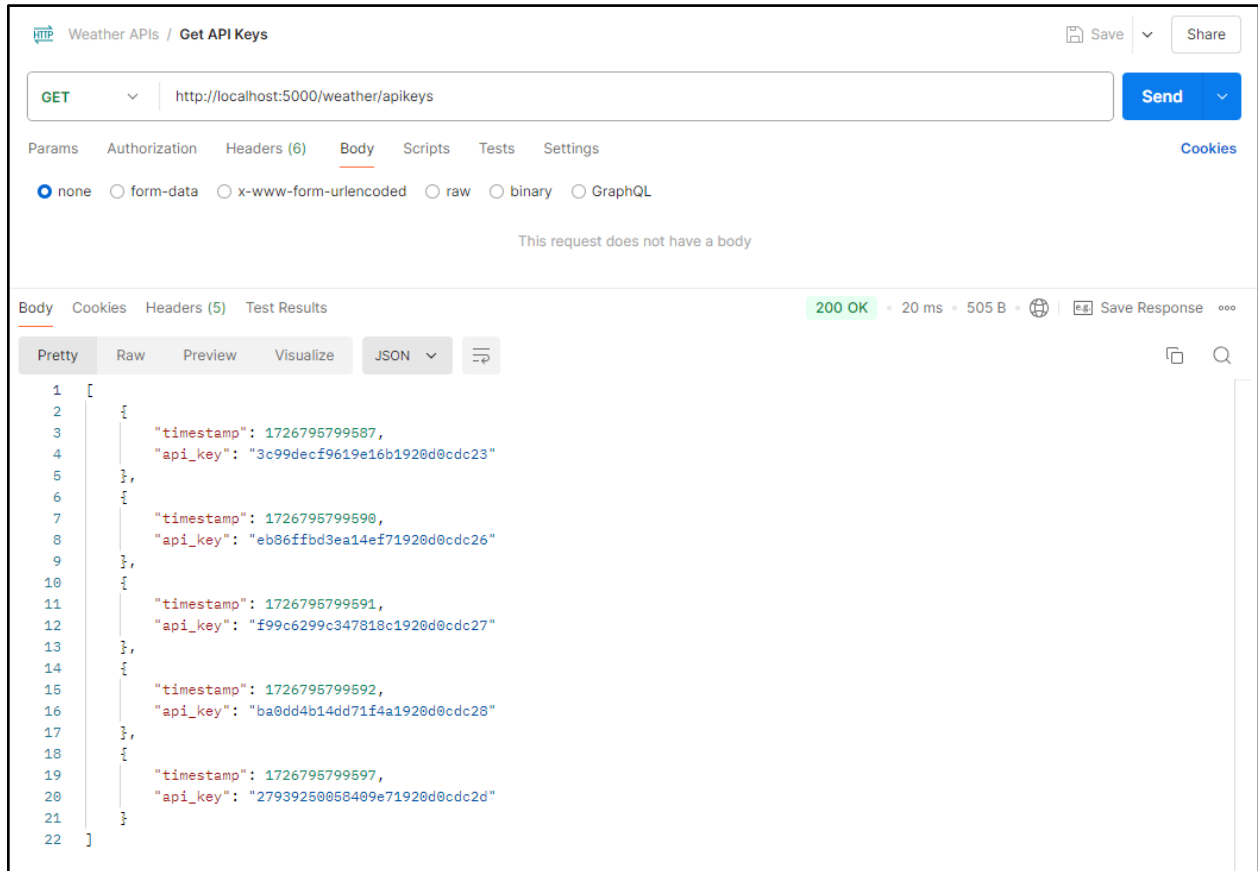
[illegible]

Please note that the embedded Tomcat run on port 5000.

4. Calling Weather Report APIs

It is recommended to use the tool Postman to call the APIs. Below are illustrations of how to send a GET request to a list of five API keys and weather report description.

The GET request end point to get API keys:**<http://localhost:5000/weather/apikeys>**



http://localhost:5000/weather/report?q=paris,france&apiKey=3c99decf9619e16b1920d0cdc23

Weather APIs / Get Weather Report

Save

Share

GET

http://localhost:5000/weather/report?q=paris,france&apiKey=3c99decf9619e16b1920d0cdc23

Send

Params

Authorization

Headers (6)

Body

Scripts

Tests

Settings

Cookies

Query Params

| <input checked="" type="checkbox"/> Key | Value | Description | ... | Bulk Edit |
|--|-----------------------------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> q | paris,france | | | |
| <input checked="" type="checkbox"/> apiKey | 3c99decf9619e16b1920d0cdc23 | | | |
| <input type="checkbox"/> Key | Value | Description | | |

Body

Cookies

Headers (5)

Test Results

200 OK

15 ms

191 B

Save Response

...

Pretty

Raw

Preview

Visualize

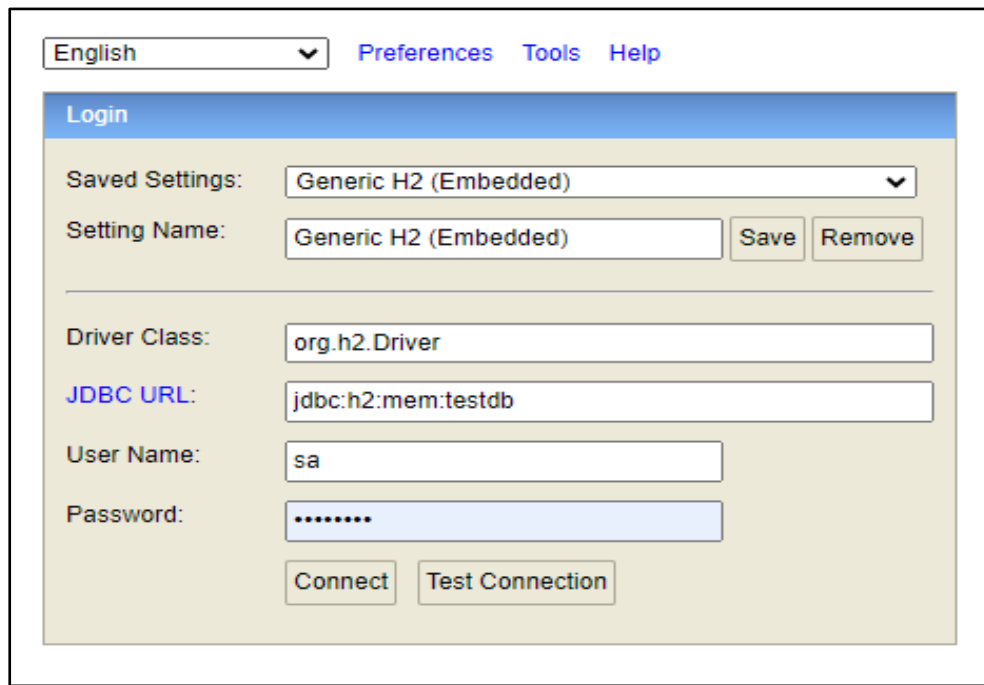
JSON

```
1 {
2   |   "description": "clear sky"
3 }
```

5. H2 Database

When a user gets a weather report for the first time, the report data is stored the H2 database. To inspect the weather reports, please login to the local H2 database by following these steps:

- Enter this URL on a browser: <http://localhost:5000/h2-console>
- Enter the details in the login pop-up window as shown below:
The password is “password”.



The screenshot shows the H2 Database console interface. At the top, there is a language dropdown menu set to 'English' and three links: 'Preferences', 'Tools', and 'Help'. Below this is a 'Login' section with a blue header. The 'Saved Settings' dropdown is set to 'Generic H2 (Embedded)'. Below it, the 'Setting Name' is also 'Generic H2 (Embedded)', with 'Save' and 'Remove' buttons. The 'Driver Class' is 'org.h2.Driver'. The 'JDBC URL' is 'jdbc:h2:mem:testdb'. The 'User Name' is 'sa'. The 'Password' field is masked with dots. At the bottom, there are 'Connect' and 'Test Connection' buttons.

| | | | |
|-----------------|-----------------------|-----------------|--------|
| English | Preferences | Tools | Help |
| Login | | | |
| Saved Settings: | Generic H2 (Embedded) | | |
| Setting Name: | Generic H2 (Embedded) | Save | Remove |
| Driver Class: | org.h2.Driver | | |
| JDBC URL: | jdbc:h2:mem:testdb | | |
| User Name: | sa | | |
| Password: | | | |
| Connect | | Test Connection | |

Once logged in, execute some “SELECT” statements as shown below:

The screenshot shows a database client interface with a sidebar on the left displaying the database structure: jdbc:h2:mem:testdb, API_KEY, COUNTRY, WEATHER_REPORT_DETAILS, INFORMATION_SCHEMA, Users, and H2 2.1.214 (2022-06-13). The main area contains a SQL statement editor with two queries. The first query, 'SELECT * FROM API_KEY;', is executed, resulting in a table with 5 rows and 3 columns: API_KEY, TIMESTAMP, and NUMBER_OF_TIME_USED. The second query, 'SELECT * FROM WEATHER_REPORT_DETAILS;', is also executed, resulting in a table with 1 row and 5 columns: ID, CITY, COUNTRY, DESCRIPTION, and TIMESTAMP.

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM API_KEY ;
```

```
SELECT * FROM WEATHER_REPORT_DETAILS |
```

```
SELECT * FROM API_KEY ;
```

| API_KEY | TIMESTAMP | NUMBER_OF_TIME_USED |
|-----------------------------|---------------|---------------------|
| f7a12491104ccca21920d151bac | 1726796340140 | 1 |
| 937be0c768b882571920d151bae | 1726796340142 | 0 |
| fe3bcd3b6492a47d1920d151bb0 | 1726796340144 | 0 |
| e9c46dda3045224f1920d151bb4 | 1726796340148 | 0 |
| 445bed030e4f577a1920d151bb5 | 1726796340149 | 0 |

(5 rows, 1 ms)

```
SELECT * FROM WEATHER_REPORT_DETAILS;
```

| ID | CITY | COUNTRY | DESCRIPTION | TIMESTAMP |
|----|-------|---------|-------------|---------------|
| 1 | Paris | France | clear sky | 1726795695515 |

(1 row, 1 ms)

The above image shows a table of five API keys and a weather report for city Paris in France.

6. Testing

This section shows some “unhappy path” testings on the Weather Report API.

6.1 Testing Parameter “q”

The API will return the following error when the parameter “q” does not exist in the URL

The screenshot shows a REST client interface for a "Weather APIs / Get Weather Report" endpoint. The request method is GET and the URL is `http://localhost:5000/weather/report?apiKey=f7a12491104ccca21920d151bac`. The "Query Params" section shows a table with parameters: "q" (value: "paris,france") and "apiKey" (value: "f7a12491104ccca21920d151bac"). The "Body" tab is selected, showing a "400 Bad Request" status with a response time of 9 ms and a size of 288 B. The response body is displayed in JSON format:

```
1 {
2   "type": "about:blank",
3   "title": "Bad Request",
4   "status": 400,
5   "detail": "Required parameter 'q' is not present.",
6   "instance": "/weather/report"
7 }
```

6.2 Testing Parameter “apiKey”

The API will return the following error when the parameter “apiKey” does not exist in the URL

The screenshot shows a REST client interface for a "Weather APIs / Get Weather Report" endpoint. The method is GET and the URL is `http://localhost:5000/weather/report?q=paris,france`. The "Params" tab is active, showing a table of query parameters:

| Key | Value | Description |
|---------------------------------------|-----------------------------|-------------|
| <input checked="" type="checkbox"/> q | paris,france | |
| <input type="checkbox"/> apiKey | f7a12491104ccca21920d151bac | |
| Key | Value | Description |

The response status is "400 Bad Request" (6 ms, 293 B). The response body is shown in JSON format:

```
1 {
2   "type": "about:blank",
3   "title": "Bad Request",
4   "status": 400,
5   "detail": "Required parameter 'apiKey' is not present.",
6   "instance": "/weather/report"
7 }
```

6.3 Testing Query

The following error will be returned when the country name is missing from the URL.

The screenshot shows the same REST client interface, but the URL is `http://localhost:5000/weather/report?q=paris&apiKey=f7a12491104ccca21920d151bac`. The "Params" tab is active, showing a table of query parameters:

| Key | Value | Description |
|--|-----------------------------|-------------|
| <input checked="" type="checkbox"/> q | paris | |
| <input checked="" type="checkbox"/> apiKey | f7a12491104ccca21920d151bac | |
| Key | Value | Description |

The response status is "400 Bad Request" (7 ms, 282 B). The response body is shown in JSON format:

```
1 {
2   "type": "about:blank",
3   "title": "Bad Request",
4   "status": 400,
5   "detail": "Missing city and/or country name",
6   "instance": "/weather/report"
7 }
```

The same error will be returned when the city name is missing from the URL:

The screenshot shows a REST client interface for a 'Weather APIs / Get Weather Report' endpoint. The method is GET and the URL is `http://localhost:5000/weather/report?q=united states&apiKey=f7a12491104ccca21920d151bac`. The 'Query Params' section shows a table with the following data:

| Key | Value | Description |
|--------|-----------------------------|-------------|
| q | united states | |
| apiKey | f7a12491104ccca21920d151bac | |

The 'Body' tab is selected, showing a JSON response with the following content:

```
{
  "type": "about:blank",
  "title": "Bad Request",
  "status": 400,
  "detail": "Missing city and/or country name",
  "instance": "/weather/report"
}
```

The status bar indicates a '400 Bad Request' error with a response time of 8 ms and a size of 282 B.

And the same error will be displayed when the query is empty.

The screenshot shows the same REST client interface, but the URL is `http://localhost:5000/weather/report?q=&apiKey=f7a12491104ccca21920d151bac`. The 'Query Params' section shows a table with the following data:

| Key | Value | Description |
|--------|-----------------------------|-------------|
| q | | |
| apiKey | f7a12491104ccca21920d151bac | |

The 'Body' tab is selected, showing the same JSON response as the previous screenshot:

```
{
  "type": "about:blank",
  "title": "Bad Request",
  "status": 400,
  "detail": "Missing city and/or country name",
  "instance": "/weather/report"
}
```

The status bar indicates a '400 Bad Request' error with a response time of 8 ms and a size of 282 B.

Extra tests are welcome for this API.