

# 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:

```

  \ / _ _ _ _ _ / _ _ _ _ _ \ / _ _ _ _ _
 (C) / _ _ _ _ _ \ / _ _ _ _ _ \ / _ _ _ _ _
 W / _ _ _ _ _ \ / _ _ _ _ _ \ / _ _ _ _ _
  / _ _ _ _ _ \ / _ _ _ _ _ \ / _ _ _ _ _
  / _ _ _ _ _ \ / _ _ _ _ _ \ / _ _ _ _ _

2024-09-20T11:15:27.218+00:00 INFO 6320 --- [
main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4099 ms
2024-09-20T11:15:27.319+00:00 INFO 6320 --- [
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-09-20T11:15:27.919+00:00 INFO 6320 --- [
main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com0: url=jdbc:h2:mem:testdb user=SA
2024-09-20T11:15:27.934+00:00 INFO 6320 --- [
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-09-20T11:15:27.969+00:00 INFO 6320 --- [
main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console', Database available at 'jdbc:h2:mem:testdb'
2024-09-20T11:15:28.390+00:00 INFO 6320 --- [
main] o.hibernate.jta.internal.util.LogHelper : HH0000294: Processing PersistenceUnitInfo [name: default]
2024-09-20T11:15:28.541+00:00 INFO 6320 --- [
main] org.hibernate.Version : HH0000412: Hibernate ORM core version 6.5.2.Final
2024-09-20T11:15:28.608+00:00 INFO 6320 --- [
main] o.h.c.internal.RegionFactoryInitiator : HH0000926: Second-level cache disabled
2024-09-20T11:15:29.525+00:00 INFO 6320 --- [
main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup; ignoring JPA class transformer
2024-09-20T11:15:29.151+00:00 INFO 6320 --- [
main] o.h.e.t.j.p.a.JtaPlatformInitiator : HH0000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2024-09-20T11:15:31.154+00:00 INFO 6320 --- [
main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-09-20T11:15:32.876+00:00 WARN 6320 --- [
main] jpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly
configure spring.jpa.open-in-view to disable this warning
2024-09-20T11:15:34.762+00:00 INFO 6320 --- [
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 5000 (http) with context path '/'
2024-09-20T11:15:34.868+00:00 INFO 6320 --- [
main] weatherapi.WeatherApiApplication : Started WeatherApiApplication in 13.047 seconds (process running for 14.397)

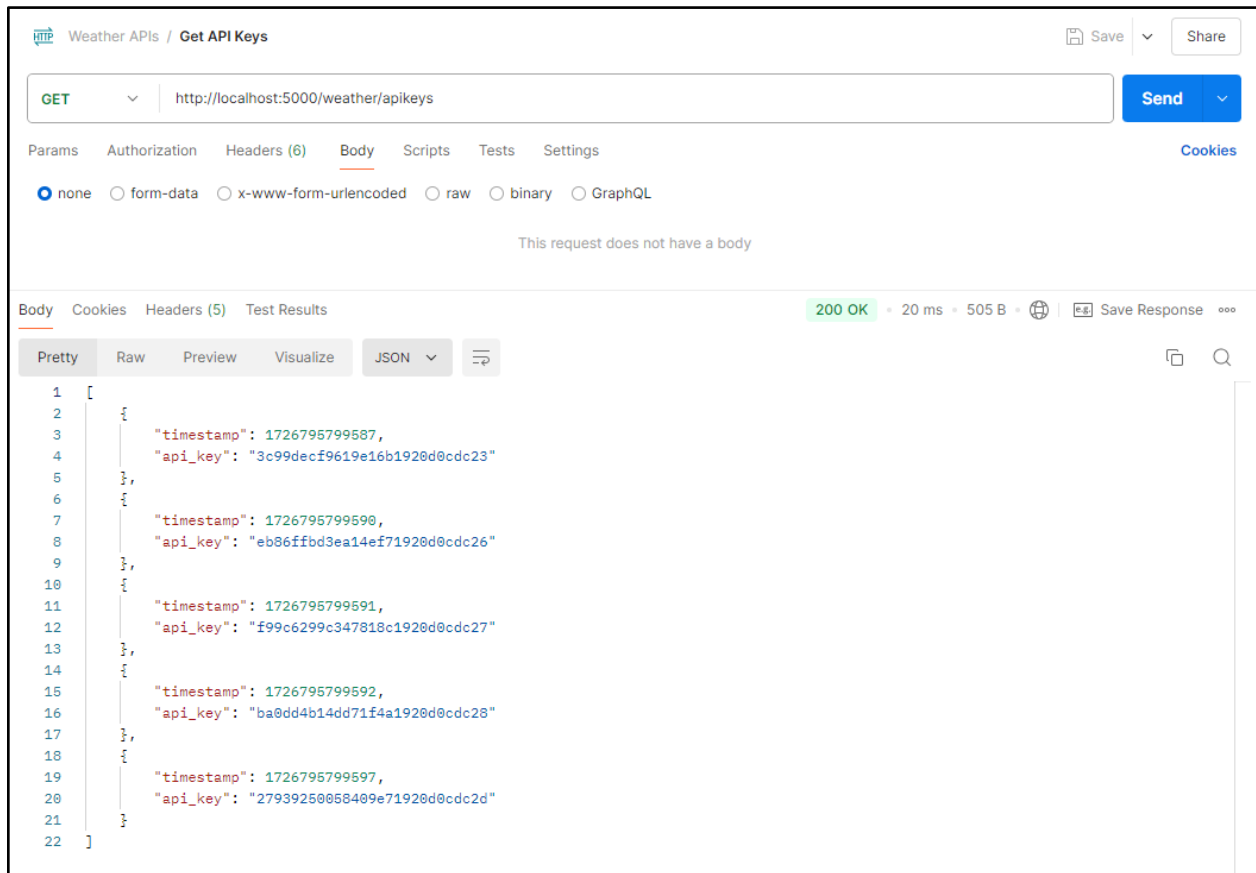
```

**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 get a list of five API keys and a weather report description.

The GET request end point to get five 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

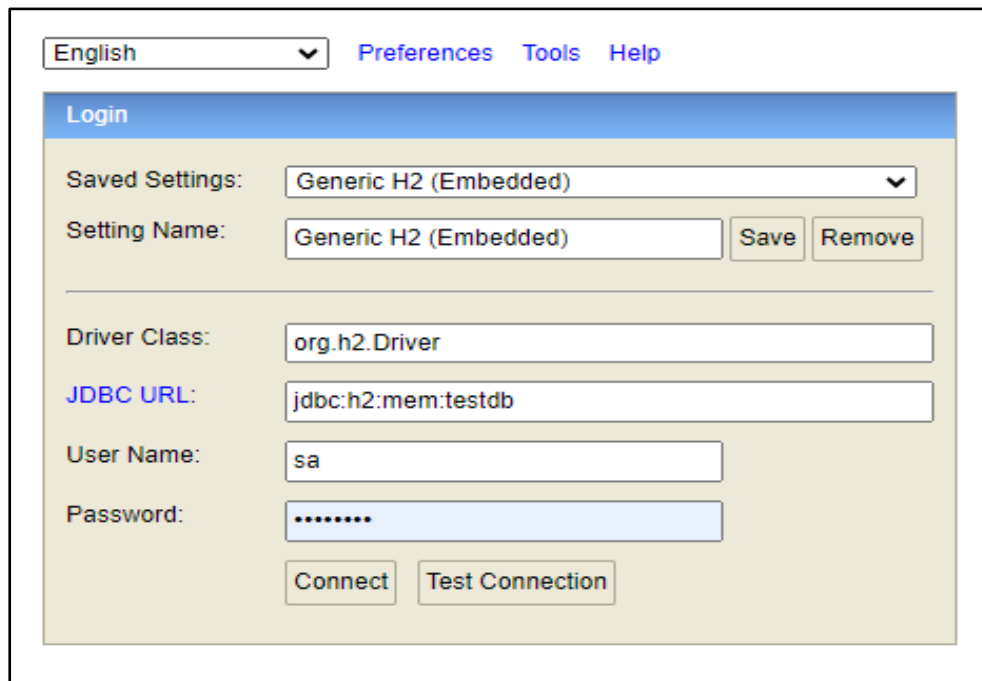
JSON

```
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 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 login window. 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. Inside the login section, there are several fields and buttons:

- Saved Settings:** A dropdown menu showing 'Generic H2 (Embedded)'.
- Setting Name:** A text field containing 'Generic H2 (Embedded)', followed by 'Save' and 'Remove' buttons.
- Driver Class:** A text field containing 'org.h2.Driver'.
- JDBC URL:** A text field containing 'jdbc:h2:mem:testdb'.
- User Name:** A text field containing 'sa'.
- Password:** A text field containing seven dots, representing the password.
- At the bottom, there are two buttons: 'Connect' and '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" response. The response status is 400, with a message "Required parameter 'q' is not present." and an instance path of `/weather/report`.

Key	Value	Description
q	paris,france	
apiKey	f7a12491104ccca21920d151bac	

```
{
  "type": "about:blank",
  "title": "Bad Request",
  "status": 400,
  "detail": "Required parameter 'q' is not present.",
  "instance": "/weather/report"
}
```

## 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 "Query Params" section shows a table with two entries: "q" with value "paris,france" and "apiKey" with value "f7a12491104ccca21920d151bac". The "Body" tab is selected, showing a JSON response with a 400 status and a message indicating the missing apiKey.

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

```
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 "Query Params" section shows a table with three entries: "q" with value "paris", "apiKey" with value "f7a12491104ccca21920d151bac", and "apiKey" with value "f7a12491104ccca21920d151bac". The "Body" tab is selected, showing a JSON response with a 400 status and a message indicating the missing city and/or country name.

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

```
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 contains three parameters: `q` with value `united states`, `apiKey` with value `f7a12491104ccca21920d151bac`, and an empty `Key` field. The "Body" tab is selected, showing a JSON response with a 400 status and a message indicating a missing city or country name.

Key	Value	Description
<input checked="" type="checkbox"/> q	united states	
<input checked="" type="checkbox"/> apiKey	f7a12491104ccca21920d151bac	
<input type="checkbox"/> Key	Value	Description

```
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 }
```

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

This screenshot is identical to the previous one, but the query parameter `q` is empty in the URL: `http://localhost:5000/weather/report?q=&apiKey=f7a12491104ccca21920d151bac`. The JSON response remains the same, indicating a "Missing city and/or country name".

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

```
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 }
```

Extra tests are welcome for this API.