OWM JAPIs

Java Wrapper Library for OpenWeatherMap.org Web APIs

Create weather-aware applications for **Java and Android platforms** in minimum time using OWM JAPIs, an easy-to-use, detailed and documented weather API library for retrieving weather data from OpenWeatherMap.org. You can easily **retrieve and use weather data** in your applications using this library.

OWM JAPIs lets you **get weather data in only 3-5 lines of code** (excluding any other/skeleton code, of course). You can develop applications for multiple platforms using this library, such as Windows, Mac OS X, Linux, and Android.

Why to use OWM JAPIs?

1. Free
2. Easy to use
3. Minimizes your code

OWM JAPIs lets you **focus just on your application's logic** and **weather retrieval code is provided** by this library. Additionally, weather retrieval code becomes very short using this library – as less as 3-5 lines of code can get you weather data from OpenWeatherMap.org in your Java or Android application. **Surprising, right? Have a look on the example(s) below.**

Downloads

Download the library's source and binaries from OWM JAPIs Downloads.

Versions

2.5 (Compatible with OpenWeatherMap.org's API v2.5)

2.5.0.3 (latest)

Implemented:

1. Current Weather
2. Daily Forecasts
3. Hourly Forecasts

New Features:

1. Faster than ever before
2. Raw Response for Caching purposes
3. APIs' URL building using StringBuilder
4. Multi-lingual (multiple languages) support
5. Support for external/third-party HTTP libraries (like Apache's HttpComponents)
6. Units and Language enums for setting configuration easily and correctly
7. Better maintain-able source code (for developers)
8. Ported the project to Gradle (for developers)

Changed:

1. Package's name from net.aksingh.java.api.owm to net.aksingh.owmjapis
2. Class's name from CurrentWeatherData to CurrentWeather
3. Class's name from DailyForecastData to DailyForecast
4. Class's name from ForecastWeatherData to HourlyForecast
5. Some functions' name and signature

Apologies for making such changes, but it was required to make things simpler. Don't worry, they're not going to change again. :)

2.5.0.2

Bug-fix version:

1. Fixed bugs which caused wrong parsing of date and time.
2. Improved code formatting and readability (for developers).

2.5.0.1

Implemented:

1. Current Weather
2. Weather Forecasts
3. Daily Forecasts
4. Wind degree to direction converter

Not implemented but planned:

1. Searching of City
2. Weather Maps
3. Country code to name converter
4. Direction code to name converter

How to use OWM JAPIs?

Anyone with little coding knowledge of Java will feel at home while using this library. **Identifiers are written to be self-explanatory and APIs' documentation** is also provided. It makes the coding process very easy, even for beginners.

1. Add this JAR file in your project's libraries:
    1. owm-japis.jar
2. Write your code as such:
    1. Create and initialize object {obj1} of "OpenWeatherMap" class
    2. Call this object's {obj1} functions to get the desired weather data (such as current weather, daily forecast, etc.).
    3. The data is returned as a new object {obj2} of a compatible class based on the type of asked/retrieved weather data (current weather data comes in a different class's object than daily forecast data).
    4. Call this returned object's {obj2} functions to get the required information from the collective weather data (such as temperature, pressure, wind speed, etc.).

Kindly have a look on the example(s) below for clear understanding.

Example

Basic Example

Sample Code

```
import java.io.IOException;
import java.net.MalformedURLException;
import net.aksingh.owmjapis.CurrentWeather;
import net.aksingh.owmjapis.OpenWeatherMap;
import org.json.JSONException;

public class OwmJapisExample1 {

    public static void main(String[] args)
            throws IOException, MalformedURLException, JSONException {

        // declaring object of "OpenWeatherMap" class
        OpenWeatherMap owm = new OpenWeatherMap("");

        // getting current weather data for the "London" city
        CurrentWeather cwd = owm.currentWeatherByCityName("London");
```

```
        //printing city name from the retrieved data
        System.out.println("City: " + cwd.getCityName());

        // printing the max./min. temperature
        System.out.println("Temperature: " + cwd.getMainInstance().getMaxTemperature()
                           + "/" + cwd.getMainInstance().getMinTemperature() + "\'F");
    }
}
```

Output

```
City: London
Temperature: 73.4/68.72 'F
```

Advance Example

You can simply use the APIs (as given in basic example) for learning, testing or experimenting with the functions provided in this library. But it may not be good enough for production or deployment environment.

Professionally, you should always **write code which can handle errors/exceptions** at the runtime. OWM JAPIs also helps here – by providing checker functions which allows you to **check if a data is available or not**, i.e., that particular data is retrieved and parsed properly or not. Of course, exception handling can still be used, but these functions are really useful and make the retrieved-data-error-handling task very simple.

Using OWM JAPIs, you can always check if a particular data is available or not. This is done by using the **has<DataName>()** functions. For example, **hasResponseCode()** function checks if the retrieved data has a response code or not; and if available, response code can be used to check if the whole data was downloaded and parsed correctly or not.

Sample Code

```
import java.io.IOException;
import java.net.MalformedURLException;
import net.aksingh.owmjapis.CurrentWeather;
import net.aksingh.owmjapis.OpenWeatherMap;
import org.json.JSONException;

public class OwmJapisExample2 {

    public static void main(String[] args)
            throws IOException, MalformedURLException, JSONException {

        // declaring object of "OpenWeatherMap" class
        OpenWeatherMap owm = new OpenWeatherMap("");

        // getting current weather data for the "London" city
        CurrentWeather cwd = owm.currentWeatherByCityName("London");

        // checking data retrieval was successful or not
        if (cwd.isValid()) {

            // checking if city name is available
            if (cwd.hasCityName()) {
                //printing city name from the retrieved data
                System.out.println("City: " + cwd.getCityName());
            }

            // checking if max. temp. and min. temp. is available
            if (cwd.getMainInstance().hasMaxTemperature() && cwd.getMainInstance().hasMinTemperature()) {
                // printing the max./min. temperature
                System.out.println("Temperature: " + cwd.getMainInstance().getMaxTemperature()
                           + "/" + cwd.getMainInstance().getMinTemperature() + "\'F");
            }
        }
    }
}
```

Output

```
City: London
Temperature: 73.4/68.72 'F
```

Source code

Download the library's source code from OWM JAPIs Source.

Bugs / Requests

Got a problem, error or bug in the library? Or want a new feature that's not present in OWM JAPIs?

Kindly post bugs or feature requests at OWM JAPIs Bugs/Requests and I will try to solve/add it in the next release.

Developer

Ashutosh Kumar Singh | AKSingh.net | me@aksingh.net

Credits

1. OpenWeatherMap.org for providing free weather data and creating easy-to-use web APIs.
2. JSON.org for providing such a great data interchange language and its library in Java.
3. ForecastIO-Lib-Java for providing ideas like support for third-party Http libraries.
4. Bug Reporters for reporting bugs, and even finding and sharing possible solutions for them.

License