Creating Well-Structured Output for API Clients Using Postman to Get Weather Report

1. Problem Statement

To get the weather report in a well-structured output, we need to have a set of APIs of the weather application and automatable tool like Postman.

1. Requisites

* Postman
* Endpoint URL(<https://samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22>)

1. Procedure
   1. Endpoint URL: <https://samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22>

Let’s breakdown the components of this Endpoint URL:

BaseURL : <https://samples.openweathermap.org>

APIVersion : /data/2.5

EndpointPath : /weather

Query Parameters:

**q=London,uk** specifies the location for which weather data is requested.

**appid=b6907d289e10d714a6e88b30761fae22** is the API key parameter.

You can make a GET request to this endpoint using a tool like Postman, passing the API key as a query parameter. The response will contain the weather data for London, UK.

* 1. Open Postman and create a new collection – **WeatherMap**
  2. Create a new request within the collection by clicking on the "New" button or by right-clicking on the collection and selecting "Add Request." – **WeatherDetail\_London**
  3. In the request editor, enter the URL without the query parameters. In this case, the URL would be `https://samples.openweathermap.org/data/2.5/weather`.Set the BaseURL and APIVersion and EndpointPath Variables as Collection variables scope.
  4. Click on the “Params” tab located below the URL field.
  5. Click on the "Add Parameter" button.
  6. 6. In the "Key" field, enter `q`, and in the "Value" field, enter `London,uk`. These values represent the query parameter key-value pair.
  7. Click on the "Add Parameter" button again to add another parameter.
  8. In the new parameter, enter `appid` as the key and your actual API key as the value. This is the parameter for your OpenWeatherMap API key.
  9. Click on the "Send" button to make the request with the added query parameters.
  10. To save this request within the collection, click on the "Save"
  11. By following these steps, you can add query parameters to a request within a Postman collection. The parameters will be included when making the request, allowing you to customize the API call with the specific values you need.

1. Process and Structure the Response:

Once you receive the response from the weather API, we can use Postman's built-in tools and scripting capabilities to process and structure the data as needed. You can extract specific information, format it, or store it in a well-structured output format such as JSON or CSV.

* Extract Specific information :
* Response Variables - Use Postman's "Tests" tab to write JavaScript code that extracts specific data from the response and assigns it to variables. For example, you can use **pm.environment.set()** or **pm.variables.set()** to store values that you want to reuse or reference in subsequent requests.
* Response Scripts - Utilize the "Tests" tab to write custom JavaScript code that parses and extracts the desired information from the response body. You can access the response body using the **responseBody** variable and apply parsing logic to extract specific data.
* Format the Data - Once we have extracted the necessary information, we can format it to suit our requirements. Postman supports different output formats such as JSON and CSV, among others. We can format the extracted data using JavaScript code and store it in variables or collections for further processing.
* JSON Formatting: Use JavaScript functions such as **JSON.stringify()** to format the extracted data into a well-structured JSON format. You can assign the formatted JSON to a variable or save it as a file.
* CSV Formatting: If you prefer a CSV format, you can use JavaScript code to concatenate the extracted data into a comma-separated format. Create a variable or save it as a file with a .csv extension.
* Store the Structured Output: Postman provides several options for storing the structured output:
* Environment Variables: Save the formatted data as environment variables using **pm.environment.set()** or **pm.variables.set()**. This allows you to reuse the data across different requests in the same collection run.
* Global Variables: Similar to environment variables, you can use **pm.globals.set()** to store the formatted data as global variables. Global variables are accessible across different collections.
* Collection Variables: Save the structured output as collection variables using **pm.collectionVariables.set()**. Collection variables are specific to the current collection.
* The console will display the output of **console.log()** statements, including the value of the environment variables.