REST API Tutorial

A Practical Guide to Working with APIs

Learning Objectives

By the end of this session, students will be able to:

- Understand what an API is and why it is essential in modern applications.
- Explain REST APIs and how they allow communication between different systems.
- Use API documentation to understand endpoints, parameters, and responses.
- Perform API requests using Postman (Extension in VSCode) to fetch real-world data.
- Analyze JSON responses and extract useful information.
- Understand how to interact with APIs programmatically using Python scripts

Technical requirements

Before starting, ensure you have the following softwares/tools installed and set up:

- **Visual Studio Code (VSCode)** – The code editor used for this session.
- **Postman Extension** – A lightweight Postman alternative to test API requests. This can be installed from the VSCode Extensions Marketplace.
- **Python (Version 3.8+)**
- **pip (Python package manager)** – Comes with Python but ensures it's installed.

The required API keys/accounts may take time to activate. So, for this session your instructor will provide you with pre-configured keys

- **A Free GeoNames API Account**
- **A Merriam-Webster API Key** (For Part 3 – Dictionary Project)

Section 1: Basic API Queries with GeoNames

The objective of this section is to introduce you to API queries using the **GeoNames API**, a service that provides geographical data such as place names, coordinates, and administrative boundaries. You will learn how to construct API requests, interpret responses, and handle different data formats returned by the API.

1. Reading API Documentation

The tasks in this section will involve using the API to retrieve geographical data and answering basic questions based on the data retrieved from GeoNames. Understanding how to work with APIs is a fundamental skill for software developers, as APIs are widely used in industry to access and exchange data between systems.

A key part of working with APIs is learning to read and understand API documentation. In industry, you will often encounter new systems where documentation is the primary resource for understanding functionality. If you are unsure about query parameters, response formats, or how to structure your requests, refer to the **GeoNames API documentation** before seeking assistance. Developing the ability to troubleshoot and research independently is an essential skill for any software developer.

2. Understanding REST APIs and HTTP Methods

Before making API calls, ensure you understand the fundamental concepts of REST APIs (Representational State Transfer) and the different HTTP methods used for sending requests to a server. These methods define how clients communicate with APIs and retrieve or modify data.

Here's a quick reminder of the key HTTP methods used in API communication:

- **POST** **C**reates a new resource on the server.
- **GET** **R**etrieves data from the server.
- **PUT** **U**pdates an existing resource with new data.
- **DELETE** **D**eletes a resource from the server.

3. Understanding API Responses

Most REST APIs return responses in **JSON (JavaScript Object Notation)** format. JSON is a lightweight and human-readable data format used for transmitting structured data between a client and a server. It consists of key-value pairs, making it easy to parse and manipulate.

Here's an example of a typical JSON response from an API:

```
{
  "geonames": [
    {
      "name": "Sheffield",
      "countryCode": "GB",
      "lat": "53.38297",
      "lng": "-1.4659",
      "population": 563749
    }
  ]
}
```

While JSON is more commonly used due to its simplicity and efficiency, some older APIs may return data in **XML (Extensible Markup Language)** .

Example of an XML Response:

```
<geonames>
  <geoname>
    <name>Sheffield</name>
    <countryCode>GB</countryCode>
    <lat>53.38297</lat>
    <lng>-1.4659</lng>0
    <population>563749</population>
  </geoname>
</geonames>
```

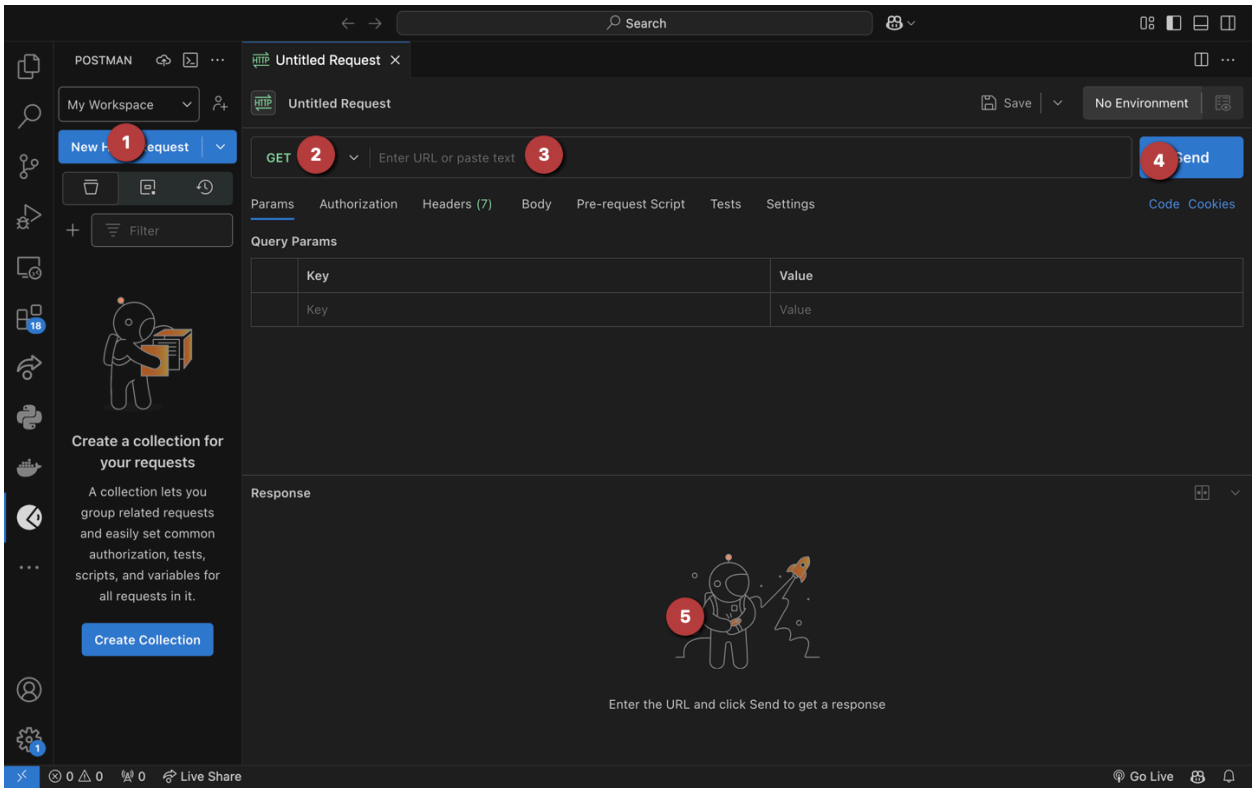
This XML response contains the same information as the JSON example but is structured differently, using tags instead of key-value pairs.

4. Postman Extension: Making API requests on Postman

What is Postman?
Postman is a popular API testing tool that allows developers to send API requests, view responses, and debug endpoints efficiently. It provides a user-friendly interface for making GET, POST, PUT, DELETE requests without writing code, making it ideal for testing APIs before integrating them into applications.

Postman is available as a standalone desktop application and a VS Code extension, which allows you to make API calls directly within your VS Code environment.

Making an API Request on Postman



Open the postman extension in VsCode

- **1.** Click the **New HTTP Request** button to open a new request window
- **2.** Select an HTTP method (e.g., GET, POST, etc).
- **3.** Enter the API endpoint (URL) in the request URL field e.g

```
https://api.geonames.org/searchJSON?q=Sheffield&username=demo
```

- **4.** Click **Send** to execute the request.
- **5.** View the **API response** in the output section.

Adding Query Parameters & Headers

- To add query parameters, click on the **Params** tab and enter key-value pairs.
- To include headers (e.g., API keys, content type), navigate to the **Headers** tab.
- If the API requires authentication, go to the **Authorization** tab and select the authentication type (e.g., API Key, OAuth).

Task 1: Your First API Call

Make a GET request to retrieve information about a city (Shibuya) using the GeoNames API and answer the following questions.

- What latitude and longitude does the API return for the city?
- What is the country of the city?

Endpoint

```
http://api.geonames.org/searchJSON?q=Shibuya&maxRows=1&username=<your_username>
```

Parameters

- **q** Name of the city to search for
- **maxRows** Number of results to return (set to 1 for a single result)
- **username** Your GeoNames username (required for authentication)

NOTE: REPLACE the username parameter with the account you have been provided.

Task 2: City Population Challenge

Find and compare the population of **three capital cities (London, Tokyo, and Washington)**.

- What parameters are needed to obtain this information? (**Hint:** Check the documentation!)
- Which city has the highest population?
- Which city has the highest elevation?

Endpoints

```
http://api.geonames.org/searchJSON
http://api.geonames.org/srtm3JSON
```

Task 3: Weather Station Detective

Find the **nearest weather station** to your location and extract key weather data.

- Which API endpoint do you need for this task? (**Hint:** Check the documentation!)

- What parameters do you need to include in your request?
- What is the **temperature**, **wind speed**, **wind direction & humidity** at the nearest weather station?

Section 2: Dictionary App (Optional)

This session will **not include specific tasks.** The goal is to run the Dictionary App and explore how the **Requests** library is used to interact with an external API programmatically using Python Code. The focus is on understanding the **Requests** library, an HTTP client library for the Python programming language.

By the end of this section, you should be able to:

- Understand how to use virtual environments to manage program dependencies.
- Use the requests library to send API requests.
- Understand how to handle different types of HTTP requests
- Parse and process API responses in Python.

1. Setting Up a Virtual Environment

Before using the **Requests** library, it is recommended to set up a virtual environment. A virtual environment helps manage dependencies separately from system-wide Python packages, preventing conflicts between different projects.

Complete the following steps to set up and activate a virtual environment.

1. Open a terminal and navigate to the **/01_introduction_to_rest_apis** folder.

2. Run the following command to create a virtual environment named **(env)**:

```
python -m venv env
```

3. Activate the virtual environment by running **one** of the following commands depending on your operating system:

On Windows:

```
env\Scripts\activate
```

On macOS/Linux:

```
source env/bin/activate
```

Once activated, your terminal should show **(env)** before the command prompt, indicating that you are now working inside the virtual environment.

2. Installing the Requests Library

pip (Python Package Installer) is the default package manager for Python. It allows you to install, update, and manage third-party libraries from the **Python Package Index (PyPI)**.

PyPI is an online repository that hosts thousands of Python packages, including requests. It allows developers to easily install and manage dependencies using pip.

PyPI Documentation for Requests: https://pypi.org/project/requests/

Once the virtual environment is activated, install the requests library using:

```
pip install requests
```

This downloads and installs the package from PyPI into your virtual environment.

Once complete, verify the installation by running:

```
pip list | grep requests
```

3. Run the Dictionary App

The code for the dictionary application is located in **dictionary.py**. This program interacts with the **Merriam-Webster API**, which provides access to a vast collection of **over 225,000 clear and precise definitions**.

Instead of manually compiling and maintaining a dictionary database, the API allows instant access to **real-time definitions**, demonstrating the power of APIs in software development.

Before running the program,

1. Ensure your virtual environment is activated.
2. Enter your API Key at **line 14** (your instructor should have provided this)

Run the following command to start the program:

```
python dictionary.py
```