# API Basics

## API (Application Programming Interface) and Client-Server Communication

An API (Application Programming Interface) is a mechanism that allows two software components to communicate with each other using a set of definitions and protocols. APIs enable interaction between a client (e.g., a web browser or mobile app) and a server (e.g., a backend system or database).

For example, when you use a weather app to check the forecast, the app requests data from a weather API, which then returns the information.

## What is JSON?

JSON (JavaScript Object Notation) is an open-standard file format used for data exchange between a client and a server. It represents data in a human-readable key-value pair structure, making it easy to store and transmit information.

Example of JSON Data:

{  
 "name": "John Doe",  
 "age": 25,  
 "email": "johndoe@example.com",  
 "skills": ["Python", "JavaScript", "SQL"]  
}

## GET vs POST Methods

GET and POST are two common HTTP request methods used in APIs to send data between a client and a server.

### What is GET?

The GET method is used to retrieve data from a server using a request-URI (Uniform Resource Identifier).

Key Features of GET:

1. Retrieves information (e.g., fetching user details, search results).

2. GET requests are stored in browser history and can be bookmarked.

3. The data sent with a GET request is visible in the URL.

4. Limited to 255 characters in URL parameters.

5. Generally faster and more efficient than POST.

6. Supports only string data types.

Example of a GET Request:

https://example.com/search?query=laptops

### What is POST?

The POST method is used to send data to the server to create a new resource (e.g., user registration, submitting a form).

Key Features of POST:

1. Sends data to the server for processing (e.g., adding new users).

2. POST requests are not stored in browser history.

3. Data is sent in the request body, making it more secure.

4. No size limitation (unlike GET).

5. Supports various data types (string, numeric, binary, etc.).

6. Slightly slower compared to GET because it processes more data.

Example of a POST Request (User Registration):

POST /register HTTP/1.1  
Host: example.com  
Content-Type: application/json  
  
{  
 "username": "john\_doe",  
 "password": "securepassword",  
 "email": "john@example.com"  
}

## PUT vs DELETE Methods

### What is PUT?

The PUT method is used to update an existing resource on the server.

Example of a PUT Request (Updating a User's Email):

PUT /users/123 HTTP/1.1  
Host: example.com  
Content-Type: application/json  
  
{  
 "email": "newemail@example.com"  
}

### What is DELETE?

The DELETE method is used to remove a resource from the server.

Example of a DELETE Request (Deleting a User):

DELETE /users/123 HTTP/1.1  
Host: example.com

## Summary Table: GET vs POST vs PUT vs DELETE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Purpose | Data Sent? | Visible in URL? | Cached? | Used for |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| GET | Retrieve data | Yes (in URL) | Yes | Yes | Fetching data |
| POST | Create a new resource | Yes (in body) | No | No | Form submission, registration |
| PUT | Update an existing resource | Yes (in body) | No | No | Updating records |
| DELETE | Remove a resource | No | No | No | Deleting records |