1. Postman KickStart

Introduction to Postman

1. Postman is an API testing tool.
2. We can do manual testing of API's using Postman
3. However, basics of Automation and JavaScript knowledge is needed for tasks like validating responses, Datadriven Testing, Running requests multiple times as Postman supports JavaScript by default.
4. Developers also use Postman to create API documentation.
5. Postman supports testing of both REST APIs and SOAP APIs.
6. It is available in both **web** and **desktop versions**, with similar features but different access methods.

Download and Installation of Postman Client Tool

➔ Refer **Postman Client Installation,Workspace&Collections** Document.

Terminologies

1. **Workspace:** A workspace in Postman is a collaborative environment where you can organize and manage your API projects, including collections, environments, and team members.
    a. we can **create, rename , delete workspaces**
    b. **Visibility of workspace -** Determines who can access workspace.
        i. **Internal -** Build and Test API's within our Team
        ii. **Partner -** Share APIs securely with trusted partners
        iii. **Public -** Make APIs accessible to the world
2. **Collection:** A collection in Postman is a group of related API requests organized together in folders for better management and reusability.
    a. We can **create, rename, delete, and run collections in Postman**.
    b. Multiple collections can be created under a single workspace.
    c. Once we create a collection we will be able to create a Request.
3. **Request:** A request in Postman is an API call that contains details like the method (GET, POST, etc.), endpoint URL, headers, and body to interact with a specific API.
    a. To Test API we need to send the Request and we will get the Response.
        i. **Request → API → Response**
        ii. **Http Requests**
            1. **Get** → retrieve the resource from database
            2. **Post** → create resource on database

3. **Put** → update existing resource on database
4. **Patch** →update partial details of resource
5. **Delete** → delete existing resource from database

4. We will do lot of validations on Response like **status code, time, size data, response body (json/xml), cookies, headers**.
5. Based on status code we can say whether the Request is successful or not.

Sample HTTP Requests

➔ To Demonstrate initially we will use Free and Open source Apis's → https://reqres.in/

TCID	Request Type	URI	Request Payload	Response Payload	Status Code
1	**GET**	https://reqres.in/api/users?page=2	NA	Returns list of users in a page	200

2	**POST**	https://reqres.in/api/users	{
			"name": "madhan",
			"job": "trainer"
			}
				{
				"name": "madhan",
				"job": "trainer",
				"id": "76",
				"createdAt":
				"2024-12-23T15:31:15.325Z"
				}
					201

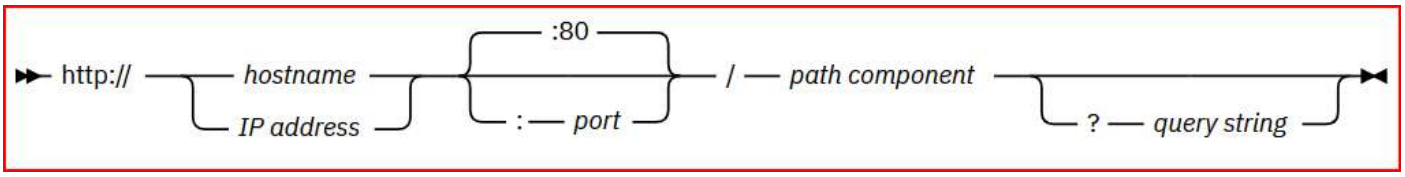
3	**PUT**	https://reqres.in/api/users/76	{
			"name": "madhan",
			"job": "engineer"
			}
				{
				"name": "madhan",
				"job": "engineer",
				"updatedAt":
				"2022-07-18T02:04:09.462Z"
				}
					200

4	**DELETE**	https://reqres.in/api/users/76	NA	NA	204

➔ Validations on Response are not done in browser will use postman tool.

Understanding HTTP/HTTPS URL Components

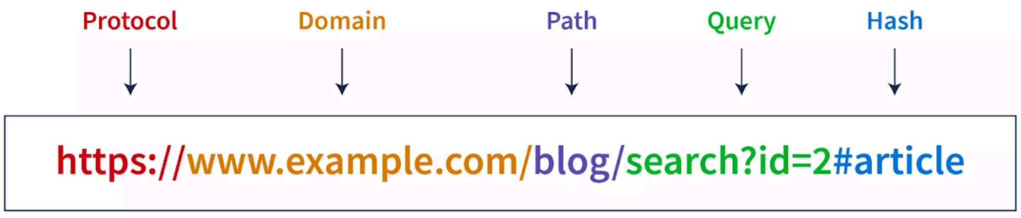
1. A URL (Uniform Resource Locator) identifies a resource's location on the internet, while a URI (Uniform Resource Identifier) is any identifier for a resource.
2. URLs are a type of URI that specifies location or access methods.



3. **HTTP/HTTPS URL Components**
   a. **Scheme:** Specifies the protocol (e.g., https or http).
   b. **Hostname:** Identifies the server (e.g., www.example.com).
   c. **Port:** Optional; follows the hostname with a colon (e.g., :80).
   d. **Path:** Points to the specific resource (e.g., /software/index.html).
   e. **Query String or Query Params:** Optional; starts with ? and contains parameters (e.g., term=bluebird).

4. A fragment identifier which is optional, added after `#` in a URL, points to a specific section or function within the retrieved resource, like a subsection in an HTML page, and its behavior depends on the media type.

https://www.ibm.com/docs/en/cics-ts/6.x?topic=concepts-components-url



5. Scheme and Hostname are case-insensitive, but path and query strings are case-sensitive.

Note

➔ We gather all API information from the development team, who provide Swagger documentation. Based on this documentation, we create test cases.

➔ Based on how the REST API is designed during development, choose the appropriate format (JSON, HTML, text, etc.) for the payload according to the API's request and response type.

Workout in Postman Tool without validations initially

http request = url+ http method+ req body+Autorization

1. Create GET, POST, PUT, DELETE requests under "**Day1_HTTPRequestsDemo**" in **myworkspace** by clicking "Add Request."
2. Save the request before sending it to view the response payload.
3. Observe response payload once we send the request in postman tool
4. POST requests return a payload with an `id`.
5. Use PUT requests with the URL and `id` from the POST response to update records.
6. In the "Body" section, select "raw" and "JSON" for POST and PUT requests.

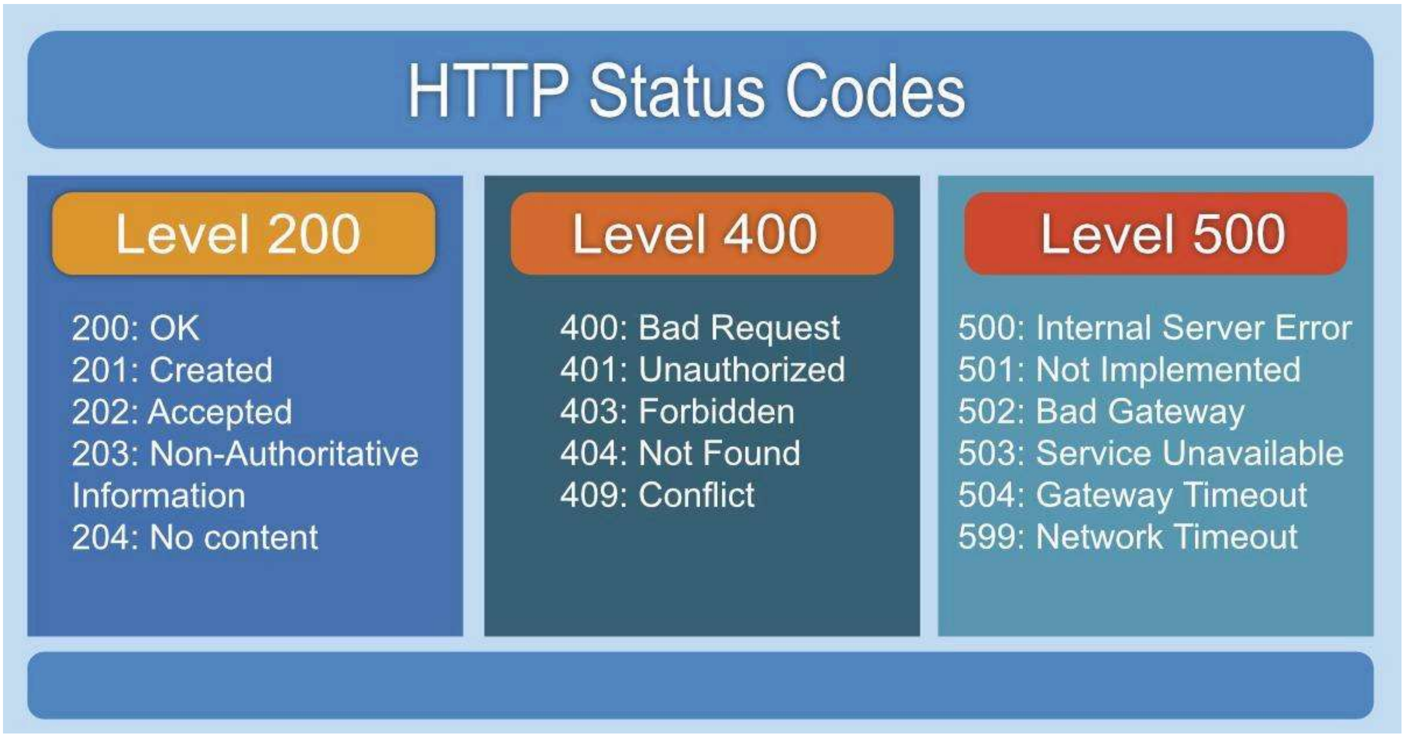
Note

➔ **Autosave** feature is also available in Postman settings under "**General → Application**".

➔ In a real-time environment, we can also validate responses with the data in the database also.

Understanding Status Codes

1. Based on status Code we can check whether the Request is successful or not.
2. They are divided into 3 Levels/Series.
    a. 200 series status codes represent a successful request.
    b. 400 series status codes represent unauthorized access to APIs. Some APIs require authentication, and if not provided, the request won't be processed.
    c. 500 series status codes indicate server issues, such as being down or under maintenance.



HTTP Status Codes

Level 200	Level 400	Level 500
200: OK	400: Bad Request	500: Internal Server Error
201: Created	401: Unauthorized	501: Not Implemented
202: Accepted	403: Forbidden	502: Bad Gateway
203: Non-Authoritative Information	404: Not Found	503: Service Unavailable
204: No content	409: Conflict	504: Gateway Timeout
		599: Network Timeout

Note

- ➔ If we want to execute all Requests in collection at a time we can use Run collection option.
- ➔ We can Export and Import Collections also.
- ➔ Response cannot be saved since it is dynamic but we can save the request.