**Header Types for options**

**RAW**

Json - application/Json

xml - application/xml

JavaScript - application/JavaScript

text - text/plain

Html - text/html

**Form data**

header type in form data

( can supply multiple text and multiple file)

header type = multipart/form-data

**x-www-form-URL encoded**

( can supply multiple text but file upload is supported/not available)

content type -application/x-www-form-URL encoded

**Binary**

only file can be uploaded. Only 1 file per request (limitation)

file name is not sent to the server

**STATELESS /STATEFULL**

**Stateless**- REST API (state means session. sessions are independent from each other). Server will not hold any kind of client information, like session id, cookie etc. In Rest API, POST, GET etc calls are independent of each other. Client supply id with every request and hence it is slower than stateful

Selenium

Script+ SID (chrome = new chrome driver) ----- Server

Req2 get ()+ sid ----- ser

**Stateful**- WebSocket (playwright), chat application. Server is maintaining the session id. Client do not supply ID with each request. It is faster. Maintenance is one of the biggest problems with stateful. Additional session storage is required.

Client (Req 1) --- Server (sessionID)

Req 2 ----- Server(SID)

Query Parameter – used to filter the data

{{BaseURL}}/public/v2/users?page=1&per\_page=30

{{BaseURL}}/public/v2/users?gender=female&status=active&per\_page=100

Query parameters are appended with the URL/Request using ?

Path Parameter:

/ is used for path parameter

: is used as shortcut in postman for path variable

|  |  |
| --- | --- |
| **Stateless Examples**: | **Stateless Example**:   1. HTTP: The Hypertext Transfer Protocol (HTTP) is a stateless protocol widely used for communication between web browsers and web servers. Each request sent by the browser contains all the necessary information for the server to process it, such as the HTTP method, headers, and payload. The server does not retain any knowledge of past requests from the same client. Each request is treated independently. 2. RESTful APIs: Representational State Transfer (REST) is an architectural style for building web services. RESTful APIs follow the stateless constraint of HTTP, where each request from a client contains all the necessary information for the server to process it. The server does not store any session or context between requests. Authentication is typically handled through tokens or credentials provided with each request. |
| **Stateful Examples:** | **Stateful Example:**   1. TCP: The Transmission Control Protocol (TCP) is a stateful protocol used for reliable and ordered communication between networked devices. TCP establishes a connection between the client and server, creating a session. The connection is maintained until explicitly closed. TCP keeps track of the current state of the connection, including sequence numbers, acknowledgments, and window sizes, to ensure reliable delivery of data. 2. Session-based Authentication: Many web applications use session-based authentication, where a user's session is established upon successful login. The server assigns a unique session ID to the client and stores session data on the server-side. The session ID is typically stored in a cookie or included in each request. The server references the session data to authenticate subsequent requests and maintain user state throughout the session. 3. WebSocket is considered a stateful protocol. Unlike traditional HTTP, which is stateless, WebSocket maintains a persistent, bidirectional connection between the client and the server. This connection is established through an initial HTTP handshake and then upgraded to the WebSocket protocol. |

VERTICAL SCALING : Add up more and more space to the server as per the need (like 16 Gb, 32 GB, 64 GB ….) but with one server only. This is old mechanism

HORIZONTAL SCALING – Add up more and more server to mee the demand. Maintenance cost is high in this. This generally take help of any load balancer to balance the load. Now a days this is preferred. Stateless is preferred as server do not have to maintain any session ID and multiple request from same client can hit any of the servers

**SCOPE OF POSTMAN VARIABLES**

L - Local

D - Data

E – Environment

C – Collections – within a specific collection. Applicable for all the request

G – Global - Rare but used when same variable is used across multiple collection

**URI vs URL vs URN:**

* **URI (Uniform Resource Identifier)**: A general term that encompasses both URLs and URNs, as well as other schemes that identify resources.
* **URL (Uniform Resource Locator)**: A specific type of URI that provides a means to locate the resource. All URLs are URIs.
* **URN (Uniform Resource Name/Number)**: Another specific type of URI that provides a unique name for a resource within a namespace but does not provide location information.

**All URNs are URIs.**

|  |  |  |
| --- | --- | --- |
| **Type** | **Example** | **Description** |
| **URI (General)** | <https://www.naveenautomation.com:8080/path/to/resource?query=example#fragment> | A general URI that identifies and locates a resource. |
| **URL (Uniform Resource Locator)** | <https://www.naveenautomation.com:8080/path/to/resource?query=example#fragment> | A type of URI that provides a means of locating the resource. |
| **URN (Uniform Resource Name)** | urn:isbn:0451450523 | A type of URI that identifies a resource by name without specifying its location. |
| **URI with URL** | <https://www.naveenautomation.com:8080/path/to/resource?query=example#fragment> | A URI that is also a URL, providing both identification and location. |
| **URI with URN** | urn:isbn:0451450523 | A URI that is also a URN, providing identification by name within a namespace. |
| **URI without URL** | <mailto:>example@naveenautomation.com?subject=Hello%20World | A URI that does not provide location information, used for email addresses. |
| **URL without URN** | <https://www.naveenautomation.com/page.html> | A URL that does not include a URN, only provides location information. |
| **URN without URL** | urn:isbn:0451450523 | A URN that does not include a URL or location information. |
| **URI without URN** | <mailto:>example@naveenautomation.com?subject=Hello%20World | A URI that does not include a URN, used for email addresses. |
| **FTP URL** | <ftp://ftp.naveenautomation.com/file.txt> | A URL using the FTP protocol to locate a file on an FTP server. |
| **SMTP URI** | smtp://mail.naveenautomation.com | A URI using the SMTP protocol for email servers, without specifying a specific email address. |
| **FTP URI** | <ftp://ftp.naveenautomation.com/file.txt> | A URI that uses the FTP scheme to identify a resource on an FTP server. |
| **SMTP URI with Email** | smtp://mail.naveenautomation.com?to=example@naveenautomation.com&subject=Hello | A URI using SMTP scheme with parameters for an email client. |
| **FTP URI without URL** | <ftp://ftp.naveenautomation.com/file.txt> | A URI that uses FTP to identify a resource but is also a URL since it includes location information. |
| **SMTP URI without URL** | smtp://mail.naveenautomation.com | A URI using the SMTP scheme, identifying a mail server without providing full details of email actions. |
| **MAILTO URI** | <mailto:>example@naveenautomation.com?subject=Hello%20World | A URI identifying an email address with optional parameters for subject and body. |
| **URI with URL (File)** | file:///C:/path/to/file.txt | A URI using the file scheme to access a local file system resource. |
| **URL with File URI** | file:///C:/path/to/file.txt | A URL that specifies a local file path. |
| **FTP URL without URN** | <ftp://ftp.naveenautomation.com/file.txt> | A URL using FTP to locate a file, does not include a URN. |
| **MAILTO URI without URL** | <mailto:>example@naveenautomation.com |  |



BASIC AUTH – Simplest/basic form of authentication. We need to supply username & password while any request (GET, PUT, POST). API will automaticaly convert it into base 64 encodedstring .

JS method to get encoded string : btoa(“username:password”)

To get decoded string atob(“username:password”)

Check this in browser console