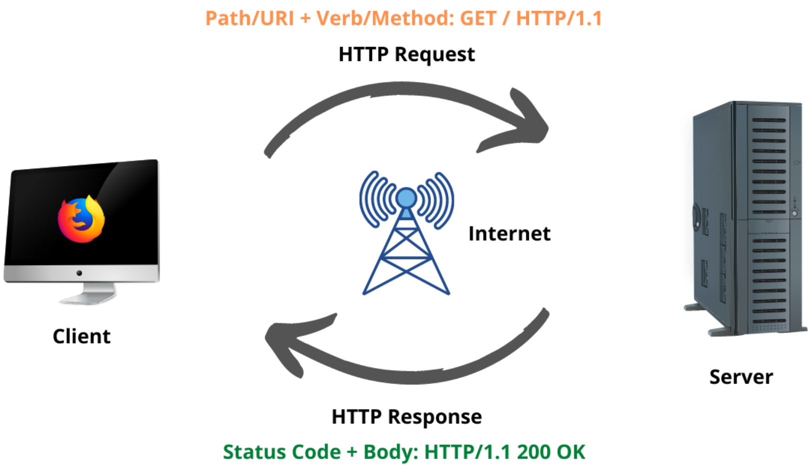
# Request Response Model Best Practices

In the request-response model, a client computer or software requests data or services, and a server computer or software responds to the request by providing the data or service. For example, when we send a spreadsheet to the printer, your spreadsheet program is the client.

### Request-Response:

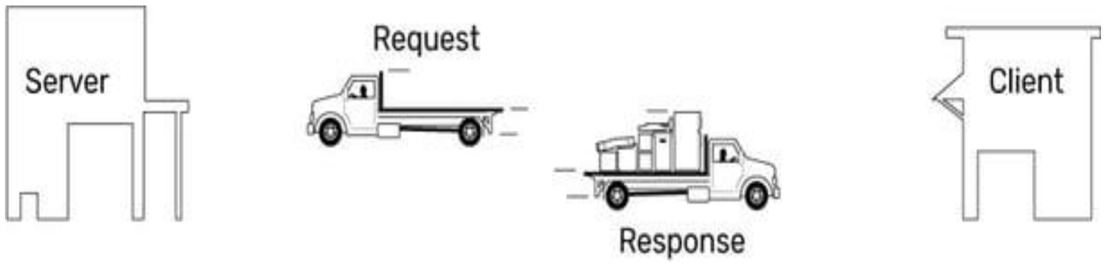
- ✓ Answer every request with a response message

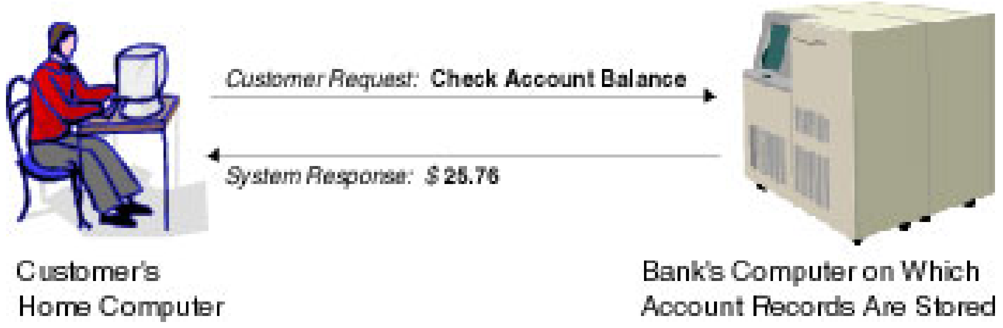

Path/URI + Verb/Method: GET / HTTP/1.1
HTTP Request
Internet
Client
Server
HTTP Response
Status Code + Body: HTTP/1.1 200 OK

- ✓ Browser is a primary HTTP client.

## HTTP/HTTPS Request Response Communication__

- ✓ In request/response communication mode.
- ✓ One software module sends a request to a second software module and waits for a response.
- ✓ The first software module performs the role of the client.
- ✓ The second module acts the role of the server.





## HTTP Request

HTTP request is the first step to initiate web request/response communication. Every request is a combination of request header, body and request URL.

**Request Method:**

The request **method** indicates the method to be performed on the resource identified by the given **Request-URI**. The method is case-sensitive and should always be mentioned in uppercase. The following table lists all the supported methods in HTTP/1.1.

| S.N. | Method and Description |
|---|---|
| 1 | **GET**<br>The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| 2 | **HEAD**<br>Same as GET, but it transfers the status line and the header section only. |
| 3 | **POST**<br>A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms. |
| 4 | **PUT**<br>Replaces all the current representations of the target resource with the uploaded content. |
| 5 | **DELETE**<br>Removes all the current representations of the target resource given by URI. |
| 6 | **CONNECT**<br>Establishes a tunnel to the server identified by a given URI. |

| | |
|---|---|
| 7 | **OPTIONS**<br>Describe the communication options for the target resource. |
| 8 | **TRACE**<br>Performs a message loop back test along with the path to the target resource. |

**Start line**

HTTP requests are messages sent by the client to initiate an action on the server. Their start-line contain three elements:

1. An HTTP method, a verb (like GET, PUT or POST) Or a noun (like HEAD or OPTIONS), that describes the action to be performed. For example, GET indicates that a resource should be fetched or POST means that data is pushed to the server (creating or modifying a resource, or generating a temporary document to send back).

2. The request target, usually a URL, or the absolute path of the protocol, port, and domain are usually characterized by the request context. The format of this request target varies between different HTTP methods. It can be
   - ✓ An absolute path, ultimately followed by a '?' and query string. This is the most common form, known as the origin form, and is used with GET, POST, HEAD, and OPTIONS methods.
     - POST / HTTP/1.1
     - GET /background.png HTTP/1.0
     - HEAD /test.html?query=alibaba HTTP/1.1
     - OPTIONS /anypage.html HTTP/1.0
   - ✓ A complete URL, known as the absolute form, is mostly used with GET when connected to a proxy. GET https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages HTTP/1.1
   - ✓ The authority component of a URL, consisting of the domain name and optionally the port (prefixed by a ':'), is called the authority form. It is only used with CONNECT when setting up an HTTP tunnel. CONNECT developer.mozilla.org:80 HTTP/1.1

- ✓ The asterisk form, a simple asterisk ('*') is used with OPTIONS, representing the server as a whole. OPTIONS * HTTP/1.1

3. The HTTP version, which defines the structure of the remaining message, acting as an indicator of the expected version to use for the response.

**HTTP Request Segment:**

| **Request Area** | **Standard Data Type** |
|---|---|
| Body | Simple String, JSON, Download, Redirect, XML |
| Header | Key Pair Value |
| URL parameter | String |

**HTTP Response:**

HTTP response is the final step of request-response communication. Every response is a combination of response header, body and cookies.

**Status line**

The start line of an HTTP response, called the status line, contains the following information:

4. The protocol version, usually HTTP/1.1.
5. A status code, indicating success or failure of the request. Common status codes are 200, 404, or 302
6. A status texts. A brief, purely informational, textual description of the status code to help a human understand the HTTP message.

A typical status line looks like: HTTP/1.1 404 Not Found.

**HTTP Response Segment:**

| Response Area | Standard Data Type |
|---|---|
| Body | Simple String, JSON, Download Redirect, XML |
| Header | Key Pair Value |
| Cookies | Key Pair Value |

**Benefits of Request-Response Model:**

- Every request gets answered.
- Timeout can be detected.
- Windowing and buffered response possible.
- Two decoupled communication events: Question and Answer

**Drawbacks:**

- Continuous data stream not possible.
- Broadcast/Multicast not possible.
- Asynchronous communication is more difficult to debug.

**POSTMAN HTTP Client**

Postman is an HTTP Client application, used to test request-response communication.

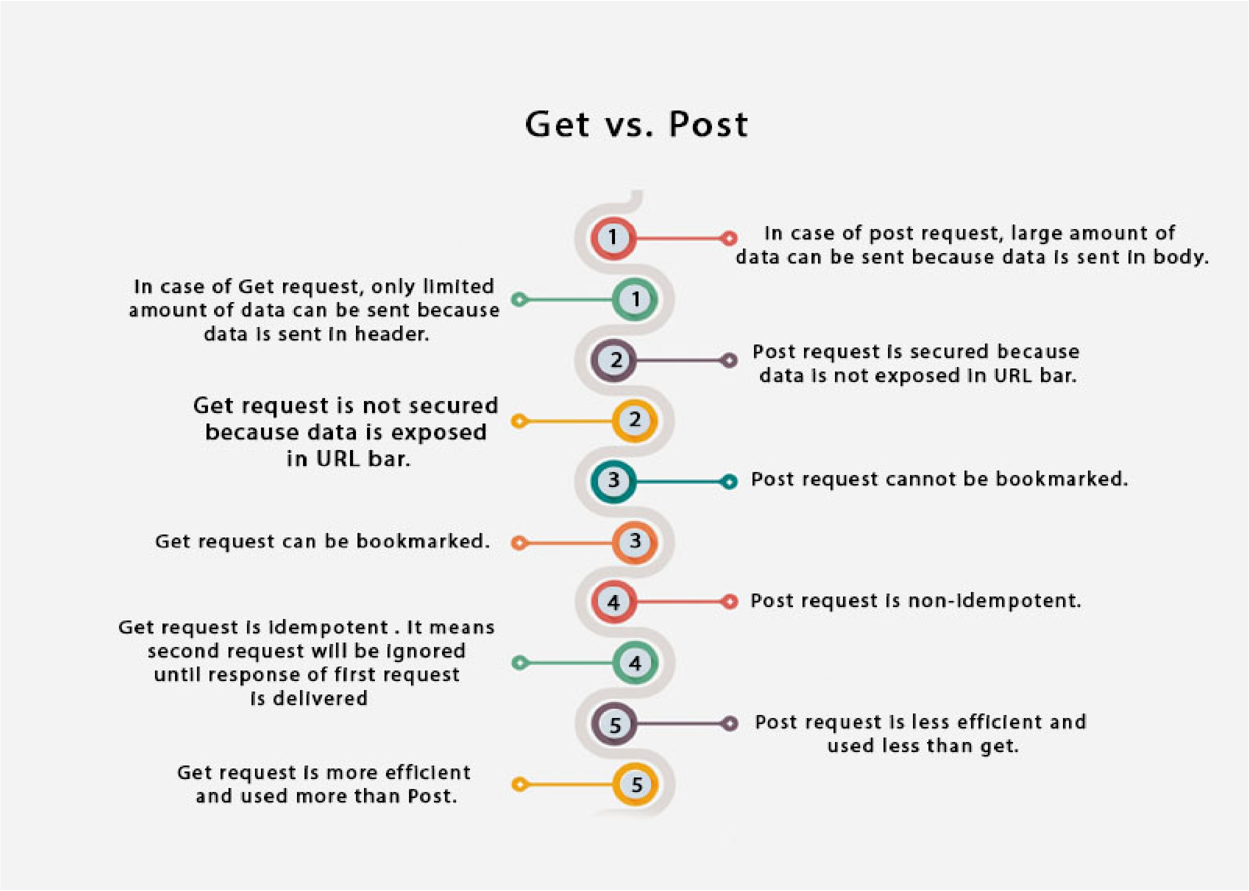
Postman is widely used for API testing and generating documentation.

- ✓ Quickly and easily send REST, SOAP and GraphQL request directly within Postman.
- ✓ Generate and publish beautiful, machine-readable API documentation.
- ✓ Checking performance and response times at scheduled intervals.

- ✓ Communicate the expected behavior of an API by simulating endpoints and their responses.

**GET vs POST method:**

| GET | POST |
|---|---|
| 1) In case of Get request, only **limited amount of data** can be sent because data is sent in header. | In case of post request, **large amount of data** can be sent because data is sent in body. |
| 2) Get request is **not secured** because data is exposed in URL bar. | Post request is **secured** because data is not exposed in URL bar. |
| 3) Get request **can be bookmarked.** | Post request **cannot be bookmarked.** |
| 4) Get request is **idempotent.** It means second request will be ignored until response of first request is delivered | Post request is **non-idempotent.** |
| 5) Get request is **more efficient** and used more than Post. | Post request is **less efficient** and used less than get. |



## Get vs. Post

1 In case of post request, large amount of data can be sent because data is sent in body.

1 In case of Get request, only limited amount of data can be sent because data is sent in header.

2 Post request is secured because data is not exposed in URL bar.

2 Get request is not secured because data is exposed in URL bar.

3 Post request cannot be bookmarked.

3 Get request can be bookmarked.

4 Post request is non-idempotent.

4 Get request is idempotent . It means second request will be ignored until response of first request is delivered

5 Post request is less efficient and used less than get.

5 Get request is more efficient and used more than Post.

**HTTP Request Throttling:**

Throttle Request refers to a process in which a user is allowed to hit the application maximum time in per second or per minute. Throttling is also known as request rate limiting.

- ✓ Essential component of Internet security, as DoS attacks can tank a server with unlimited requests.
- ✓ Rate limiting also helps make your API scalable by avoid unexpected spikes in traffic, causing severe lag time.

**Response Status:**

The Status-Code element in a server response, is a 3-digit integer where the first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are 5 values for the first digit:

| S.N. | Code and Description |
|---|---|
| 1 | **1xx: Informational**<br>It means the request has been received and the process is continuing. |
| 2 | **2xx: Success**<br>It means the action was successfully received, understood, and accepted. |
| 3 | **3xx: Redirection**<br>It means further action must be taken in order to complete the request. |

| | |
|---|---|
| 4 | **4xx: Client Error**<br>It means the request contains incorrect syntax or cannot be fulfilled. |
| 5 | **5xx: Server Error**<br>It means the server failed to fulfill an apparently valid request. |

HTTP status codes are extensible and HTTP applications are not required to understand the meaning of all the registered status codes. Given below is a list of all the status codes.

**1xx: Information**

| Message | Description |
|---|---|
| 100 Continue | Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request. |
| 101 Switching Protocols | The server switches protocol. |

**2xx: Successful**

| Message | Description |
|---|---|
| 200 OK | The request is OK. |
| 201 Created | The request is complete, and a new resource is created. |
| 202 Accepted | The request is accepted for processing, but the processing is not complete. |

| | |
|---|---|
| 203 Non-authoritative Information | The information in the entity header is from a local or third-party copy, not from the original server. |
| 204 No Content | A status code and a header are given in the response, but there is no entity-body in the reply. |
| 205 Reset Content | The browser should clear the form used for this transaction for additional input. |
| 206 Partial Content | The server is returning partial data of the size requested. Used in response to a request specifying a *Range* header. The server must specify the range included in the response with the *Content-Range* header. |

**3xx: Redirection**

| **Message** | **Description** |
|---|---|
| 300 Multiple Choices | A link lists. The user can select a link and go to that location. Maximum five addresses. |
| 301 Moved Permanently | The requested page has moved to a new URL. |
| 302 Found | The requested page has moved temporarily to a new URL. |
| 303 See Other | The requested page can be found under a different URL. |

| | |
|---|---|
| 304 Not Modified | This is the response code to an *If-Modified-Since* or *If-None-Match* header, where the URL has not been modified since the specified date. |
| 305 Use Proxy | The requested URL must be accessed through the proxy mentioned in the *Location* header. |
| 306 *Unused* | This code was used in a previous version. It is no longer used, but the code is reserved. |
| 307 Temporary Redirect | The requested page has moved temporarily to a new URL. |

**4xx: Client Error**

| **Message** | **Description** |
|---|---|
| 400 Bad Request | The server did not understand the request. |
| 401 Unauthorized | The requested page needs a username and a password. |
| 402 Payment Required | *You cannot use this code yet*. |
| 403 Forbidden | Access is forbidden to the requested page. |
| 404 Not Found | The server cannot find the requested page. |
| 405 Method Not Allowed | The method specified in the request is not allowed. |

| | |
|---|---|
| 406 Not Acceptable | The server can only generate a response that is not accepted by the client. |
| 407 Proxy Authentication Required | You must authenticate with a proxy server before this request can be served. |
| 408 Request Timeout | The request took longer than the server was prepared to wait. |
| 409 Conflict | The request could not be completed because of a conflict. |
| 410 Gone | The requested page is no longer available. |
| 411 Length Required | The "Content-Length" is not defined. The server will not accept the request without it. |
| 412 Precondition Failed | The pre-condition given in the request evaluated to false by the server. |
| 413 Request Entity Too Large | The server will not accept the request, because the request entity is too large. |
| 414 Request-URL Too Long | The server will not accept the request, because the URL is too long. Occurs when you convert a "post" request to a "get" request with a long query information. |
| 415 Unsupported Media Type | The server will not accept the request, because the MediaType is not supported. |
| 416 Requested Range Not Satisfiable | The requested byte range is not available and is out of bounds. |

| | |
|---|---|
| 417 Expectation Failed | The expectation given in an Expect request-header field could not be met by this server. |

### 5xx: Server Error

| Message | Description |
|---|---|
| 500 Internal Server Error | The request was not completed. The server met an unexpected condition. |
| 501 Not Implemented | The request was not completed. The server did not support the functionality required. |
| 502 Bad Gateway | The request was not completed. The server received an invalid response from the upstream server. |
| 503 Service Unavailable | The request was not completed. The server is temporarily overloading or down. |
| 504 Gateway Timeout | The gateway has timed out. |
| 505 HTTP Version Not Supported | The server does not support the "http protocol" version. |

### Conclusion:

HTTP messages are the key in using HTTP; their structure is simple, and they are highly extensible. The HTTP/2 framing mechanism adds a new intermediate layer between the HTTP/1.x syntax and the underlying transport protocol, without fundamentally modifying it: building upon proven mechanisms.