**World Wide Web (Overview, Discovery, and Original Purpose)**

**Types of Clients**

**Internet**

**How the Web Works**

**HTTP**

**URL**

**FTP**

**HTTP Fundamentals**

**HTTP**

HTTP is an abbreviation of Hyper Text Transfer Protocol. It is an application-layer protocol used for transmitting hypermedia documents like HTML. It runs in a client-server computing model where the client requests for a specific web content and the server provides it. Moreover, it defines the procedures and rules that the client and server follows for transmitting information resources.

**HTTP Version History**

**IETF**

**HTTP Support for other functionalities**

**HTTP Resource Addressing**

**Scheme**

**Authority**

**Path**

**Query**

**HTTP Request**

**Request Line**

Request Line pecifies the method token(GET, PUT…) followed by the request URI then the HTTP Protocol that is being used.

3 PARTS

1. The HTTP version
2. Response status code that gives the result of the request
3. English reason phrase describing the status code

**HTTP Request Methods**

HTTP has a set of methods to signify the actions being done on resources. HTTP methods are also called as “HTTP verbs”. However, there is one of the methods that is noun. Each of the methods have different actions and some of them are similar.

GET

This method is used to redeem or read a representation of resources. If the method successfully retrieve, the method returns a representation in XML, JSON and an HTTP response code of 200 (OK). Otherwise, if there are errors with regards to retrieving, it will either return HTTP codes like 404 (NOT FOUND) or 400 (BAD REQUEST). GET corresponds to Read in CRUD operations. GET is idempotent, having multiple identical requests will return as a single request.

For example, a user is surfing in a web browser and the user types a domain name like “google.com”. The client browser requests a GET method to the server to retrieve the resources in that URL.

HEAD

This method is similar to GET method but without its message-body in response. This method is used to retrieve meta-information written in response headers so that the entire content is not needed to be transported.

POST

This is method is used to send data/resource to the API server. The method will request for the server to accept datum/data from the client. Upon successful appliance, it will return HTTP code 200 (OK) for updating and 201 (CREATED) for creating. Otherwise, if there are errors with regards to posting, it will return HTTP code like 204 (NO CONTENT). POST is neither idempotent nor safe, having multiple identical requests end up in two resources containing the same information.

Submitting a contact form is one of the simplest example. When a user fills out inputs in a form and hit send, The inputs will be added in the response body of post request and it will be sent to the server.

PUT

This method is used to create or update resources from the client to the API server. Moreover, the method is to URI containing the value of non-existent resource ID. Upon successful request, it will return HTTP codes like 200 (OK) for updating and 201 (CREATED) for creating. Otherwise, if there are errors with regards to understanding or implementing, it will return HTTP code like 501 (NOT IMPLEMENTED). PUT method is similar to POST but PUT method is idempotent, having multiple requests ending up the same result.

PATCH

This method is used for partial modifications to a resource. Moreover, the request only needs the changes contained to a resource. PATCH is similar to PUT but the request body contains a set of instructions that tells how the resource to be modified to produce a new version. Patch is neither idempotent nor safe but PATCH request can be idempotent.

DELETE

This method is simply to delete a resource/resources identified by a URI. Upon successful deletion of resource, it will return HTTP code like 200 (OK) with a response body. Otherwise, HTTP code like 204 (NO CONTENT) will be returned if there are no response body. If the request is for a non-existent resource, it will return HTTP code like 404 (NOT FOUND). DELETE is idempotent.

TRACE

This method is for the client to see things that are being received and the data is used for testing and diagnosing. This method is mainly used to invoke a remote, application-layer loop-back of the request message. The request reflects back the message from client to the server and returns back HTTP code like 200 (OK).

OPTIONS

This method represents a request for information about the communication options available on the request/response chain identified by the Request-URI. This method allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval.

**Safe Methods**

* Are considered safe when their request semantics are read-only. This means that there should be no change of state to the server.
* The methods considered safe are the GET, HEAD, TRACE, and OPTIONS methods.

**Idempotent Methods**

* If multiple identical requests are sent to the server, the effect will be the same as the first issuance of such request.
* The methods considered idempotent are the following:
  + PUT, DELETE, and the safe request methods.

**Cacheable Methods**

* As the term implies, request methods are termed to be cacheable if server responses to those requests are allowed to be stored for reuse.(LINK RFC 7231)
* Only the GET, HEAD, and POST methods are cacheable.

**HTTP Response Message**

**Status Line**

* Consists of the protocol version, status code and its corresponding Reason-Phrase
  + <Protocol>/<Version> <Space> <Status Code> <Space> <Reason Phrase>

**Protocol Version**

**Status Codes**

* + The status code is a 3-digit integer result code of the attempt to understand and satisfy the request (link RFC 2616).
    - Fist-digit = defines the class of response
      * 1xx: Informational
      * 2xx: Success
      * 3xx: Redirection
      * 4xx: Client error
      * 5xx: Server error
  + The reason phrase is a short textual description of the associated status code.
    - Status Code Overview
      * 100: Continue
      * 101: Switching Protocols
      * 200: OK
      * 201: Created
      * 202: Accepted
      * 203: Non-Authorative Information
      * 204: No Content
      * 205: Reset Content
      * 206: Partial Content
      * 300: Multiple Choices
      * 301: Moved Permanently
      * 302: Found
      * 303: See Other
      * 304: Not Modified
      * 305: Use Proxy
      * 307: Temporary Redirect
      * 400: Bad Request
      * 401: Unauthorized
      * 402: Payment Required
      * 403: Forbidden
      * 404: Not Found
      * 405: Method Not Allowed
      * 406: Not Acceptable
      * 407: Proxy Authentication Required
      * 408: Request Timeout
      * 409: Conflict
      * 410: Gone
      * 411: Length Required
      * 412: Precondition Failed
      * 413: Payload Too Large
      * 414:URI Too Long
      * 415: Unsupported Media Type
      * 416: Range Not Satisfiable
      * 417: Expectation Failed
      * 426: Upgrade Required
      * 500: Internal Server Error
      * 501: Not Implemented
      * 502: Bad Gateway
      * 503: Service Unavailable
      * 504: Gateway Timeout
      * 505: HTTP Version Not Supported

**Message Headers**

MESSAGE HEADERS (https://www.tutorialspoint.com/http/http\_header\_fields.htm)

1. GENERAL HEADER FIELDS - These header fields have general applicability for both request and response messages.

2. REQUEST HEADER FIELDS - These header fields have applicability only for request messages.

3. RESPONSE HEADER FIELDS - These header fields have applicability only for response messages.

4. ENTITY HEADER FIELDS - These header fields define meta information about the entity-body or, if nothing is present, about the resource identified by the request.

**HTTP Extensions**

**WebDAV**

WebDAV is an extension to HTTP that allows clients to perform remote Web content authoring operations such as copy, move, delete, and create. It is defined in RFC 4918 by a working group of the Internet Engineering Task Force. Its features include the maintenance of properties, which includes creation, removal, and querying of file information, about an author or modification date, namespace management, collections, and overwrite protection. Namespace management tackles the ability to copy and move web pages within a server’s namespace and collections deals with the creating, removing, and listing of various resources. Lastly, overwrite protection handles aspects related to locking or securing of files.

The DAV protocol enables property setting, deleting, and retrieving. The DASL (DAV Searching and Locating) protocol enables searches based on property values for locating resources on the Web.

History

It began in 1996 when Jim Whitehead partnered with the World Wide Web Consortium (W3C) to host two meetings to discuss the problem of distributed authoring on the World Wide Web because as the Web grew, it became a read-only medium for most users. Dealing with both distributed and versioning together was a lot to handle so it was decided that WebDAV would focus more on distributed authoring, which facilitates collaborative editing and file management between users located remotely from each other on the Internet.

WebDav extends the set of standard HTTP verbs and headers allowed for request methods:

COPY

This allows you to copy a resource from one URI to another.

LOCK

This puts a lock on a resource. WebDAV supports both shared and exclusive locks.

MKCOL

This creates collections or directories.

MOVE

This moves a resource from one URI to another.

PROPFIND

This retrieves properties that are stored as XML from a web resource. It can also retrieve the collection structure (directory hierarchy) of a remote system.

PROPPATCH

Changes and deletes multiple properties on a resource in a single atomic act.

UNLOCK

This removes a lock from a resource.

**Content Negotiation**

Content Negotiation is the mechanism that is used for serving different representations, like language of a document, image format, or content encoding, of a resource at the same URI, so that the user agent or software can specify which is best suited for the user. Since not all users have the same preferences of how elements should be presented, content negotiation is basically the process of selecting the best representation for a given response when there are multiple representations available. Any response that has an entity-body may be subjected to content negotiation, including error responses.

Server-driven Content Negotiation

Also known as proactive content negotiation, the browser or any user-agent sends HTTP headers that describe the preferred choice of the user along with the URL. The server uses the headers as hints then an internal server-specific algorithm chooses the best content to serve to the client. The selection is based on the available representations of the response and the contents of particular header fields in the request message, or on other information pertaining to the request.

Request Header Fields:

1. Accept - lists the MIME types of media resources that the agent is willing to process. The lists of MIME types are comma-separated, each combined with a quality factor and a parameter indicating the relative degree of preference between the different MIME types.
2. Accept-CH - lists configuration data that can be used by the server to select an appropriate response. Valid values are:
3. Device-Memory - indicates the approximate amount of device RAM.
4. DPR - indicates the client’s device pixel ratio.
5. Viewport-Width - indicates the layout viewport width in CSS pixels.
6. Width - indicates the resource width in physical pixels.
7. Accept-Charset - indicates what kinds of character encodings are understood by the user-agent.
8. Accept-CH-Lifetime – used with the Device-Memory value of the Accept-CH header and indicates the amount of time the device should choose to share the amount of device memory with the server.
9. Accept-Encoding – defines the acceptable content-encoding supported compressions. The value is a q-factor list that indicates the priority of the encoding values. The default value identity is at the lowest priority.
10. Accept-Language – used to indicate the language preference of the user. A default values is often set according to the language of the graphical interface of the user agent, but most browsers allow to set different language preferences.
11. User-Agent – identifies the browser sending the request. This may contain a space-separated list of product tokens (name followed by a ’\’ and a version number) and comments (free string delimited by parantheses).
12. Vary – sent by the web browser in its response. It indicates the list of headers that were used by the server during the server-driven content negotiation phase. The Vary header is needed in order to inform the cache of the decision criteria so that it can reproduce it, allowing the cache to be functional while preventing serving incorrect content to the user.

Agent-driven Negotiation

This negotiation allows the server to send back a page containing links to the available alternative resources when faced by an ambiguous request. This resolves the issue of the server-driven negotiation not being able to scale well and sending of headers are done on every request. In agent-driven negotiation, the user is presented the resources and choose the one to use.

Transparent Negotiation

Transparent Negotiation is a combination of both server-driven and agent-driven negotiation. When a cache is supplied with a form of the list of available representations of the response (as in agent-driven negotiation), and the dimensions of variance are completely understood by the cache, the cache becomes capable of performing server-driven negotiation on behalf of the origin server for subsequent requests on that resource. Because of this, distributing the negotiation work and removing the second request delay of agent-driven negotiation when the cache is able to correctly guess the right response is possible.

**Accept Encoding**

**Using Accept Language**

**HTML**

**History of HTML and Browser Wars**

**HTML 4.02**

**XHTML**

**XHTML 1.0**

First document type in the XHTML. XHTML documents are XML conforming which it is readily viewed, edited, and validated. XML is shorthand name for Extensible Markup Language. In this version, it is well-formedness concept which all elements must be in nested elements. The naming for elements and attribute are lower case. Non-empty elements, end tags are required. Attribute values must be quoted, terminated empty elements, style and script elements and more.

**XHTML 1.1**

**HTML**

**HTML 1.0**

HTML 1.0 first version released for HTML. There are only few people knows website creation and the language are limited. There are elements like header, paragraph, list, figures, tables, forms, literal or preformatted text and math formula to create simple html file.

**HTML 2.0**

HTML 2.0 first official standard for HTML. In RFC 1866, contains the specifications for HTML 2.0. There are some specifications for the HTML elements, like having a start-tag followed by the content then an end-tag. Some elements also do not need end-tag, like the BR tag. In the specification the attribute names and elements names are not case-sensitive but the entities are case sensitive.

**HTML 3.2**

From the W3C Recommendation, the structure of HTML documents starts with the <! DOCTYPE> then followed with html element and head element which contains title element. After the head element is the body element which contains the content. The head element contents are title element, style element, link element and meta element. In HTML 3.2, table element and form elements were supported. It also supports CSS or Cascading Style Sheet.

**HTML 4.01**

HTML 4.0 was given a code-name COUGAR. It was developed by the W3C HTML Working Group, which includes Adobe, Hewlett Packard, IBM, Microsoft, Netscape Communications Corporation, Novell, SoftQuad, and Sun Microsystems. HTML 4.0 was recommended by W3C in 1997 December and in April 1998 became official standard. The documentation was revised and it was entitled as HTML 4.01.

In this version the naming conventions were different, the elements are written in uppercase letters while the attributes name are written in lowercase letters. In the specification includes the accessibility, table model, compound documents, style sheets, scripting and printing.

**HTML 5**

HTML5 is designed for the web, both now and in the future. There are new elements included such as section element, article, main, aside, hgoup, header, footer, nav, figure and many more. There are also new attributes like download, ping, charset, autofocus and others. There are changes in other elements and attributes. And new APIs, changed APIs and extensions are included.

**HTML 5.1**

**HTML 5.2**

By the W3C Recommendation in 14 December 2017. There are not yet official specifications released.

**HTML Living Standard**

**HTML Document Components**

* Doctype

-indicates HTML document version

* HTML Elements and Tags

-Text, image, video and other content attributes is contained in a HTML Elements.

* HTML Comments

-Enables excluding some part of the code.

* HTML Frames

-Enables loading of HTML pages with another HTML page in same browser window.

**Kinds of Content**