HTTP GET/POST METHOD

What is HTTP?

HTTP is what's used whenever you view a website, developed by Tim Berners-Lee and his team between 1989-1991. HTTP is the set of rules used for communicating with web servers for the transmitting of webpage data, whether that is HTML, Images, Videos, etc.

The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack. A typical flow over HTTP involves a client machine making a request to a server, which then sends a response message.

What is HTTPS?

HTTPS is the secure version of HTTP. HTTPS data is encrypted so it not only stops people from seeing the data you are receiving and sending, but it also gives you assurances that you're talking to the correct web server and not something impersonating it.

HTTPS is encrypted to increase security of data transfer. This is particularly important when users transmit sensitive data, such as by logging into a bank account, email service, or health insurance provider.

Request and Response

When we access a website, your browser will need to make requests to a web server for assets such as HTML, Images, and download the responses. Before that, you need to tell the browser specifically how and where to access these resources, this is where URLs will help.

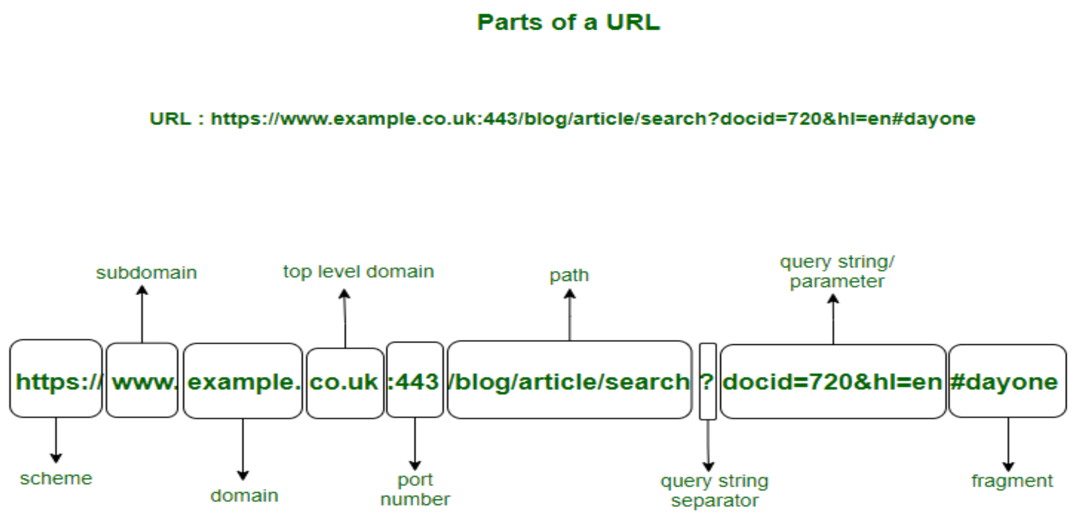
An HTTP request is the way Internet communications platforms such as web browsers ask for the information, they need to load a website.

Each HTTP request made across the Internet carries with it a series of encoded data that carries different types of information. A typical HTTP request contains:

1. HTTP version type
2. a URL
3. an HTTP method
4. HTTP request headers
5. Optional HTTP body.

What is a URL? (Uniform Resource Locator)

If you've used the internet, you've used a URL before. A URL is predominantly an instruction on how to access a resource on the internet. The below image shows what a URL looks like with all its features (it does not use all features in every request).

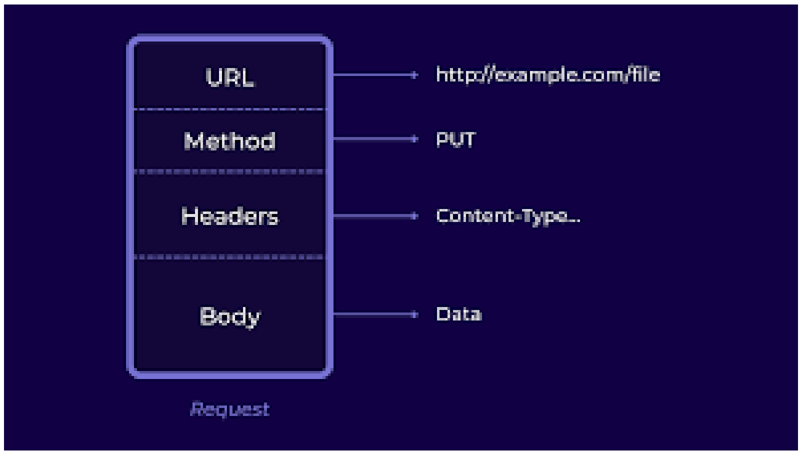


- **Scheme:** This instructs on what protocol to use for accessing resources such as HTTP, HTTPS, FTP (File Transfer Protocol).
- **User:** Some services require authentication to log in, you can put a username and password into the URL to log in.
- **Host:** The domain name or IP address of the server you wish to access.
- **Port:** The Port that you are going to connect to, usually 80 for HTTP and 443 for HTTPS, but this can be hosted on any port between 1 - 65535.
- **Path:** The file name or location of the resource you are trying to access.
- **Query String:** Extra bits of information that can be sent to the requested path. For example, /blog?**id=1** would tell the blog path that you wish to receive the blog article with the id of 1.

- **Fragment:** This is a reference to a location on the actual page requested. This is commonly used for pages with long content and can have a certain part of the page directly linked to it, so it is viewable to the user as soon as they access the page.

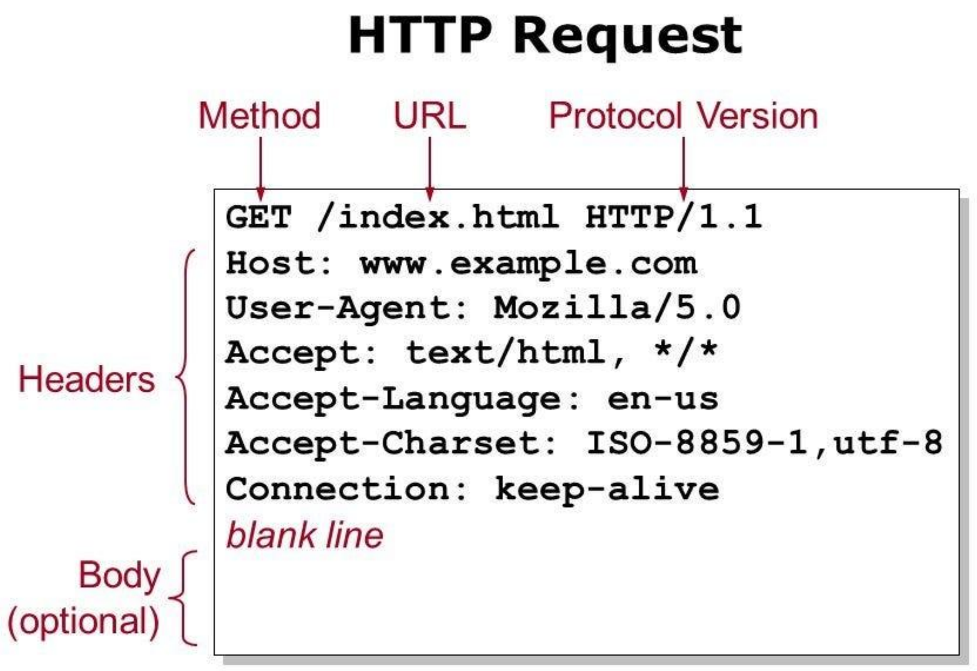
Making a HTTP Request

It's possible to make a request to a web server with just one line "**GET / HTTP/1.1**"



But for a much richer web experience, you'll need to send other data as well. This other data is sent in what is called headers, where headers contain extra information to give to the web server you're communicating with, but we'll go more into this in the Header task.

Example: Request



```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
```

Line 1: This request is sending the GET method (more on this in the HTTP Methods task), request the home page with / and telling the web server we are using HTTP protocol version 1.1.

Line 2: We tell the web server we want the website.

Line 3: We tell the web server we are using the Firefox version 5.0 Browser.

Line 4: We are telling the web server that the web page.

Line 5: HTTP requests always end with a blank line to inform the web server that the request has finished.

Example: Response

```
HTTP/1.1 200 OK
Date: Sun, 28 Aug 2017 08:56:53 GMT
Server: Apache/2.4.27 (Linux)
Last-Modified: Fri, 20 Jan 2017 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<html><body><h1>It works!</h1></body></html>
```

Line 1: HTTP 1.1 is the version of the HTTP protocol the server is using and then followed by the HTTP Status Code in this case "200 Ok" which tells us the request has been completed successfully.

Line 2: This tells us the web server software and version number.

Line 3: The current date, time, and time zone of the web server.

Line 4: The Content-Type header tells the client what sort of information is going to be sent, such as HTML, images, videos, pdf, XML.

Line 5: Content-Length tells the client how long the response is, this way we can confirm no data is missing.

Line 6: HTTP response contains a blank line to confirm the end of the HTTP response.

Lines 7-14: The information that has been requested, in this instance the homepage.

What’s an HTTP status code?

HTTP status codes are 3-digit codes most often used to indicate whether an HTTP request has been successfully completed. Status codes are broken into the following 5 blocks:

1. 1xx Informational
2. 2xx Success
3. 3xx Redirection
4. 4xx Client Error
5. 5xx Server Error

The “xx” refers to different numbers between 00 and 99.

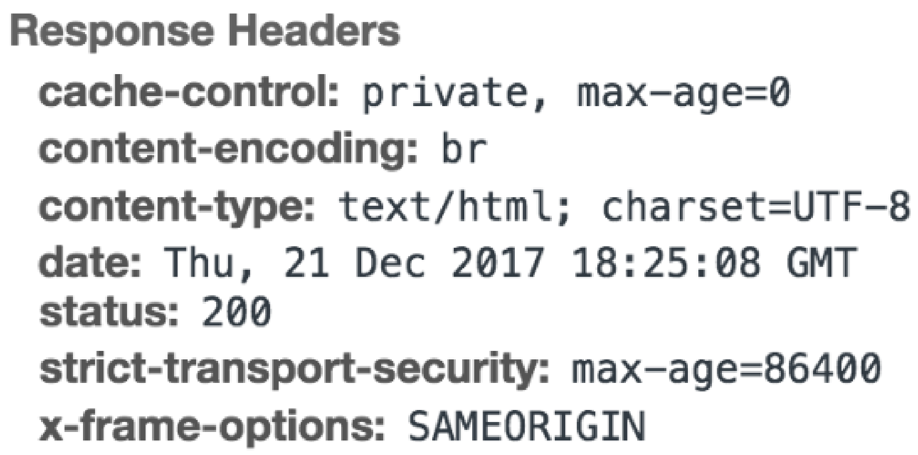
Status codes starting with the number ‘2’ indicate a success. For example, after a client requests a webpage, the most seen responses have a status code of ‘200 OK’, indicating that the request was properly completed.

If the response starts with a ‘4’ or a ‘5’ that means, there was an error, and the webpage will not be displayed. A status code that begins with a ‘4’ indicates a client-side error (it is very common to encounter a ‘404 NOT FOUND’ status code when making a typo in a URL). A status code beginning in ‘5’ means something went wrong on the server side. Status codes can also begin with a ‘1’ or a ‘3’, which indicate an informational response and a redirect, respectively.

What are HTTP response headers?

Much like an HTTP request, an HTTP response comes with headers that convey important information such as the language and format of the data being sent in the response body.

Example of HTTP response headers from Google Chrome's network tab:



```
Response Headers
  cache-control: private, max-age=0
  content-encoding: br
  content-type: text/html; charset=UTF-8
  date: Thu, 21 Dec 2017 18:25:08 GMT
  status: 200
  strict-transport-security: max-age=86400
  x-frame-options: SAMEORIGIN
```

What is in an HTTP response body?

Successful HTTP responses to ‘GET’ requests generally have a body which contains the requested information. In most web requests, this is HTML data that a web browser will translate into a webpage.

How does HTTPS work?

HTTPS uses an encryption protocol to encrypt communications. The protocol is called Transport Layer Security (TLS), although formerly it was known as Secure Sockets Layer (SSL). This protocol secures communications by using what’s known as an asymmetric public key infrastructure. This type of security system uses two different keys to encrypt communications between two parties:

1. **The private key** - this key is controlled by the owner of a website, and it’s kept, as the reader may have speculated, private. This key lives on a web server and is used to decrypt information encrypted by the public key.

2. **The public key** - this key is available to everyone who wants to interact with the server in a way that’s secure. Information that’s encrypted by the public key can only be decrypted by the private key.

Why is HTTPS important? What happens if a website doesn’t have HTTPS?

HTTPS prevents websites from having their information broadcast in a way that’s easily viewed by anyone snooping on the network. When information is sent over regular HTTP, the information is broken into packets of data that can be easily “sniffed” using free software. This makes communication over the unsecure medium, such as public Wi-Fi, highly vulnerable to interception. In fact, all communications that occur over HTTP occur in plain text, making them highly accessible to anyone with the correct tools, and vulnerable to on-path attacks.

With HTTPS, traffic is encrypted such that even if the packets are sniffed or otherwise intercepted, they will come across as nonsensical characters. Let’s look at an example:

Before Encryption:

```
This is a string of text that is completely readable
```

After Encryption:

```
ITM0IRyiEhVpa6VnKyExMiEgNveroyWBPlgGyfkflYjDaaFf/Kn3bo3OfghBPDWo6AfSHlNtL8N7ITEwIXc
```

In websites without HTTPS, it is possible for Internet service providers (ISPs) or other intermediaries to inject content into webpages without the approval of the website owner. This commonly takes the form of advertising, where an ISP looking to increase revenue injects paid advertising into the webpages of their customers. Unsurprisingly, when this occurs, the profits from the advertisements and the quality control of those advertisements are in no way shared with the website owner. HTTPS eliminates the ability of unmoderated third parties to inject advertising into web content.