PHP

VivifyIdeas

VIVIFY
ACADEMY

Overview

- HTTP
- Request
- GET
- POST
- Response

HTTP

HTTP

H - Hyper

T - Text

T - Transfer

P - Protocol

HTTP

HTTP is just a small piece in a very complex model of layers and protocols, and it has all the conventions for how hypertext gets transferred over the network.

HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response.

HTTP

- HTTP is connectionless: The HTTP client initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server processes the request and re-establishes the connection with the client to send a response back.
- HTTP is media independent: Any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content.
- HTTP is stateless: The server and client are aware of each other only during a current request. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages.

Anatomy of an HTTP request

```
GET / HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)

Host: zink.demon.co.uk:1126

Accept: image/gif, */*

Accept-Encoding: gzip

Accept-Language: en

Accept-Charset: iso-8859-1,*,utf-8

Cookie: name=xyz
```

Anatomy of an HTTP request

- GET: Request Method
- HTTP/1.0: HTTP version
- Connection: Allows the sender to specify options that are desired for that particular connection
- User-Agent: Contains information about the user agent originating the request
- Host: Used to specify the Internet host and the port number of the resource being requested
- Accept: Used to specify certain media types which are acceptable for the response

Anatomy of an HTTP request

- Accept-Encoding: Restricts the content-codings that are acceptable in the response
- Accept-Language: Restricts the set of languages that are preferred as a response to the request
- Accept-Charset: Used to indicate which character sets are acceptable for the response
- Cookie: Contains a name/value pair of information stored for that URL

Anatomy of HTTP response

```
HTTP/1.1 200 OK

Date: Fri, 04 Feb 2000 21:03:38 GMT

Server: Apache/1.3.9 (UNIX) PHP/4.0b3

Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT; path=/;
domain=tutorialspoint.com

Connection: close

Content-Type: text/html
```

HTTP Request

HTTP Request

The client sends an HTTP request to the server in the form of a request message which is disclosed in the following format:

- A Request-line
- Zero or more headers
- Optionally a message body

HTTP Request

The Request-Line begins with a method token, followed by the Request-URI and the protocol version.

Method types:

- **GET**: Used to retrieve information from the given server using a given URI.
- **HEAD**: Same as GET, but it transfers the status line and the header section only.
- **POST**: Used to send data to the server.
- **PUT**: Replaces all the current representations of the target resource with the uploaded content.

HTTP Request

- **DELETE**: Removes all the current representations of the target resource given by URI.
- **OPTIONS**: Describes the communication options for the target resource.
- **CONNECT**: Establishes a tunnel to the server identified by a given URI.
- **TRACE**: Performs a message loop back test along with the path to the target resource.

HTTP Request

Header fields allow the client to pass additional information about the request, and about the client itself, to the server.

Message body is used to send data to the server.

Chrome DevTools

Chrome DevTools is a set of authoring, debugging, and profiling tools built into Google Chrome, with which every web developer needs to be very familiar with.

Opening Chrome DevTools

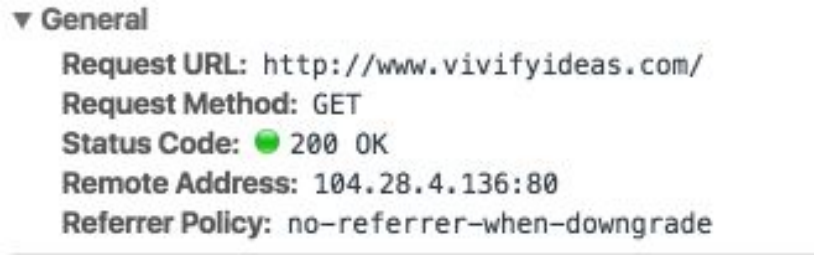
- Open any site in Chrome.
- Right mouse click on the page and click on “Inspect” menu item.

Every browser (Firefox, Edge, Safari...) has its own developer tools.

With this tool, you can inspect various behaviors on a page, for example: requests, responses, total size of data being transferred, time spent to complete the request, etc...

Chrome DevTools - Inspect a request

- Go to **Network** tab in Chrome DevTools
- Open “**All**” tab inside Network tab. If there are no items, refresh the page.
- Every item in this list is separate **HTTP request**. Click on any item in the list to inspect it.
- A **General** section like this one is visible:



- Pay attention to **Request Methods** parameter.
- Check **Request Headers** also.

GET Request

HTTP GET Request

This is the main method used for retrieving HTML, images, JavaScript, CSS, etc.

Most data that loads in your browser was requested using this method.

HTTP GET Request

- Can be cached by the browser
- Remain in the browser history
- Can be bookmarked
- Should never be used when dealing with sensitive data (passwords, credit card info and such)
- Have length restrictions: The GET method adds the data to the URL and the length of a URL is limited (maximum URL length is 2048 characters)
- Have character restrictions: Only ASCII characters are allowed
- Should be used only to retrieve data, never to submit

GET Request

1. Create script “vezba1.php”. It should just echo the string “Hello world”.
2. Start the web server in the same folder with the command `php -S localhost:8000`.
3. Run the script in Chrome.
4. Open Chrome DevTools and observe what is happening.

Query parameters

Have you ever noticed something like this in URL?

“https://www.google.rs/search**?q=test**”

- The part written in bold is called a **Query string**.
- Query string consists of **query parameters** (name/value pairs).
- These parameters are sent via **GET** method.
- Now click on this link - Google will take this parameter and trigger the search with the string “**test**”.
- Multiple parameters are concatenated via **&** sign (**?q=test&n=100**).
- In PHP we can get these parameters with **$_GET** superglobal variable.
- For this exact reason, you should never send sensitive data through a GET request.

Query parameters

1. Run in browser “/vezba1.php?paramA=200&paramB=500”.
2. Inside the script use superglobal variable **$_GET** to get these parameters. You can do it, for example, like this:
   a. `$a = $_GET['paramA'];`
   b. `$b = $_GET['paramB'];`
3. Echo the values.
4. Try also to catch parameter that doesn’t exist, for example “paramC”. Display it’s value.

POST Request

POST Request

Even though you can send data to the server using GET and the query string, in many cases POST will be preferable.

Sending large amounts of data using GET is not practical and has limitations and POST is safer for sending sensitive info.

POST requests are most commonly sent by web forms.

POST Request

- Never cached
- Do not remain in the browser history
- Cannot be bookmarked
- Have no restrictions on data length; Binary data is also allowed.
- In PHP you can use **$_POST** superglobal variable to catch the parameters.

POST Request

When a web browser sends a POST request from a form, the default media type is “application/x-www-form-urlencoded”.

The value "multipart/form-data" should be used in combination with the input element, type="file".

HTML forms

```
<form action="/vezba1.php" method=”POST”>
  <label for=”name”>Your name</label>
  <input type=”name” id=”name” placeholder=”John Doe”/>
  <label for=”email”>Your Email</label>
  <input type=”email” id=”email” placeholder=”john@doe.com”/>
  <label for=”phone”>Your Phone number</label>
  <input type=”phone” id=”phone” placeholder=”000-000-000”/>
  <label>Can I call you back</label>
  <input type=”radio” name=”callback” id=”callback-yes” checked>
  <label for=”callback”>Yes</label>
  <input type=”radio” name=”callback” id=”callback-no”>
  <label for=”callback”>No</label>
</form>
```

HTML Forms

The **method** attribute specifies how to send **form-data**.

The **action** attribute is used to inform the browser what page (or script) to call once the "**submit**" button is pressed.

Here, the **action** attribute tells the browser to send the form data using the **POST** method, to '**/vezba1.php**', so it could process it.

When using data from forms, you will probably never be in the need to use **GET** request, except in cases when the form is some kind of a search form.

Exercise

- Create HTML form in script “**vezba1.php**” that will have two text fields: **name** and **email**.
- Action attribute should point to the same script (Fun fact: HTTP Post to the same page that the form is on is called postback).
- Method should be **POST**.
- Open Chrome DevTools and observe what is happening.
- Submit the form.

Exercise

- Check the Request method of the new request to script “vezba1.php”.
- Check the sent parameters in Request payload section.
- Inside the script use superglobal variable **$_POST** to get these parameters. You can do it, for example, like this:
  - `$name = $_POST['name'];`
  - `$email = $_POST['email'];`
- Echo the values.
- Try also to catch parameter that doesn’t exist, for example “age”. Display its value.

HTTP Response

HTTP Response

Consist of three parts:

- Status line
- Zero or more headers
- Optional message body

HTTP Response

Status line contains info about HTTP version and a status code.

The Status-Code element is a 3-digit integer where first digit of the Status-Code defines the class of response.

HTTP Response

There are 5 values for the first digit:

- **1xx**: **Informational** - It means the request was received and the process is continuing.
- **2xx**: **Success** - It means the action was successfully received, understood, and accepted.
- **3xx**: **Redirection** - It means further action must be taken in order to complete the request.
- **4xx**: **Client Error** - It means the request contains incorrect syntax or cannot be fulfilled.
- **5xx**: **Server Error** - It means the server failed to fulfill an apparently valid request.

HTTP Response

Most notable status codes:

- **200 OK**: Standard response for successful HTTP requests.
- **400 Bad Request**: The server cannot or will not process the request due to an apparent client error.
- **401 Unauthorized**: Authentication is required and has failed or has not yet been provided.
- **404 Not Found**: The requested resource could not be found.
- **500 Internal Server Error**: A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

HTTP Response Headers

Headers contain various info about the response. Most common used are:

- **Content-type** - Defines the type of the body (HTML, json…).
- **Content-length** - Indicates the size of the message body.
- **Set-cookie** (we will speak about this tomorrow)

HTTP Response Body

Message body contains what the browser will get and parse in order to display data to the end user.

Now open “vezba1.php” and like you checked the requests, check the responses in Chrome DevTools.

Useful links

- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://www.tutorialspoint.com/http/index.htm