Shahzaib Khan 20F1079 Bscs5B

WebAssignment#2

* **Versions of HTTP**

HTTP is a request-response protocol in the client-server computing model. It has 4 versions.

1. **HTTP/0.9**

* Initial version of HTTP, a simple client-server, request-response, tele net-friendly protocol
* Request nature: single line (method + path for requested document)
* Methods supported: GET only
* Response type: hypertext only
* Connection nature: terminated immediately after the response
* No HTTP headers (cannot transfer other content type files), No status/error codes, No URLs, No versioning

1. **HTTP/1.0**

* Browser-friendly protocol
* Provided header fields including rich metadata about both request and response (HTTP version number, status code, content type)
* Response: not limited to hypertext (transmit files other than plain HTML files e.g. scripts, stylesheets, media)
* Methods supported: GET, HEAD, POST
* Connection nature: terminated immediately after the response

1. **HTTP/1.1**

* This is the HTTP version currently in common use.
* Introduced critical performance optimizations and feature enhancements — persistent and pipelined connections, chunked transfers, compression/decompression, content negotiations, virtual hosting (a server with a single IP Address hosting multiple domains), faster response, and great bandwidth savings by adding cache support.
* Methods supported: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS
* Connection nature: long-lived

1. **HTTP/2.0**

HTTP/2 was developed over the SPDY protocol. HTTP/2 works on the binary framing layer instead of textual that converts all the messages into binary format. it works on fully multiplexed that is one TCP connection is used for multiple requests. HTTP/2 uses HPACK which is used to split data from the header. it compresses the header. The server sends all the other files like CSS & JS without the request of the client using the PUSH frame.

* **Status codes of HTTP**

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.

**100 to 199: Informational**

It means the request has been received and the process is continuing.

|  |  |
| --- | --- |
| 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. |

**200 to 299: Success**

It means the action was successfully received, understood, and accepted.

|  |  |
| --- | --- |
| 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. |

**300 to 399: Redirection**

It means further action must be taken in order to complete the request.

|  |  |
| --- | --- |
| 300 Multiple Choices | A link list. 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. |

**400 to 499: Client Error**

It means the request contains incorrect syntax or cannot be fulfilled.

|  |  |
| --- | --- |
| 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 can not use this code yet*. |
| 403 Forbidden | Access is forbidden to the requested page. |
| 404 Not Found | The server can not 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. |

**500 to 599: Server Error**

It means the server failed to fulfill an apparently valid request.

|  |  |
| --- | --- |
| 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. |

* **Request Methods of HTTP**

**HTTP (Hypertext Transfer Protocol)** specifies a collection of request methods to specify what action is to be performed on a particular resource. The most commonly used HTTP request methods are **GET, POST, PUT, PATCH, and DELETE**. These are equivalent to the **CRUD operations (create, read, update, and delete)**.

**GET:**GET request is used to read/retrieve data from a web server. GET returns an HTTP status code of **200 (OK)** if the data is successfully retrieved from the server.

* Sent when URL is submitted in the browser location bar or a user clicks a link
* Send data in the URL and query string
* Reloadable
* Can bookmark
* Used for read-only operations
  + View, search, sort, or filter data
  + Data does not change

**POST:**POST request is used to send data (file, form data, etc.) to the server. On successful creation, it returns an HTTP status code of **201**.

* Sent when web form is submitted
* Send data in the URL and as an attachment
* Semi-reloadable (prompt to send data again)
* Can not bookmark
* Used for write operations
  + Create, update, or delete data
  + Data does change

**PUT:**A PUT request is used to modify the data on the server. It replaces the entire content at a particular location with data that is passed in the body payload. If there are no resources that match the request, it will generate one.

**PATCH:**PATCH is similar to PUT request, but the only difference is, it modifies a part of the data. It will only replace the content that you want to update.

**DELETE:**ADELETE request is used to delete the data on the server at a specified location.

[**CONNECT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT)**:** The CONNECT method establishes a tunnel to the server identified by the target resource.

[**OPTIONS**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS): The OPTIONS method describes the communication options for the target resource.

[**TRACE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/TRACE)**:** The TRACE method performs a message loop-back test along the path to the target resource.

[**PATCH**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH): The PATCH method applies partial modifications to a resource.