Full Stack Application Development- SE ZG503

Akshaya Ganesan

CSIS, WILP

BITS Pilani

Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Lecture No: 4 – Web Protocols

Module 3: Web Protocols

HTTP
- HTTP Request- Response and its structure
- HTTP Methods;
- HTTP Headers
- Connection management - HTTP/1.1 and HTTP/2

Synchronous and asynchronous communication

Communication with Backend
- AJAX, Fetch API
- Webhooks
- Server-Sent Events

Polling

Bidirectional communication - Web sockets

innovate achieve lead

BITS Pilani

Pilani Campus

HTTP

HyperText Transfer Protocol(HTTP)

HTTP is a request-response client-server protocol

HTTP is a stateless protocol.

HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems.

Each HTTP communication (request or response) between a browser and a Web server consists of two parts:

- A header and
- A body

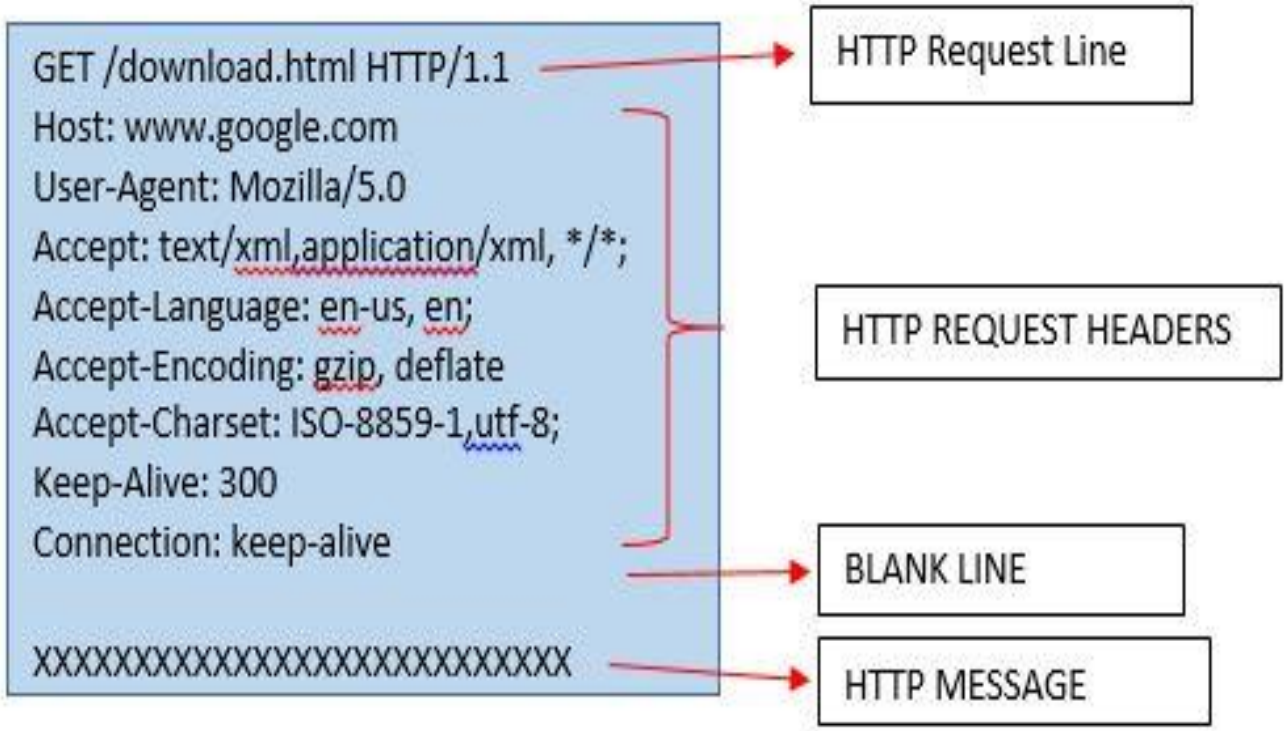
The header contains information about the communication.

The body contains the data of the communication (optional).

HTTP Request

The general form of an HTTP request is:

1. HTTP Request Line
2. Header fields
3. Blank line
4. Message body (Optional)



Request Line

The first line of the header is called the request line:

request-method-name /request-URI /HTTP-version

Examples of request line are:

GET /test.html HTTP/1.1

HEAD /query.html HTTP/1.0

POST /index.html HTTP/1.1

Request Headers

The request headers are in the form of name:value pairs. Multiple values, separated by commas, can be specified.

request-header-name: request-header-value1, request-header-value2, ...

Examples of request headers are:

Host: www.xyz.com

Connection: Keep-Alive

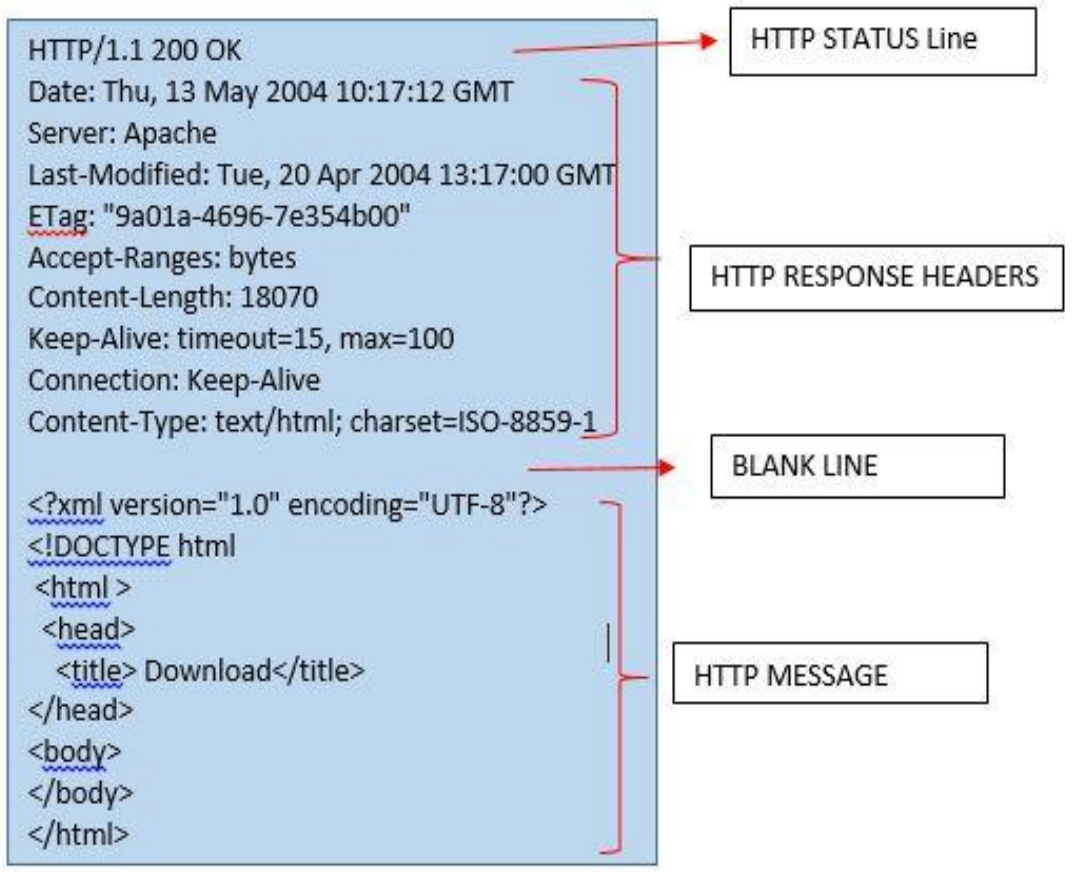
Accept: image/gif, image/jpeg, */*

Accept-Language: us-en, fr, cn

HTTP Response

The general form of an HTTP response is:

1. Status line
2. Response header fields
3. Blank line
4. Response body



Status Line

The first line is called the status line.

HTTP-version status-code reason-phrase

status-code: a 3-digit number generated by the server to reflect the outcome of the request.

reason-phrase: gives a short explanation to the status code.

Common status code and reason phrase are "200 OK", "404 Not Found", "403 Forbidden", "500 Internal Server Error".

Examples of status line are:

HTTP/1.1 200 OK

HTTP/1.0 404 Not Found

HTTP/1.1 403 Forbidden

STATUS CODE

Status code is a three-digit number; first digit the specifies the general status

1 => Informational

2 => Success

3 => Redirection

4 => Client error

5 => Server error

Common HTTP Response Status Codes

200 OK Indicates a nonspecific success

201 Created Sent primarily by collections and stores but sometimes also by controllers, to indicate that a new resource has been created

204 No Content Indicates that the body has been intentionally left blank

301 Moved Permanently Indicates that a new *permanent* URI has been assigned to the client's requested resource

303 See Other Sent by controllers to return results that it considers optional

401 Unauthorized Sent when the client either provided invalid credentials or forgot to send them

402 Forbidden Sent to deny access to a protected resource

404 Not Found Sent when the client tried to interact with a URI that the REST API could not map to a resource

405 Method Not Allowed Sent when the client tried to interact using an unsupported HTTP method

406 Not Acceptable Sent when the client tried to request data in an unsupported media type format

500 Internal Server Error Tells the client that the API is having problems of its own

Response Headers

The response headers are in the form name:value pairs:

response-header-name: response-header-value1, response-header-value2, ...

Examples of response headers are:

Content-Type: text/html

Content-Length: 35

Connection: Keep-Alive

Keep-Alive: timeout=15, max=100

The response message body contains the resource data requested.

HTTP Methods (Verbs)

GET - Fetch a URL

HEAD - Fetch information about a URL

PUT - Store to an URL

POST - Send form data to a URL and get a response back

DELETE - Delete a resource in URL

GET and POST (forms) are commonly used.

HTTP Methods

HTTP defines a set of **request methods** to indicate the desired action for a given resource.

The request methods are sometimes referred to as *HTTP verbs*.

GET - Fetch a URL

HEAD - Fetch information about a URL

PUT - Store to an URL

POST - Send form data to a URL and get a response back

DELETE - Delete a resource in URL

GET and POST (forms) are commonly used.

HTTP Methods

GET
The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

The GET request method is said to be a safe operation, which means it should not change the state of any resource on the server.

The GET method is used to request any of the following resources:

A webpage or HTML file.

An image or video.

A JSON document.

A CSS file or JavaScript file.

An XML file.

HTTP Methods

POST:

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

The POST HTTP request method sends data to the server for processing.

The data sent to the server is typically in the following form:

- Input fields from online forms.
- XML or JSON data.
- Text data from query parameters.

A POST operation is not considered a safe operation, as it has the power to update the state of the server and cause potential side effects to the server's state when executed.

The HTTP POST method is not required to be idempotent either, which means it can leave data and resources on the server in a different state each time it is invoked.

HTTP Methods

HEAD
The HEAD method requests a response identical to a GET request but without the response body.

The HTTP HEAD method returns metadata about a resource on the server.

The HTTP HEAD method is commonly used to check the following conditions:

- The size of a resource on the server.
- If a resource exists on the server or not.
- The last-modified date of a resource.
- Validity of a cached resource on the server.
- The following example shows sample data returned from a HEAD request:

HTTP/1.1 200 OK
Date: Fri, 19 Aug 2023 12:00:00 GMT
Content-Type: text/html
Content-Length: 1234
Last-Modified: Thu, 18 Aug 2023 15:30:00 GMT

HTTP Methods

PUT

The HTTP PUT method is used to replace a resource identified with a given URL completely.

The HTTP PUT request method includes two rules:

A PUT operation always includes a payload that describes a completely new resource definition to be saved by the server.

The PUT operation uses the exact URL of the target resource.

If a resource exists at the URL provided by a PUT operation, the resource's representation is completely replaced.

A new resource is created if a resource does not exist at that URL.

The payload of a PUT operation can be anything that the server understands, although JSON and XML are the most common data exchange formats for RESTful web services.

HTTP Methods

DELETE
The DELETE method deletes the specified resource.

CONNECT
The CONNECT method establishes a tunnel to the server identified by the target resource.

OPTIONS
The OPTIONS method describes the communication options for the target resource.

TRACE
The TRACE method performs a message loop-back test along the path to the target resource.

PATCH
The PATCH method applies partial modifications to a resource.

HTTP Request Methods

innovate achieve lead

HTTP Request methods		
	SAFE	**IDEMPOTENT**
GET	Yes	Yes
POST	No	No
PUT	No	Yes
PATCH	No	No
DELETE	No	Yes
TRACE	Yes	Yes
HEAD	Yes	Yes
OPTIONS	Yes	Yes
CONNECT	No	No

HTTP HEADERS

HTTP headers allow the client and the server to pass additional information with the request or the response.

An HTTP header consists of its name followed by a colon ':', then by its value.

Headers can be grouped according to their contexts:

General header: Headers applying to both requests and responses but with no relation to the data eventually transmitted in the body.

Request header: Headers containing more information about the resource to be fetched or about the client.

Response header: Headers with additional information about the response, like its location or about the server itself (name and version etc.).

Entity header: Headers containing more information about the body of the entity, like its content length or its MIME-type.

HTTP Headers –Content

The Accept header

The Accept header lists the MIME types of media resources that the agent is willing to process.

Each combined with a quality factor, a parameter indicating the relative degree of preference between the different MIME types.

A media type also known as MIME type is a two-part identifier for file formats and format contents transmitted on the Internet.

Form: type/subtype

Examples: `text/plain`, `text/html`, `image/gif`, `image/jpeg`

Accept: image/gif, image/jpeg, */*

The Accept-Charset header

It indicates to the server what kinds of character encodings are understood by the user-agent.

Accept-Charset:utf-8

HTTP Headers -Content

The Accept-Encoding header

The Accept-Encoding header defines the acceptable content-encoding (supported compressions).

The value is a q-factor list (e.g.: br, gzip;q=0.8) that indicates the priority of the encoding values.

Compressing HTTP messages is one of the most important ways to improve the performance of a Web site.

Accept-Encoding:gzip, deflate

The Accept-Language header

It is used to indicate the language preference of the user.

Accept-Language: en-us

The User-Agent header

It identifies the browser sending the request.

HTTP Headers- Caching

Cache-Control header

The Cache-Control general-header field is used to specify directives for caching mechanisms in both requests and responses.

Caching directives are unidirectional, i.e., directive in a request is not implying that the same directive is to be given in the response.

Standard Cache-Control directives that can be used by the client in an HTTP request.

- Cache-Control: max-age=<seconds>
- Cache-Control: no-cache
- Cache-Control: no-store
- Cache-Control: no-transform

Standard Cache-Control directives that can be used by the server in an HTTP response.

- Cache-Control: must-revalidate
- Cache-Control: no-cache
- Cache-Control: no-store
- Cache-Control: no-transform
- Cache-Control: public
- Cache-Control: private

HTTP Headers- Caching

Expires Header

The Expires header contains the date/time after which the response is considered stale.

Invalid dates, like the value 0, represent a date in the past and mean that the resource is already expired.

If there is a Cache-Control header with the "max-age" directive in the response, the Expires header is ignored.

Expires: Wed, 21 Oct 2015 07:28:00 GMT

HTTP Headers –Caching Etag

The ETag HTTP response header is an identifier for a specific version of a resource.

If the resource at a given URL changes, a new Etag value must be generated.

ETag: "33a64df551425fcc55e4d42a148795d9f25f89d4“

The client will send the Etag value of its cached resource along in an If-None-Match header field:

If-None-Match: "33a64df551425fcc55e4d42a148795d9f25f89d4“

The server compares the client's ETag (sent with If-None-Match) with the ETag for its current version of the resource

If both values match, the server send back a 304 Not Modified status, without any body

Tells the client that the cached version of the response is still good.

HTTP Headers-Caching Last-Modified Header

The Last-Modified response HTTP header contains the date and time at which the origin server believes the resource was last modified.

It is used as a validator to determine if a resource received or stored is the same.

Less accurate than an ETag header

Redirection

In HTTP, a redirection is triggered by the server by sending special responses to a request: redirects.

HTTP redirects are responses with a status code of 3xx.

A browser, when receiving a redirect response, uses the new URL provided in the location header.

Location: /index.html

Permanent redirections

301 Moved Permanently

Temporary Redirections

302 Temporary Redirect

HTTP Headers-Cookies

An HTTP cookie is a small piece of data that a server sends to the user's web browser.

The browser may store it and send it back with the next request to the same server.

The Set-Cookie HTTP response header sends cookies from the server to the user agent.

Set-Cookie: <cookie-name>=<cookie-value>

The Cookie HTTP request header contains stored HTTP cookies previously sent by the server with the Set-Cookie header.

Cookie: name=value; name2=value2; name3=value3

Summary

HTTP Request response

HTTP Methods

HTTP Headers

innovate achieve lead

BITS Pilani

Pilani Campus

Thank you