HTTP Verbs

.NET

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs.

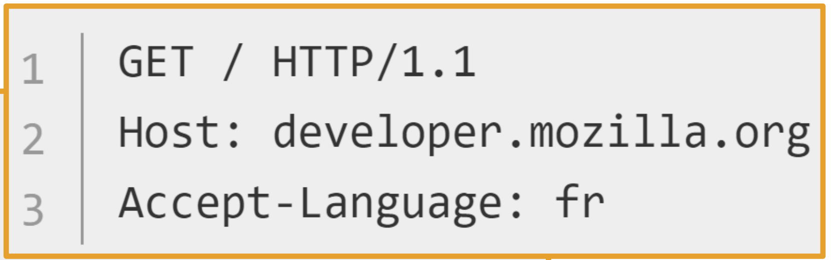
HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/WEB/HTTP/METHODS

Request Methods (verbs)

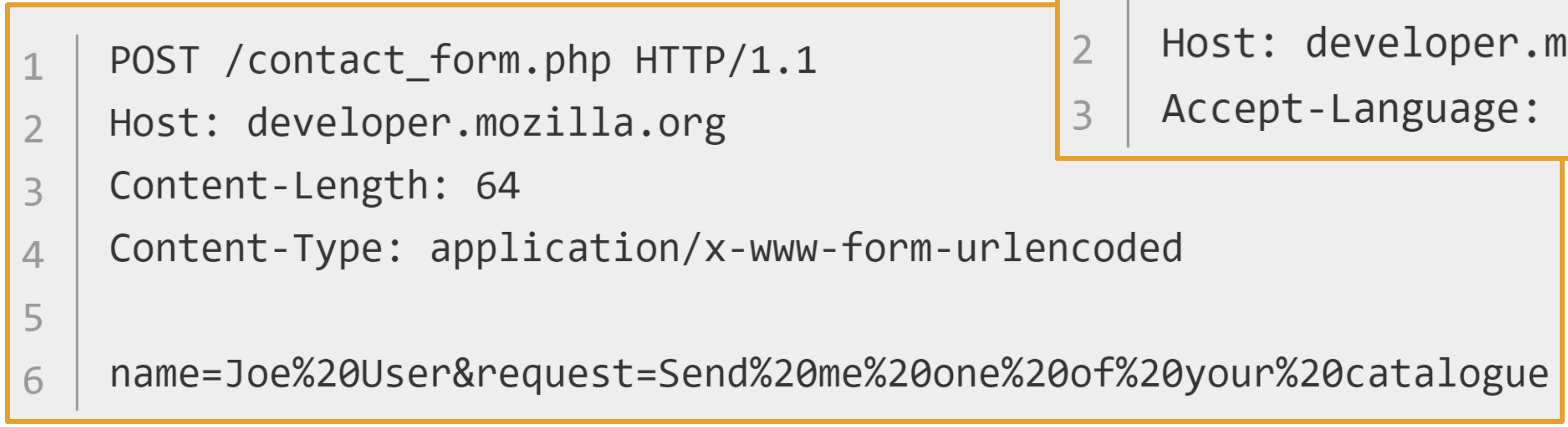
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
https://developer.mozilla.org/en-US/docs/Web/HTTP/Session

Request methods (***HTTP verbs***) indicate the desired action to be performed on a resource. HTTP verbs are ***GET***, ***POST***, ***PUT***, ***DELETE***, ***TRACE***, ***HEAD***, ***CONNECT***, and ***OPTIONS***.

- The ***POST*** method sends data to a server. ***POST*** is used mainly for HTML ***Forms***.
- The ***GET*** method requests/retrieves data.



```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```



```
POST /contact_form.php HTTP/1.1
Host: developer.mozilla.org
Content-Length: 64
Content-Type: application/x-www-form-urlencoded

name=Joe%20User&request=Send%20me%20one%20of%20your%20catalogue
```

Request Methods (verbs)

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

HTTP Verb	Description
*GET	Requests the specified resource. GET should only retrieve data.
HEAD	Just like a GET request, but without the response body.
*POST	Used to submit an entity to the specified ***resource***.
*PUT	Replaces all current representations of the target resource with the payload.
*DELETE	Deletes the specified resource.
CONNECT	Establishes a tunnel to the server identified by the target resource.
OPTIONS	Describes the communication options for the target resource.
TRACE	Performs a message loop-back test along the path to the target resource.
PATCH	Applies partial modifications to a resource.

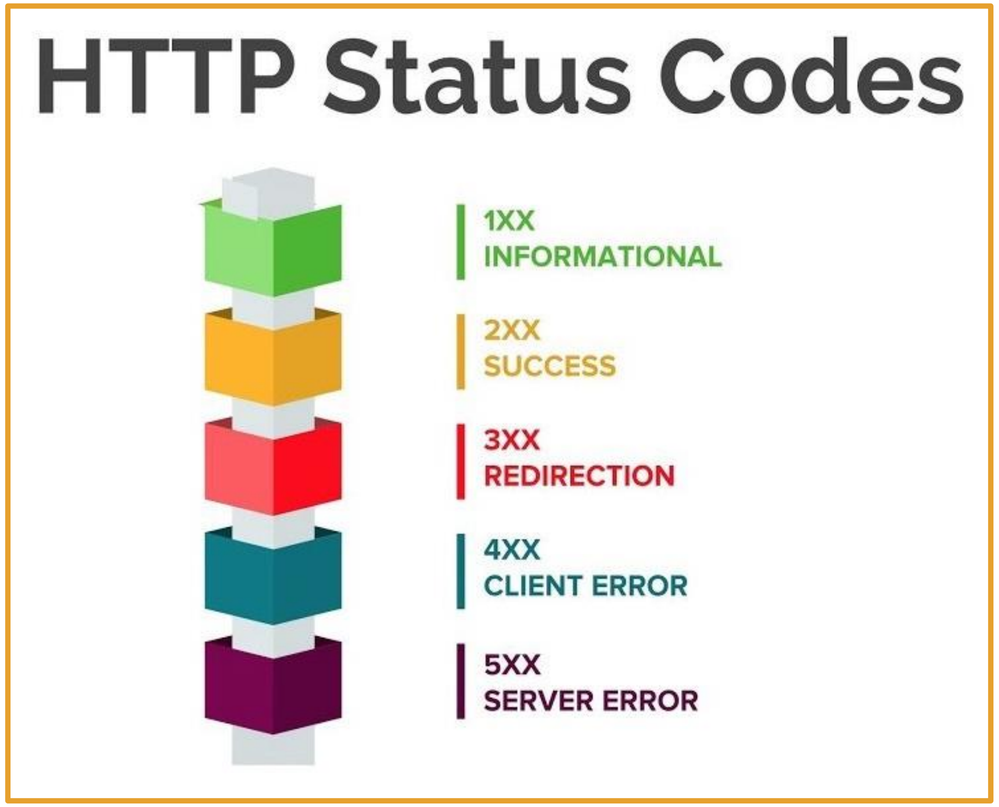
*most common

Response Status Codes

https://developer.mozilla.org/en-US/docs/Web/HTTP/Session
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
https://tools.ietf.org/html/rfc2616#section-10

HTTP response status codes give the result of an HTTP request. Responses are grouped in five general categories:

- Informational responses (100–199),
- Successful responses (200–299),
- Redirects (300–399),
- Client errors (400–499),
- and Server errors (500–599).
- Cheat Sheet



Common Response Status Codes

https://www.smartlabsoftware.com/ref/http-status-codes.htm
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

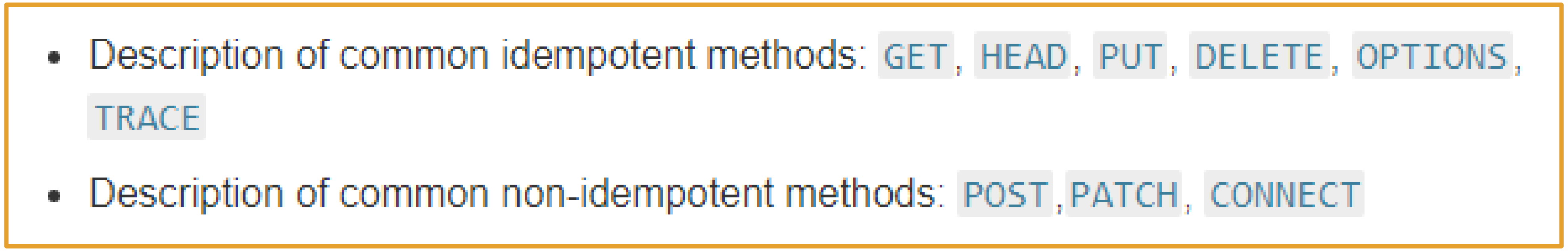
Code number	Code Meaning
200 OK, 201	The request succeeded. Request has been fulfilled resulting in new resource(s) created.
300 Multiple Choices	The requested resource has different choices and cannot be resolved into one.
301 Moved Permanently	The requested resource has been assigned a new permanent URI
304 Not Modified	Client performed a conditional GET request. Access is allowed. The document is unmodified
307 Temporary Redirect	The requested resource resides temporarily under a different URI.
400 Bad Request	The request could not be understood by the server due to malformed syntax.
401 Unauthorized	The request requires user authentication.
403 Forbidden	The server understood the request but is refusing to fulfill it.
404 Not Found	The server has not found anything matching the Request-URI.
410 Gone	The requested resource is no longer available at the server and no forwarding address is known.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
501 Not Implemented	The server does not support the functionality required to fulfill the request.
503 Service Unavailable	Your web server is unable to handle your HTTP request at the time.
550 Permission Denied	Your account does not have permission to perform the action you are attempting.

'Safe' and 'Idempotent'

https://developer.mozilla.org/en-US/docs/Glossary/safe
https://developer.mozilla.org/en-US/docs/Glossary/Idempotent

Safe – An HTTP method that doesn't alter the state of the resource, meaning it leads to a read-only operation. ***GET***, ***HEAD***, and ***OPTIONS*** are ***safe.*** All ***safe*** methods are also ***idempotent***.

Idempotent – An identical request can be made once or several times in a row with the same effect while leaving the server in the same state. Implemented correctly, the *GET*, *HEAD*, *PUT*, and *DELETE* method are ***idempotent***, but not the *POST* method. Not all ***idempotent*** methods are ***safe***. For example, *PUT* and *DELETE* are both ***idempotent*** but unsafe.

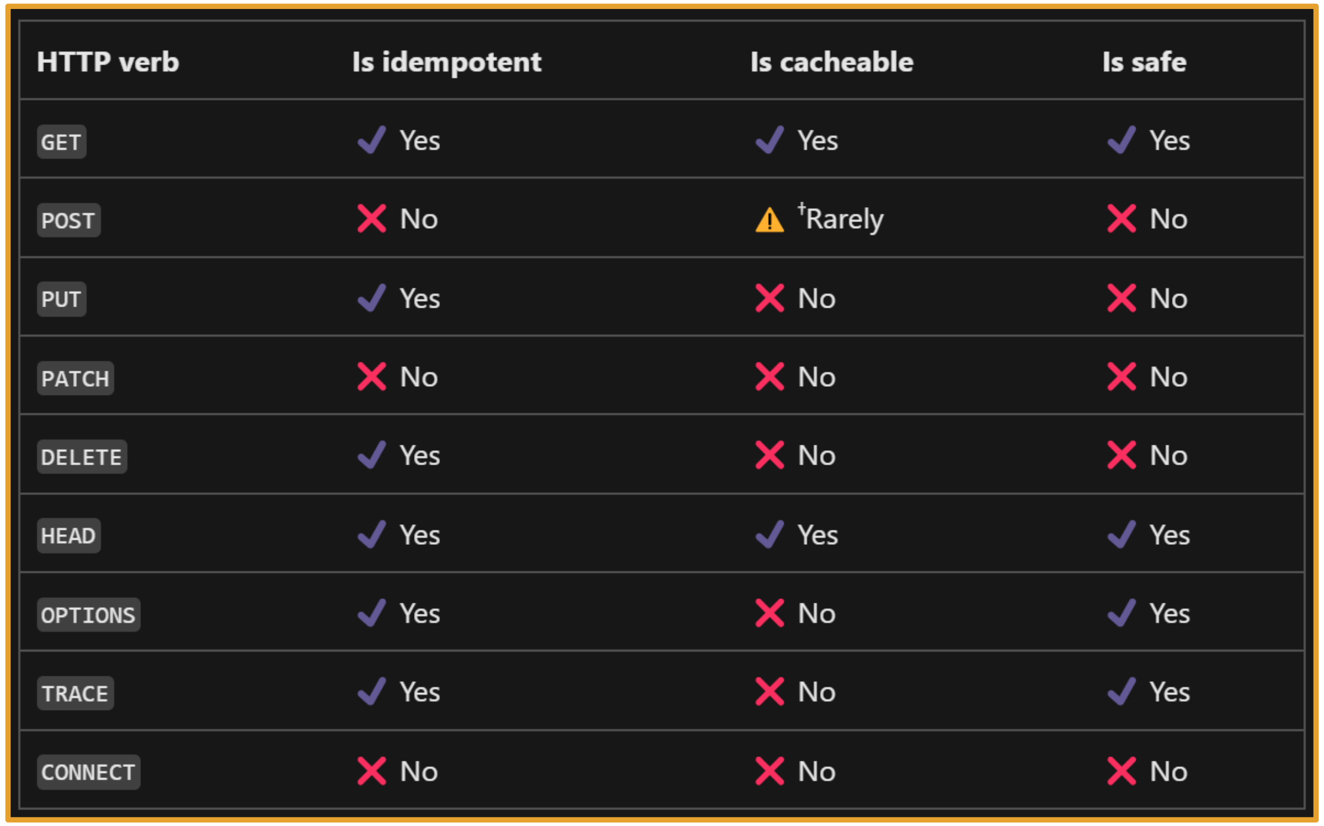


- Description of common idempotent methods: `GET`, `HEAD`, `PUT`, `DELETE`, `OPTIONS`, `TRACE`
- Description of common non-idempotent methods: `POST`,`PATCH`, `CONNECT`

Is my request Idempotent?

https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/http-overview#:~:text=methods%20are%20differentiated
https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/http-overview#:~:text=the%20request

- Safe – The request doesn't alter the state of the resource, meaning it leads to a read-only operation. All safe methods are idempotent.
- Idempotent – An identical request can be made once or several times in a row with the same overall effect.



HTTP verb	Is idempotent	Is cacheable	Is safe
GET	✓ Yes	✓ Yes	✓ Yes
POST	✗ No	⚠ †Rarely	✗ No
PUT	✓ Yes	✗ No	✗ No
PATCH	✗ No	✗ No	✗ No
DELETE	✓ Yes	✗ No	✗ No
HEAD	✓ Yes	✓ Yes	✓ Yes
OPTIONS	✓ Yes	✗ No	✓ Yes
TRACE	✓ Yes	✗ No	✓ Yes
CONNECT	✗ No	✗ No	✗ No

Additional Resources

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#HTTP_Messages

https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design