

IDEMPOTENCY

Idempotency is the concept of where if an action is performed multiple times, it yields the same result as if it has only been done once. This means if it was an action that would change the state of the system, it only does it on the first time the action is triggered.

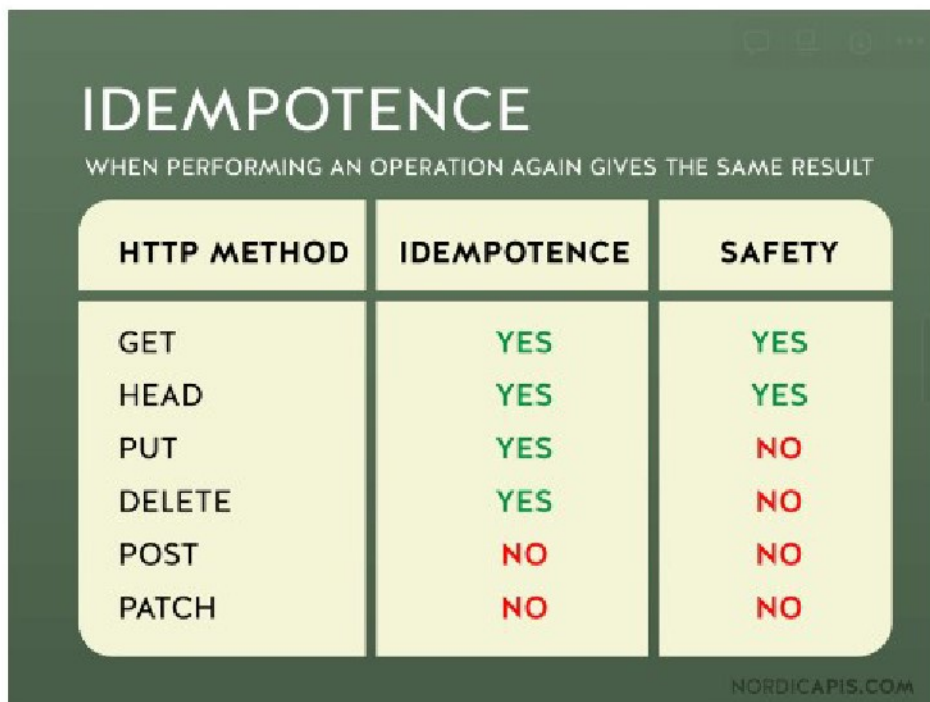
For example:

If a device has a separate ON and OFF switch, the ON switch turns it on, the OFF switch turns it off. Once the device is off, pushing the OFF button more times will not provide different outcomes, and the same applies to the OFF switch, once something is off, pressing this button will not change the state.

A toggle switch however, is not Idempotent, every time it is used, a state is changed, every use of the switch will affect the state of something.

In the context of RESTful web APIs, Idempotency means that if the same request is made multiple times, the server's response will be the same each time and the state of the system will not be changed beyond the first request. This means that if a request is made accidentally or due to a network error, it can simply be retried without causing any unintended side effects. Idempotency is an important concept in RESTful API design because it helps ensure that the API can be used safely and reliably by clients.

When developing our API's we need to consider Idempotency when we design our endpoints and try to follow the following standards when developing our endpoints and endpoint methods.



IDEMPOTENCE
WHEN PERFORMING AN OPERATION AGAIN GIVES THE SAME RESULT

HTTP METHOD	IDEMPOTENCE	SAFETY
GET	YES	YES
HEAD	YES	YES
PUT	YES	NO
DELETE	YES	NO
POST	NO	NO
PATCH	NO	NO

NORDICAPIS.COM

NOTE: An options request method would also be Idempotent and safe as it does not change the state of the system.

- GET methods should be Idempotent as they change no state, they only view data.
- HEAD methods should also be idempotent as they make no changes to the state
- OPTIONS methods should also be idempotent as they make no changes to the state
- PUT and DELETE methods should be designed to be idempotent so that they only target the same records no matter how many times they are called.
- POST and PATCH methods are generally not required to be idempotent as they are adding/changing data which can be repeated.
- We can still write our code to minimise this

Read only methods such as the GET, OPTIONS and HEAD are also considered safe methods, because they do not modify or change any state in the server or data. They only Read and view it.