**WHAT IS IDEMPOTENCY?**

You can divide HTTP methods into two main categories **safe** and **idempotent**.

SAFE METHODS:

Safe methods are HTTP methods that do not modify the resource

e.g. a GET request

GET request is safe  because it doesn't modify the resource you are requesting

e.g. data of a Book.

Another safe HTTP method is HEAD, which doesn't change the resource representation on the Server, but all other HTTP methods e.g. POST, PUT, or DELETE are non-safe.

Safe methods can be **cached**and  **prefected**without any repercussions or side-effect to the resource . Here is an example of safe method

GET /order/123 HTTP/1.1

This will retrieve the order with orderId 123.

No matter how many times you execute this method, the order in the server will not be modified or impacted. That's why [GET method](http://java67.blogspot.com/2014/08/difference-between-post-and-get-request.html) is a safe method.

IDEMPOTENT METHODS:

Coming to **idempotent methods**, they are HTTP methods which can be called multiple times and they will produce the same result.

They are considered the safe option to update a resource on the Server.  
  
Some examples of idempotent HTTP methods are GET, PUT, and PATCH.

No matter how many times you call them, they will produce the same result with same URI.

Maths is good place to explain idempotent methods, consider the following example:

int i = 30; // idempotent

i++; // not idempotent

Here the assignment operation is idempotent, no matter how many times you execute this statement, i will always be 30. The**second example is not idempotent**.

Idempotency is an important thing while building a fault-tolerant RESTful API.

Idempotency is also the reason of [why should you use PUT over POST to update a resource in REST](http://java67.blogspot.com/2015/09/top-10-restful-web-service-interview-questions-answers.html).   
  
For example, suppose a client wants to update a resource through POST. Since **POST is not an idempotent method**, calling it multiple times may result in incorrect updates.  
  
In the real world its quietly likely that your POST request may timeout, what will happen to the resource that.