

REST API (Introduction)

Representational State Transfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. REST API is a way of accessing web services in a simple and flexible way without having any processing.

REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses less bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web service. All communication done via REST API uses only HTTP request.

Working

A request is sent from client to server in the form of web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from server in the form of a resource which can be anything like HTML, XML, Image or JSON. But now JSON is the most popular format being used in Web Services.



In HTTP there are five methods which are commonly used in a REST based Architecture i.e., POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations respectively. There are other methods which are less frequently used like OPTIONS and HEAD.

GET: The HTTP GET method is used to read (or retrieve) a representation of a resource. In the safe path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

POST: The POST verb is most-often utilized to create new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource. On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.

NOTE: POST is neither safe nor idempotent.

PUT: It is used for updating the capabilities. However, PUT can also be used to create a resource in the case where the resource ID is chosen by the client instead of by the server. In other words, if the PUT is to a URI that contains the value of a non-existent resource ID. On successful update, return 200 (or 204 if not returning any content in the body) from a PUT. If using PUT for create, return HTTP status 201 on successful creation. PUT is not safe operation but it's idempotent.

PATCH: It is used for modify capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource. This resembles PUT, but the body contains a set of instructions describing how a resource currently residing on the server should be modified to produce a new version. This means that the PATCH body should not just be a modified part of the resource, but in some kind of patch language like JSON Patch or XML Patch. PATCH is neither safe nor idempotent.

DELETE: It is used to delete a resource identified by a URI. On successful deletion, return HTTP status 200 (OK) along with a response body.

Idempotence: An idempotent HTTP method is a HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. Again, this only applies to the result, not the resource itself. Example,

1. $a = 4$ // It is Idempotence, as final value($a = 4$)

```
// would not change after executing it multiple  
// times.
```

2. a++ // It is not Idempotence because the final value
// will depend upon the number of times the
// statement is executed.

RESTful web services are very popular because they are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.