Caching REST API Response

Caching is the ability to store copies of frequently accessed data in several places along the request-response path. When a consumer requests a resource representation, the request goes through a cache or a series of caches (local cache, proxy cache, or reverse proxy) toward the service hosting the resource. If any of the caches along the request path has a fresh copy of the requested representation, it uses that copy to satisfy the request. If none of the caches can satisfy the request, the request travels all the way to the service (or origin server as it is formally known).

Using HTTP headers, an origin server indicates whether a response can be cached and, if so, by whom, and for how long. Caches along the response path can take a copy of a response, but only if the caching metadata allows them to do so.

can be made cacheable if either an `Expires` header or a `Cache-Control` header with a directive, to explicitly allows caching, is added to the response. Responses to `PUT` and `DELETE` requests are not cacheable at all.

There are two main HTTP response headers that we can use to control caching behavior:

Expires

The Expires HTTP header specifies an absolute expiry time for a cached representation. Beyond that time, a cached representation is considered stale and must be re-validated with the origin server. To indicate that a representation never expires, a service can include a time up to one year in the future.

- Information about the device's operating system
- Cookie information
- Information about other identifiers assigned to the device
- The IP address from which the device accesses a client's website or mobile application
- Information about the user's activity on that device, including web pages and mobile apps visited or used
- Information about the geographic location of the device when it accesses a website or mobile application

An ETag value is an opaque string token that a server associates with a resource to uniquely identify the state of the resource over its lifetime. When the resource changes, the ETag changes accordingly.

```
ETag: "abcd1234567n34jv"
```

Last-Modified

Whereas a response's Date header indicates when the response was generated, the Last-Modified header indicates when the associated resource last changed. The Last-Modified value cannot be less than the Date value.

```
Last-Modified: Fri, 10 May 2016 09:17:49 IST
```

If you are caching on a field level basis, and you want to update three fields of a record with a PUT request, the PUT response needs to contain the CURRENT modified date as well as the PREVIOUS modified date, using both the "Last-Modified:" and "If-Last-Modified:" headers.

Then this logic can be applied for updating the cache:
If the "If-Last-Modified:" date DOES NOT match the cache record's last modified date, invalidate the record and keep only the three fields that were updated in the PUT request.
If the "If-Last-Modified:" date DOES match the cache record's last modified date, simply update the three fields in the cache.

Of course, the cache's record's last modified date will also need to be updated using the value in the "Last-Modified:" header.

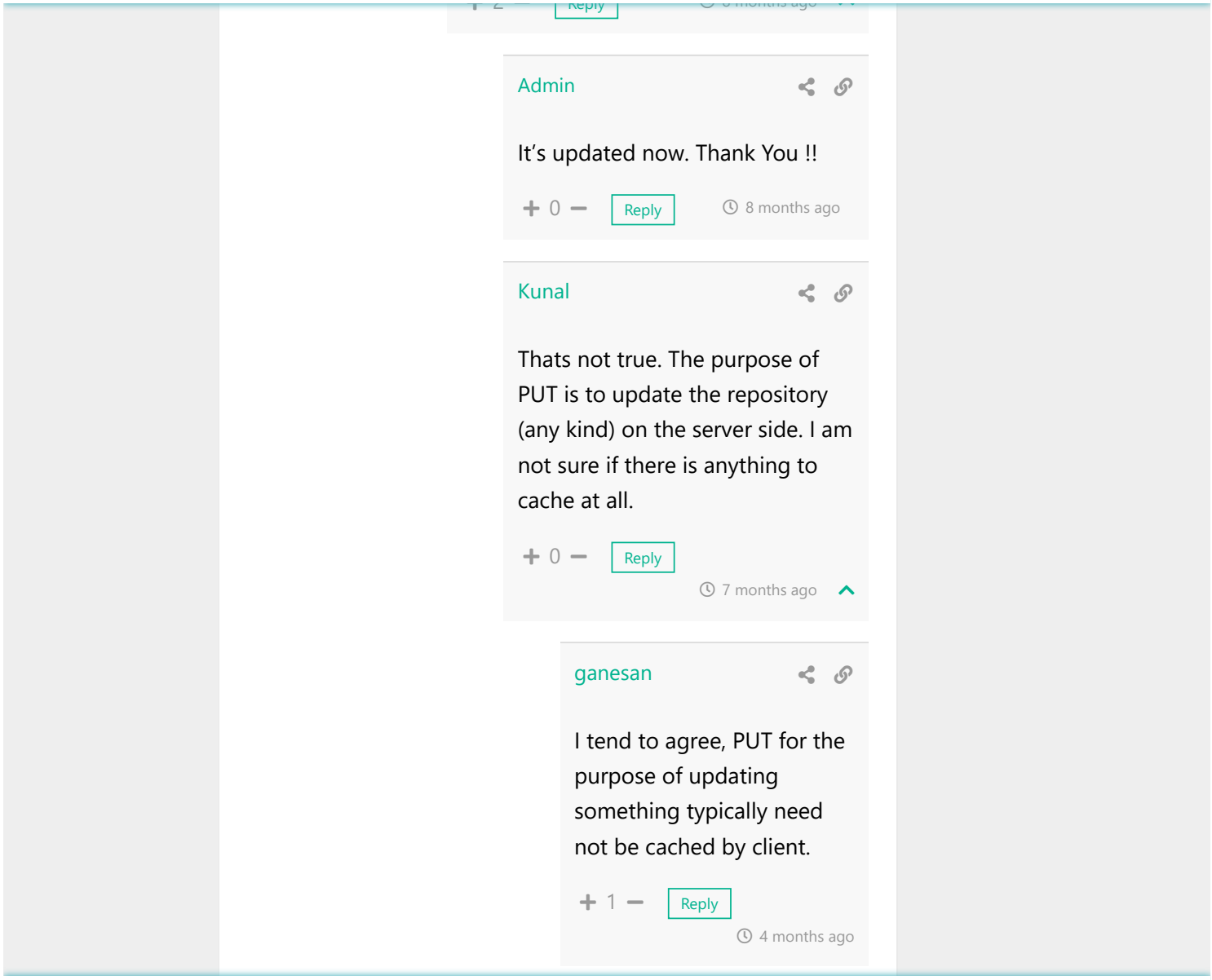
+ 0 − Reply 6 months ago

Nodon Darkeye

I'm confused on the "PUT vs POST" page I read this: "PUT is idempotent, so you can cache the response.".
On this page under "Caching in REST APIs" I read this: "Responses to PUT and DELETE requests are not cacheable at all."

Feels like one of the two needs to be altered.

- Information about the device's operating system
- Cookie information
- Information about other identifiers assigned to the device
- The IP address from which the device accesses a client's website or mobile application
- Information about the user's activity on that device, including web pages and mobile apps visited or used
- Information about the geographic location of the device when it accesses a website or mobile application



- Information about the device's operating system
- Cookie information
- Information about other identifiers assigned to the device
- The IP address from which the device accesses a client's website or mobile application
- Information about the user's activity on that device, including web pages and mobile apps visited or used
- Information about the geographic location of the device when it accesses a website or mobile application