**REST (Representational State Transfer)** is a fairly common way of interaction between client applications and services on the Internet. It is a set of restrictions considered when creating applications and web services. This architectural style was developed to help easily build convenient web services. A service written with regard to REST rules is called **RESTful**.

**REST** is NOT a protocol or a standard. REST usually works on top of HTTP and is one of the possible ways to use HTTP. It is not a standard, but rather a set of useful recommendations.

**Short history**

In 2000, *Roy Fielding* published his doctoral thesis on *"Architectural Styles and Network Architecture Design"*. In this paper, he proposes the idea that if a system meets certain six conditions, it becomes more convenient to use. He called this concept REST.

With this idea, Roy became a revolutionary for his time. Let's look at these conditions and see what they are about.

**Six REST principles**

Abiding by the following six important principles guarantees that you write a RESTful service:

1. **Client-server interaction model.** By separating the user interface from the data warehouse server, we improve and simplify application operations.
2. **Stateless.** Each request from a client to a server must contain all necessary information and cannot rely on any state stored on the server side. According to REST, the service does not store the results of the previous operation. Simply put, it works according to the *"Asked, answered, forgotten"* concept.
3. **Cacheable.** A request-response pair can be marked as cached and stored on the user side. It is possible to draw an analogy to web page caching: if a page was downloaded once, it could be accessed without addressing the server again.
4. **Uniform interface.** REST architecture specifies that any REST service must be understandable without its developer.
5. **Layered system.**A client cannot just tell whether it is connected directly to the end server or an intermediary along the way.
6. **Code on demand [Optional]**. On request, the service must give executable code in the form of an applet or script to be executed on the client's side. In practice, it is very seldom used.

**HTTP methods for RESTful services**

In the REST concept, interaction with resources is performed by calling the URL of the resource and four basic HTTP methods: GET, POST, PUT, DELETE. Let's consider in detail in which situations they are used.

* POST is used to create new resources. If the resource is successfully created, the HTTP code *201 (Created)*is returned, and also the address of the created resource is passed to the header Location.
* GET is used to retrieve or read the resource. If the GET method is successful and does not contain address errors, it returns an XML or JSON representation of the resource in combination with an HTTP *200 (OK)* status code. In case of errors, the code *404 (NOT FOUND)* or *400 (BAD REQUEST)* is usually returned.
* PUT method either creates a resource by the specified ID or updates an existing one. If the update is successful, the code is *200 (OK)*; *204 (NO CONTENT)* is returned if no content in the response body has been transmitted.
* DELETEmethod is used to remove a resource identified by a specific URI (ID). If the deletion is successful, *200 (OK)* HTTP code is returned, together with the response body containing the data of the remote resource. It is also possible to use HTTP code *204 (NO CONTENT)* without the response body.

Suppose we have a hypothetical book web service that uses the described methods to manage books as a resource. Each book stored in the service has its own numeric identifier. Then:

* POST https://example.com/books/ adds a new book
* GET https://example.com/books/1 gets an existing book
* PUT https://example.com/books/1 creates or updates a book by the given ID
* DELETE https://example.com/books/1 deletes an existing book by the given ID

This is what working with a typical RESTful web service looks like.

Read more on this topic in [Beginners Guide to REST API Testing](https://hyperskill.org/blog/post/beginners-guide-to-rest-api-testing)on Hyperskill Blog.

**Conclusion**

REST is a rich yet easy concept. It is one of the most common ways of organizing client/server connections. Using its six basic principles could make your work easier and will help you avoid common problems of web service building. If you want more information and clarification, check out some [external resources](https://www.restapitutorial.com/lessons/httpmethods.html) that may help you.

You need to remove a poem by the specified ID according to the REST concept.

Which of the answer option is more suitable for this task?

DELETE poems/12

What does REST stand for?

REpresentational State Transfer

You need to update an article on a given ID according to the concept of REST.

Which of the answer options is more suitable for this task?

PUT articles/3

You need to receive a book by a given ID according to the concept of REST.

Which of the answer options is more suitable for this task?

GET /books/4

Which of the following is NOT a principle of REST?

he correct answer based on REST principles and the context of commonly defined constraints is:

✅ **Full Idempotence**

### Explanation:

* **Full Idempotence** is **not** one of the six formal REST principles.
* The six REST principles are:
  + Client-server interaction model
  + Stateless
  + Cacheable
  + Uniform interface
  + Layered system
  + Code on demand (optional)

So, **Full Idempotence** is **not** a core REST constraint — that makes it the correct selection in a "which one doesn't belong" type of question.

What does RESTful stand for? It is a type of app that follows the REST rules

## Work of a RESTful service

 Report a typo

You are working with a service that stores articles. You need to implement the following algorithm:

* Alternately add articles with ID 1 and 2
* Get an article with ID 1
* Delete article with ID 2

Sort the HTTP methods according to the algorithm.

 Here’s the correct order of HTTP methods based on the algorithm:

1. ✅ **PUT articles/1** – Add article with ID 1
2. ✅ **PUT articles/2** – Add article with ID 2
3. ✅ **GET articles/1** – Retrieve article with ID 1
4. ✅ **DELETE articles/2** – Delete article with ID 2

What are the four basic HTTP methods used to interact with resources in the REST concept?

DELETE

GET

POST

PUT

You need to create a new resource with articles according to the REST concept.

Which of the answer options is more suitable for this task?

POST articles