Theory: REST

9 minutes 0 / 5 problems solved

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 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.

§1. Short history

In 2000, *Roy Fielding* publishes 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.

§2. 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, we improve and simplify application operation.
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 can 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.

§3. 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 in 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*

Current topic:

REST

Topic depends on:

× HTTP messages

Topic is required for:

Postman

HTTP clients

Table of contents:

Content) is returned if no content in the response body has been transmitted.

- `DELETE` method 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.

Well, that was a lot! Dizzy yet? If you want more information and clarification, check out some external resources that may help you. Feeling confident? Let's practice!