What is REST? Slightly confused [closed]

Asked 13 years, 11 months ago Modified 2 years, 7 months ago

Viewed 113k times

175

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 9 years ago.

Improve this question

I was under the assumption that REST was a web service but it seems that I am incorrect in thinking this - so, what is REST?

I've read through Wikipedia but still cant quite wrap my head around it. Why to do many places refer to API's as REST API's?

rest

Share

edited Sep 16, 2012 at 15:43

Improve this question

Follow

Bill the Lizard
405k ● 211 ● 572 ● 889

asked Jan 11, 2011 at 23:40

anon

26 @John Saunders: How is this a possible duplicate? The other guy apparently knows what REST is whereas, Nathan, on the other hand, is confused. – Fake Code Monkey Rashid Jan 12, 2011 at 0:11

1 I felt the other would answer his question. If nobody else agrees, then the close vote will age off. We have about ten answers to this question. Just click the "rest" tag and you'll see them all. – John Saunders Jan 12, 2011 at 1:05

2 REST is a set of rules for building web services. If an API is built according to those rules it is a REST API. How I explained REST to my rubber duck explains some of those rules informally. – User42 Jul 27, 2017 at 12:49

5 Answers

Sorted by: Highest score (default)

143

REST is not a specific web service but a design concept (architecture) for managing state information. The seminal paper on this was Roy Thomas Fielding's dissertation (2000), "Architectural Styles and the Design of Network-based Software Architectures" (available online from the University of California, Irvine).

First read Ryan Tomayko's post How I explained REST to my wife; it's a great starting point. Then read Fielding's actual dissertation. It's not that advanced, nor is it long

(six chapters, 180 pages)! (I know you kids in school like it short).

EDIT: I feel it's pointless to try to explain REST. It has so many concepts like scalability, visibility (stateless) etc. that the reader needs to grasp, and the best source for understanding those are the actual dissertation. It's much more than POST/GET etc.

Share Improve this answer

Follow

edited May 1, 2022 at 1:53

0Valt

10.3k ● 8 ● 39 ● 64

answered Jan 11, 2011 at 23:48

Anders

6,218 ● 4 ● 27 ● 31

@Nathan, Trust me, I had the same problem as you did before. Read the thesis, perhaps go over it a few times slowly, but you will grasp the concept, it's actually not hard at all. People just have a tendency to explain it poorly. – Anders Jan 12, 2011 at 0:01

When developers try to employ REST and try to do it only on their code (without planning the whole system with REST in mind), hell comes :-) – karatedog Jan 12, 2011 at 0:06

@Anders, Considering REST is what it is, how can you compair REST vs Web Services? – Prisoner Jan 12, 2011 at 0:30

7 got it web.archive.org/web/20080429231452/http://tomayko.com/writings/… – user961954 Jul 24, 2013 at 4:30

1 maybe this chapter will be enough instead of reading whole dissertation: ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm – andilabs Dec 6, 2013 at 0:37

85

REST is a software design pattern typically used for web applications. In layman's terms this means that it is a commonly used idea used in many different projects. It stands for **REpresentational State Transfer**. The basic idea of REST is treating objects on the server-side (as in rows in a database table) as resources than can be created or destroyed.

The most basic way of thinking about REST is as a way of formatting the URLs of your web applications. For example, if your resource was called "posts", then:

`/posts` Would be how a user would access ALL the posts, for displaying.

`/posts/:id` Would be how a user would access and view an individual post, retrieved based on their unique id.

`/posts/new` Would be how you would display a form for creating a new post.

Sending a POST request to `/users` would be how you would actually *create* a new post on the database level.

Sending a PUT request to `/users/:id` would be how you would update the attributes of a given post, again

identified by a unique id.

Sending a DELETE request to `/users/:id` would be how you would delete a given post, again identified by a unique id.

As I understand it, the REST pattern was mainly popularized (for web apps) by the Ruby on Rails framework, which puts a big emphasis on RESTful routes. I could be wrong about that though.

I may not be the most qualified to talk about it, but this is how I've learned it (specifically for Rails development).

When someone refers to a "REST api," generally what they mean is an api that uses RESTful urls for retrieving data.

Share Improve this answer

Follow

edited May 14, 2018 at 2:43

David Klempfner

9,762 ● 24 ● 86 ● 181

answered Jan 11, 2011 at 23:54

maxluzuriaga

1,327 ● 9 ● 16

2 You are lucky, because Rails is built on REST principles and if you use the Rails tools, your code will be RESTful. However there are coders out there who don't understand jack about REST, they code what they want and at the end they utilize this 'URL formatting' because it is trendy.
– karatedog Jan 12, 2011 at 0:03

10 where ever /users has been used in this answer, shouldn't it be /posts ? – Mayuresh Srivastava Jul 1, 2017 at 15:37

@MayureshSrivastava Observe that PUT examples have :id and POST examples don't have :id. Google for the rest of the story. (POST: non-safe,non-idempotent; PUT:non-safe,idempotent) – Ajeet Ganga Oct 2, 2017 at 3:36

44

`REST` is an *architectural style* and a *design* for network-based software architectures.

`REST` concepts are referred to as resources. A representation of a resource must be stateless. It is represented via some media type. Some examples of media types include `XML`, `JSON`, and `RDF`. Resources are manipulated by components. Components request and manipulate resources via a standard uniform interface. In the case of HTTP, this interface consists of standard HTTP ops e.g. `GET`, `PUT`, `POST`, `DELETE`.

`REST` is typically used over `HTTP`, primarily due to the simplicity of HTTP and its very natural mapping to RESTful principles. REST however is not tied to any specific protocol.

Fundamental REST Principles

Client-Server Communication

Client-server architectures have a very distinct separation of concerns. All applications built in the RESTful style must also be client-server in principle.

Stateless

Each client request to the server requires that its state be fully represented. The server must be able to completely understand the client request without using any server context or server session state. It follows that all state must be kept on the client. We will discuss stateless representation in more detail later.

Cacheable

Cache constraints may be used, thus enabling response data to to be marked as cacheable or not-cachable. Any data marked as cacheable may be reused as the response to the same subsequent request.

Uniform Interface

All components must interact through a single uniform interface. Because all component interaction occurs via this interface, interaction with different services is very simple. The interface is the same! This also means that implementation changes can be made in isolation. Such changes, will not affect fundamental component interaction because the uniform interface is always unchanged. One disadvantage is that you are stuck with the interface. If an optimization could be provided to a specific service by changing the interface, you are out of luck as REST prohibits this. On the bright side, however, REST is optimized for the web, hence incredible popularity of REST over HTTP!

The above concepts represent defining characteristics of REST and differentiate the REST architecture from other architectures like web services. It is useful to note that a REST service is a web service, but a web service is not necessarily a REST service.

See this blog post on **REST Design Principals** for more details on **REST** and the above principles.

Share Improve this answer

Follow

edited Oct 27, 2014 at 11:31

Parvej Solkar

13 • 2

answered Mar 13, 2014 at 2:39

cmd

11.8k • 8 • 52 • 63

17

It stands for Representational State Transfer and it can mean a lot of things, but usually when you are talking about APIs and applications, you are talking about REST as a way to do web services or get programs to talk over the web.

REST is basically a way of communicating between systems and does much of what SOAP RPC was designed to do, but while SOAP generally makes a connection, authenticates and then does stuff over that connection, REST works pretty much the same way that that the web works. You have a URL and when you request that URL you get something back. This is where things start getting confusing because people describe

the web as a the largest REST application and while this is technically correct it doesn't really help explain what it is.

In a nutshell, REST allows you to get two applications talking over the Internet using tools that are similar to what a web browser uses. This is much simpler than SOAP and a lot of what REST does is says, "Hey, things don't have to be so complex."

Worth reading:

- How I Explained REST to My Wife (now available here)
- Architectural Styles and the Design of Network-based Software Architectures

Share Improve this answer

Follow

edited Jul 3, 2015 at 3:54

ChickenWing24

205 ● 3 ● 12

answered Jan 11, 2011 at 23:57

Mark

1,238 ● 1 ● 10 ● 18

REST is an architecture based upon constraints, SOAP is a protocol, those are completely different things. I don't like when people are talking about SOAP and REST in the same concept, it's no wonder people get confused about it.
– Anders Jan 12, 2011 at 0:04

@Anders - He said he was looking at a REST API and thought it was a way to use Webservices. You can use REST like that and in that capacity it accomplishes much of what

SOAP accomplishes. It is also possible to talk about the web as the world's largest RESTful application, but that doesn't really explain what you'd use a REST API for. – Mark Jan 12, 2011 at 0:10

This has actually been my main problem, seeing REST and SOAP in the same concept. It seems API's are just RESTful in design. – Prisoner Jan 12, 2011 at 0:13

4

http://en.wikipedia.org/wiki/Representational_State_Transfer

The basic idea is that instead of having an ongoing connection to the server, you make a request, get some data, show that to a user, but maybe not all of it, and then when the user does something which calls for more data, or to pass some up to the server, the client initiates a change to a new state.

Share Improve this answer Follow

edited Jan 11, 2011 at 23:51

answered Jan 11, 2011 at 23:45

Hack Saw

2,781 ● 1 ● 20 ● 37

4 Maybe it's just me but I like seeing the relevant answer on stackoverflow along with the appropriate citation. Just humor me and suppose wikipedia went poof. What good does your link do then hmm? :) – Fake Code Monkey Rashid Jan 11, 2011 at 23:47

1 @Hack Saw: Shouldn't that be in your answer? – Fake Code Monkey Rashid Jan 11, 2011 at 23:50

I just noticed the edit feature. :) – Hack Saw Jan 11, 2011 at 23:52

1 @karatedog: REST can be achieved with HTTP, but it could be done with a variety of data transfer methods. – Hack Saw Jan 11, 2011 at 23:54

1 This is the HTTP protocol you are writing about, REST is an architecture. – karatedog Jan 11, 2011 at 23:55