What is the difference between HTTP and REST?

Ask Question

238

133

After reading a lot about the differences between REST and SOAP, I got the impression that REST is just another word for HTTP. Can someone explain what functionality REST adds to HTTP?

Note: I'm not looking for a comparison of REST versus SOAP.

Update: Thanks for your answers. Now it has become clear to me that REST is just a set of rules about how to use HTTP. Hence I posted a follow-up about what the advantages of these conventions are .

Note: I now grasp the meaning of REST; as Emil Ivanov remarks, REST means using HTTP the way it's meant to be. However, I'm not sure whether this deserves a term of its own, and I certainly http rest

edited May 23 '17 at 12:26

Community ♦
1 1

asked Feb 3 '10 at 9:20

Dimitri C.
11.2k 18 74 98

36 Just as a side note, probably 90% of the hype that you hear about REST these days are from people who don't actually understand the complete picture about REST. REST unfortunately has become a sales buzzword. You have to cut through a lot of crap to find out the real benefits. – Darrel Miller Feb 3 '10 at 13:18

7 The hype around REST is probably due to people being heavily annoyed by SOAP. Everybody's just happy to escape the SOAP hell :D – aefxx Feb 3 '10 at 13:31

I'm the newbie coder at work here, and SOAP issues and moving away from it is how I ended up here. Thanks for the verification it is indeed HTTP. I was also confused. – kyle Nov 20 '13 at 21:47

13 Think of HTTP as a ball to play games with and REST as a

does it deserve it's own term? Because calling all ball games, "ball game" means there's no way of determining which rule-set you are using. This way, everyone is reading from the same song sheet (sorry, mixed metaphor) – Ross Drew Oct 29 '15 at 16:26

Now we have another option GraphQL compared with REST. Both are using HTTP. – Hongbo Miao May 1 '17 at 21:15

13 Answers

177

No, **REST** is the way **HTTP** should be *used*.

Today we only use a tiny bit of the HTTP protocol's methods – namely `GET` and `POST`. The REST way to do it is to use all of the protocol's methods.

For example, REST dictates the usage of `DELETE` to erase a document (be it a file, state, etc.) behind a URI, whereas, with HTTP, you would misuse a `GET` or `POST` query like `...product/?delete_id=22`.

6,959 9 42 53

answered Feb 3 '10 at 9:25

aefxx
18.8k 5 34 51

19 And what would be the big advantage of using those other methods? – Dimitri C. Feb 3 '10 at 9:26

6 I posted a link to a real world example that would show you the advantages. Cheers. – aefxx Feb 3 '10 at 9:30

7 +1 for understanding the *meaning* of the OP's question. – Withheld Dec 24 '12 at 14:48

6 -1 for giving wrong definition to rest. rest is a type of architecture, not a way to send messages via web. for more information: en.wikipedia.org/wiki/Representational_state_transfer – Yuval Perelman Jan 18 '16 at 16:06

3 @aefxx thank you, i didnt know that, and never read the full dissertation. i would change the votedown to voteup if it wasnt locked. you have an interesting way of debating, you could just give me a link and be done with that. shish. – Yuval Perelman

70

HTTP is a protocol used for communication, usually used to communicate with internet resources or any application with a web browser client.

REST means that the main concept you are using while designing the application is the Resource: for each action you want to perform you need to define a resource on which you usually do only CRUD operation, which is a simple task. for that its very convenient to use 4 verbs used in HTTP protocol against the 4 CRUD operations (Get for Read, POST is for CREATE, PUT is for UPDATE and DELETE is for DELETE). that's unlike the older concept of RPC (Remote Procedure Call), in which you have a set of actions you want to perform as a result of the user's call. if you think for example on how to describe a facebook like on a post, with RPC you might create services called with all your other services related to FB posts, thus you won't need to create special object for Like. with REST you will have a Like object which will be managed separately with Delete and Create functions. It also means it will describe a separate entity in your db. that might look like a small difference, but working like that would usually yield a much simpler code and a much simpler application. with that design, most of the app's logic is obvious from the object's structure (model), unlike RPC with which you would usually have to explicitly add a lot more logic.

designing RESTful application is usually a lot harder because it requires you to describe complicated things in a simple manner. describing all functionalities using only CRUD functions is tricky, but after doing that your life would be a lot simpler and you will find that you will write a lot shorter methods.

One more restraint REST architecture present is not to use information needs to understand who is the client and what he wants is passed with the web message. each call to a function is self descriptive, there is no previous conversation with the client which can be referenced in the message. therefor a client could not tell you "give me the next page" since you don't have a session to store what is the previous page and what kind of page you want, the client would have to say "my name is yuval, get me page 2 of a specific post in a specific forum". that means a bit more data would have to transfer in the communication, but think of the difference between finding a bug reported from the "get me next page" function in oppose to "get me page 2 of question id 2190836 in stack overflow".

Of course there is a lot more to it, but to my opinion that's the main concepts in a teaspoon.

edited May 9 '17 at 9:48

answered Sep 26 '15 at 11:38

Detailed and good explanation to what REST actually is in a nutshell. – LogixMaster Nov 6 '15 at 8:53

1 +1 hands-on and if I could, I'd add +1 for the creative English too. (I mean that nicely and with a friendly smile)... Seriously, "in a teaspoon" yr answer was concrete and good. Tx. – Cbhihe Apr 14 '17 at 21:05

1 This post should be the answer. – Steven Zack Sep 27 '17 at 16:27

50

REST doesn't add any specific functionality to HTTP but is an architectural style that was developed alongside HTTP and most commonly uses HTTP for its application layer protocol.

answered Feb 3 '10 at 9:26

Mark
24.1k 4 47 83

5 What does "architectural style" mean? – Dimitri C. Feb 3 '10 at 9:31

10 The architectural style define the guiding principles behind a given application. It is not strongly tied to a

application is composed. How many modules you use. How they interact each other. Type of message exchanged. – Massimo Fazzolari Feb 3 '10 at 9:36

An architectural style would be a common way of structuring a software system. See en.wikipedia.org/wiki/… for examples of architectural styles. – Mark Feb 3 '10 at 9:37

1 From Roy Fielding's dissertation sec 1.5: *"An architectural style is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style."* I keep remembering it just as 'an architectural style is a set of constraints'. – icc97 Mar 23 '18 at 8:47

25

HTTP is an application protocol. REST is a set of rules, that when followed, enable you to build a distributed application that has a

If you are looking for the most significant constraints of REST that distinguish a RESTful application from just any HTTP application, I would say the "self-description" constraint and the hypermedia constraint (aka Hypermedia as the Engine of Application State (HATEOAS)) are the most important.

The self-description constraint requires a RESTful request to be completely self descriptive in the users intent. This allows intermediaries (proxies and caches) to act on the message safely.

The HATEOAS constraint is about turning your application into a web of links where the client's current state is based on its place in that web. It is a tricky concept and requires more time to explain than I have right now.

answered Feb 3 '10 at 12:30

Darrel Miller
113k 27 169 224

13

Not quite...

http://en.wikipedia.org/

> REST was initially described in the context of HTTP, but is not limited to that protocol. RESTful architectures can be based on other Application Layer protocols if they already provide a rich and uniform vocabulary for applications based on the transfer of meaningful representational state. RESTful applications maximise the use of the pre-existing, well-defined interface and other built-in capabilities provided by the chosen network protocol, and minimise the addition of new application-specific features on top of it.

http://www.looselycoupled.com/glossary/SOAP

> (Simple Object Access Protocol) The standard for web services messages. Based on XML, SOAP defines an envelope format and various rules for describing its

> the three foundation standards of web services, it is the preferred protocol for exchanging web services, but by no means the only one; proponents of REST say that it adds unnecessary complexity.

edited Feb 3 '10 at 16:46
Darrel Miller
113k 27 169 224

answered Feb 3 '10 at 9:22
LiamB
11.2k 17 66 107

3 Who said anything about SOAP? – Darrel Miller Feb 3 '10 at 13:41

9 The guy who asked the question...."After reading a lot about the differences between REST and SOAP" – LiamB Feb 3 '10 at 14:37

2 My bad, I guess I needed more coffee this morning. Downvote removed. – Darrel Miller Feb 3 '10 at 16:46

10

As I understand it, REST enforces the use of the available HTTP commands as they were meant to be used.

```
GET
http://example.com?metl
```

But with rest I would use the "DELETE" request method, removing the need for the "method" query param

```
DELETE
http://example.com?iter
```

answered Mar 24 '15 at 17:03

Dss
1,036 1 16 24

7

REST is a specific way of approaching the design of big systems (like the web).

It's a set of 'rules' (or 'constraints').

HTTP is a protocol that tries to obey those rules.

answered Feb 3 '10 at 16:12

Mike
3,167 1 12 14

I'd say that if you use HTTP as a transport for your REST service it's easy to obey those rules. – abatishchev Feb 20 '14 at 23:54

HTTP is a communications

is a protocol to exchange XML-based messages that can use HTTP to transport those messages. Rest is a protocol to exchange any(XML or JSON) messages that can use HTTP to transport those messages.

answered Aug 16 '15 at 14:07
vamsi
51 1

Your answer does not answer the question. – Anix PasBesoin Aug 22 '15 at 8:37

Your HTTP and SOAP definition was great and cleared up the question for me. But I do not believe Rest is a protocol. It is an architectural guideline which enforces the correct use of the HTTP transport protocol. – CapturedTree Apr 25 '18 at 21:46

3

REST is not necessarily tied to **HTTP**. RESTful web services are just web services that follow a RESTful architecture.

```
What is Rest -
1- Client-server
2- Stateless
3- Cacheable
4- Layered system
5- Code on demand
```

Rahul Patel
771 8 11

HTTP is `1-Client-server` , `2-stateless` , `3-casheable` . Then What extra features/constraints REST put on HTTP? What can we do with REST that cannot be done with HTTP alone? – Wafeeq Nov 29 '16 at 13:28

2

REST = Representational State Transfer

REST is a set of rules, that when followed, enable you to build a distributed application that has a specific set of desirable constraints.

REST is a protocol to exchange any(XML, JSON etc) messages that can use HTTP to transport those messages.

Features:

It is stateless which means that ideally no connection should be maintained between the client and server. It is the responsibility of the client to pass its context to the server and then the server can store this context is identified by session identifier passed by the client.

Advantages of Statelessness:

1. Web Services can treat each method calls separately.
2. Web Services need not maintain the client's previous interaction.
3. This in turn simplifies application design.
4. HTTP is itself a stateless protocol unlike TCP and thus RESTful Web Services work seamlessly with the HTTP protocols.

Disadvantages of Statelessness:

1. One extra layer in the form of heading needs to be added to every request to preserve the client's state.
2. For security we need to add a header info to every request.

HTTP Methods supported by REST:

GET:

the same results every time a call is made

PUT: Same like GET. Idempotent and is used to update resources.

POST: should contain a url and body Used for creating resources. Multiple calls should ideally return different results and should create multiple products.

DELETE: Used to delete resources on the server.

HEAD:

The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response. The meta information contained in the HTTP headers in response to a HEAD request SHOULD be identical to the information sent in response to a GET request.

OPTIONS:

This method allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource

Go here for all the responses.

Here are a few important ones: 200 - OK 3XX - Additional information needed from the client and url redirection 400 - Bad request
401 - Unauthorized to access
403 - Forbidden
The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort.

404 - Not Found
The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.

405 - Method Not Allowed A request method is not supported for the requested resource; for example, a GET request on a form that requires data to be presented via POST, or a PUT request on a read-only resource.

404 - Request not found
500 - Internal Server Failure
502 - Bad Gateway

answered Jun 24 '17 at 0:28

Pritam Banerjee
10.9k 6 44 67

1

From You don't know the difference between HTTP and REST

> So REST architecture and HTTP 1.1 protocol are independent from each other, but the HTTP 1.1 protocol was built to be the ideal protocol to follow the principles and constraints of REST. One way to look at the relationship between HTTP and REST is, that REST is the design, and HTTP 1.1 is an implementation of that design.

answered Oct 15 '18 at 5:12

Farsan Rashid
690 9 22

0

REST APIs must be hypertext-driven

From Roy Fielding's blog here's a set of ways to check if you're building a HTTP API

> API designers, please note the following rules before calling your creation a REST API:
>
> - A REST API should not be dependent on any single communicatio n protocol, though its successful mapping to a given protocol may be dependent on the availability of metadata, choice of methods, etc. In general, any protocol element that uses a URI for identification must allow any URI scheme to be used for the sake of that identification. [Failure here implies that identification is not separated from interaction.]
> - A REST API should not contain any changes to the communicatio n protocols aside from

> underspecified bits of standard protocols, such as HTTP’s PATCH method or Link header field. Workarounds for broken implementations (such as those browsers stupid enough to believe that HTML defines HTTP’s method set) should be defined separately, or at least in appendices, with an expectation that the workaround will eventually be obsolete. [Failure here implies that the resource interfaces are object-specific, not generic.]
>
> - A REST API should spend almost all of its descriptive effort in defining the media type(s) used for representing resources and

> extended relation names and/or hypertext-enabled mark-up for existing standard media types. Any effort spent describing what methods to use on what URIs of interest should be entirely defined within the scope of the processing rules for a media type (and, in most cases, already defined by existing media types). [Failure here implies that out-of-band information is driving interaction instead of hypertext.]
>
> - A REST API must not define fixed resource names or hierarchies (an obvious coupling of client and server). Servers must have the freedom to

> servers to instruct clients on how to construct appropriate URIs, such as is done in HTML forms and URI templates, by defining those instructions within media types and link relations. [Failure here implies that clients are assuming a resource structure due to out-of band information, such as a domain-specific standard, which is the data-oriented equivalent to RPC's functional coupling].
>
> - A REST API should never have "typed" resources that are significant to the client. Specification authors may use resource types for describing server implementation behind the

invisible to the client. The only types that are significant to a client are the current representation's media type and standardized relation names. [ditto]

- A REST API should be entered with no prior knowledge beyond the initial URI (bookmark) and set of standardized media types that are appropriate for the intended audience (i.e., expected to be understood by any client that might use the API). From that point on, all application state transitions must be driven by client selection of server-provided choices that are present in the received representations or implied by the user's

> transitions may be determined (or limited by) the client's knowledge of media types and resource communication mechanisms, both of which may be improved on-the-fly (e.g., code-on-demand). [Failure here implies that out-of-band information is driving interaction instead of hypertext.]

answered Mar 23 '18 at 9:02

icc97
6,507 5 40 62

0

`HTTP is a contract, a communication protocol and REST is a concept, an architectural style` which may use HTTP, FTP or other communication protocols but is widely used with HTTP.

`REST implies a series of constraints about how Server and Client should interact . HTTP is a communication` used in REST API just because `REST was inspired by WWW (world wide web) which largely used HTTP` before REST was defined, so it's easier to implement REST API style with HTTP.

```
There are three major 
```

`1.` Interaction between server and client should be described via hypertext only.

`2.` Server and client should be loosely coupled and make no assumptions about each other. Client should only know resource entry point. Interaction data should be provided by the server in the response.

`3.` Server shouldn't store any information about request context. Requests must be independent and idempotent (means if same request is repeated infinitely, exactly same result is retrieved)

And HTTP is just a communication protocol (a tool) that can help to achieve this.

For more info check these links:

aturityModel.html
http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

edited Jun 1 '18 at 19:22

answered Jun 1 '18 at 17:30

Daniel
105 2 10

protected by cassiomolin Feb 27 at 14:43

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?