Is REST the New SOAP?

Leia em PortuguÃªs

This item in japanese

Almost a decade ago there was a flurry of activity around REST and SOAP based systems. Several authors wrote about the pros and cons of one or the other, or when you should consider using one instead of the other. However, with much attention moving away from SOAP-based Web Services to REST and HTTP, the arguments and discussions died down and many SOA practitioners adopted REST (or plain HTTP) as the basis for their distributed systems. However, recently Pakal De Bonchamp wrote an article called "REST is the new SOAP" in which he compares using REST to "a testimony to insanity".

His article is long and detailed but some of the key points he makes include the complexity, in his view, of simply exposing a straightforward API which could be done via an RPC mechanism in "a few hours" yet with REST can take much longer. Why? Because:

> *No more standards, no more precise specifications. Just a vague "RESTful philosophy", prone to endless metaphysical debates, and as many ugly workarounds.*

The mapping of the methods to CRUD operations is not straightforward. How do you determine whether and when to create new resource instances rather than re-using an existing resource? HTTP error codes are limited, in his view, and impose problems when trying to express richer error situations. And his list goes on. The end result?

> *And you're gone for hours, reinventing the wheel. Not even a tailored, smart wheel. A broken and fragile wheel, requiring tons of documentation to be understood, and violating specifications without even knowing it.*

He dives into a lot of detail on specific HTTP verbs, including PUT and rarely discussed PATCH and DELETE:

> *You want to use PUT to update your resource? OK, but some Holy Specifications state that the data input has to be equivalent to the representation received via a GET. So what do you do with the numerous read-only parameters returned by GET (creation time, last update time, server-generated token…)? You omit them and violate the PUT principles? You include them anyway, and expect an "HTTP 409 Conflict" if they don't match server-side values (forcing you to then issue a GET…)? You give them random values and expect servers to ignore them (the joy of silent errors)? Pick your poison, REST clearly has no clue what a read-only attribute it, and this won't be fixed anytime soon. Meanwhile, a GET is dangerously supposed to return the password (or credit card number) which was sent in a previous POST/PUT; good luck dealing with such write-only parameters too. Did I forget to mention that PUT also brings dangerous race conditions, where several clients will override each other's changes, whereas they just wanted to update different fields?*

Error handling, REST architectural concepts and many other aspects get a detailed review and don't come away well. For both REST supporters and those who aren't convinced, the article is worth reading. He concludes with the following:

> *Almost-transparent remote procedure call was what 99% people really needed, and existing protocols, as imperfect as they were, did the job just fine. This mass monomania for the lowest common denominator of the web, HTTP, has mainly resulted in a huge waste of time and grey matter. REST promised simplicity and delivered complexity.*
> *REST promised robustness and delivered fragility.*
> *REST promised interoperability and delivered heterogeneity.*
> *REST is the new SOAP.*

Reading through the many comments on the article it is clear some people agree with him but the majority disagree, pointing out numerous flaws in his argument. For example, Filippos Vasilakis states:

> *Maybe you should take a look on Roy's thesis. What you are attacking here is the common misconception of REST. As Roy noted in his thesis REST might not fit all cases, but it fits pretty good the case where you can't control the client. So, if you know any other model that is self-descriptive and thus evolvable let us know. Because REST is all about that. Talking to devices/clients/hardware that we don't/can't control (like mobile app in apple store which needs 10 days to deploy a new change, if your update does not get rejected by Apple, sensor devices that you can't even access/update etc). For those scenarios we have REST and then comes GraphQL with many limitations and issues. But if you come up with another model that solves the issue of evolvability please let us know. RPC is not one of them.*

Another comment, this time from a Vlad Ko:

> *Ouch. What did I just read? You are complaining about taking your time to architect a decent API? I believe that's your responsibility and a goal as a developer to ensure that. Complaining about every REST-related or "unrelated" issue under the sun is meaningless and kind of childish. Every language, protocol, specification and concept has problems, bugs, illogical syntax, slow VM's, lack of type, too strict, too loose, too functional, not enough OOP.Welcome to the world of software engineering.*

And from Christopher Patti:

> *This is an excellent article. It has the punch of a rant but makes its case in a well reasoned detailed way with a lot of supporting evidence. There's one factor you don't discuss though—tooling. People gave up on the paragon of logical purity that SOAP is or was if your article is to be believed, because if you weren't doing your work in either Java or .NET the tooling was really, *really* awful. I'm told it's gotten better, but when people feel pain they react to that pain by adopting new tools. I get that your thesis is that we may have chosen an unfortunate path and should move on, but to what? You cite other more modern protocols, but I'd be curious about specific examples you can cite that could work as a direct replacement for either SOAP or REST.*

As well as the comments, the original article promoted a lot of activity on various social media platforms. Eventually Phil Sturgeon, platform engineer at WeWork, wrote an article called simply "A Response to REST is the new SOAP" because apparently many people had asked him to refute claims made in the original article.

> *The entire article is full of common misunderstandings about REST and HTTP. Despite dedicating my career to trying to educate people through these confusions, they continue to be rife. Clearly I am not being loud enough, writing effectively enough, or doing a good enough job. That is the frustration you might hear in my writing, but nothing is aimed at the author.*

As with the initial article, this response is long and Sturgeon dissects the problems with REST which De Bonchamp had discussed in lots of detail. For example, to De Bonchamps's reference about building "a broken and fragile wheel", Sturgeon has this to say:

> *Ok this is frustrating. REST APIs are often mocked because the proponents explain that you should not need documentation. A REST API absolutely should not need documentation, but I have spent the last few months working on generating documentation for our APIs, because they are all RPC APIs. When an API represents its own state, uses hypermedia to declare its affordances, and provides a contract, you can chose to generate human readable documentation, but that's only going there for people treating the REST API like a RPC API… A REST API quite definitively requires less documentation, unless you've just built an unspecified RPC which is pretending to be REST, like so many people do.*

To De Bonchamps's discussion on the perils of PUT et al, "*This sounds like a frustration born from not understanding the purpose of PUT.*" The dissection and responses from Sturgeon go on and on. In fact even without reading the original article, Sturgeon's response is worth reading on its own merits. But the general underlying thread throughout is that much, if not all, of De Bonchamp's complaints around REST are due to his misunderstandings. Sturgeon concludes with:

> *I get it, REST is a complex topic. Too many people think they understand it, and are falsely validated when they bump into other people who don't understand it. Folks everywhere are building RESTish APIs which are basically just RPC + HTTP verbs + Pretty URLs, and as that doesn't seem very helpful they write giant articles explaining why that's not very useful…*

And that might have been that. However, it seems that Sturgeon and De Bonchamp have continued to have discussions and Sturgeon has written a further update.

> *I'm still having a productive dialog with the author, helping him understand how REST works. I thought it might interest some of you too.*

Hopefully the interested reader will check it out and see how the conversation is evolving.