SOAP is officially dead, long live REST

Tags: PYTHON

When someone mentions “web services”, most of the people in my environment still automatically think SOAP instead of REST. Their perception, if they even know REST at all, is that REST is something simple (and perhaps pure) and that SOAP is what everyone is using and should be using for serious business.

At least, it is that way in *my* experience.

SOAP, the way I explain it, does most/all of the things that you can do with generic http/https/urls/files, only embedded in an xml document. So instead of a http GET request to `https://some.server/some/thing/orders.xml`, you’d have half a megabyte of XML telling some SOAP server at the other end what to do and how to do it and how to decrypt the embedded binary junk in some way.

So: just use generic http(s) GET/POST(/DELETE/PUT) calls to URLs to download/update/delete/change bits and pieces of json or xml. Preferrably with links inside them to further actions, just like you’d do in a regular webpage. And give everything its own URL. That’s more or less what REST is.

Yesterday’s announcement was that SOAP’s (or rather the whole huge whopping big so-called “WS-i” stack’s) organization would stop and be folded (in maintenance mode) into an existing standards organisation. **Many took that to be the official end of the whole SOAP idea**, for instance Simon Phipps.

Two choice quotes from his article:

All the work of WS-I was clearly doomed to lead a webless existence as part of the complexity big vendors encourage their corporate customers to use internally, ironically leading them to be locked-in to the tools the vendors supply to mitigate the complexity.

and:

It took many, many years for that doom to be made reality. In the course of a decade of corporate politics in smoke-filled rooms, many fine and talented corporate *standards engineers have spent countless hours perfecting a set of specifications that are expertly crafted and logically complete. Fine work, and many lessons learned, but sadly irrelevant to most of us. Goodbye, WS-I. I know and respect many of your participants, but I won't mourn your passing.*

To celebrate the passing-away, here's a photo of **soap bubbles** on a medieval market in Utrecht:

Soap bubbles on a medieval market in Utrecht

Comments for this thread are now closed ×

15 Comments **Reinout van Rees** **Disqus' Privacy Policy** 1 **Login**

Recommend 8 Tweet Share **Sort by Best**

Martin De Wulf • 11 years ago

I will not mourn WS-I neither. Nice article.

1 | • Share ›

Rajeev J • 10 years ago

There is a reason still why would you want to use SOAP instead of REST. The reason is exchange of complicated data structures. If your clients would like to work with the data structures used by your services (A tight coupling between client and server and exchange via those DS) then SOAP maybe a better choice. For read only cases or exposing data to world cases REST is a better choice.

| • Share ›

Jean-Pierre Duval-Durandeau • 11 years ago

Does SAP have REST support? Nope. Then SOAP is far from being dead.

| • Share ›

Reinout van Rees Mod → Jean-Pierre Duval-Durandeau • 11 years ago

Well, it might take a while for the corpse to stop twitching :-)

I'm surprised that SAP doesn't have REST support. On the other hand, I had to integrate one company's SAP-emitted SOAP messages with another company's inventory system. That took at least three SAP consultants on the other side to get it all working. On my side, I was the only one. It was inflexible as hell. And all the error handling and exceptions had to be handled on my side of the code, as SAP (through SOAP) wasn't capable of handling an "out of stock" response: too much work to get it working inside SAP.

| • Share ›

Hao • 11 years ago

Soap is very tightly coupled with your service layer. In case of WSDL changes, all your client may have to recreate skeleton objects, you never known what u will break. it's hard to scale. One WSDL bundles many operations, if one of the operation changes, for example, adding a

new parameter, then the WSDL changes, the client have to change also. I have seen a logistic company use ugly versions to control different WSDL for different customers.
Also it creates lots of unnecessary xml messages back and forth.
The good thing of SOAP is, at least now it supported very well by tools, security, policy, coordination etc. This is definitely better than xml rpc or POX.

REST is build on the resource level, and it is client responsibility to munipulate resources, therefore it is loose coupled.
But in the long term run, REST will be the next gerneration of webservice.

^ | ⌄ • Share ›

Aboukirev • 11 years ago

One of the most important parts of SOAP is discover-ability. With REST you have to know exact URLs and parameters that it accepts beforehand. And you are prone to a lot of mistakes. There will also be a hassle of URL encoding (mitigated by libraries) and proper interpretation/casting and validation of passed values. SOAP does parameters marshaling for you.

^ | ⌄ • Share ›

Reinout van Rees Mod ➦ Aboukirev • 11 years ago

With SOAP you *can* have discoverability with WSDL. Often, for me, it was missing.

And in the end, your own code has to deal with whatever it receives, discoverable in WSDL or not. Your code won't write itself. To me, WSDL is just a different kind of API documentation

And: for WSDL you also have to know the exact URL of the WSDL :-) And your code has to work with the parameters. And you have the hassle of re-doing most of http inside a SOAP message envelope.

With a proper REST API you'll have to know one starting point URL and it will contain links ("hrefs") to the URLs of the other actions you can do. GET my photo site information. Find the link to the category listing. GET that. GET one of the linked categories. POST a new photo there. Etc.

^ | ⌄ • Share ›

Łukasz Ka • 11 years ago

For next 10 years SOAP will be surely in use, so libraries should be created for it. It cannot be dropped from day to day.

And I have a question: what are libraries for REST in Python, PHP and Java? Can you recommend some?

^ | ⌄ • Share ›

Reinout van Rees Mod ➦ Łukasz Ka • 11 years ago

PHP and java: no idea. But the good thing is that, when consuming someone else's REST web service, anything will basically do. So python's urllib/urllib2 will happily download everything over http that you want. GET and POST requests are no problem in any language. Once you get to http PUT and DELETE, you can run into library limitations: these two http methods are uses way less often than GET/PUT.

On the producer side, so on the webserver, it all depends on the framework you're using. Django, for instance, has a "piston" library that's supposed to be pretty good.

1 ^ | ˅ • Share ›

Łukasz Ka ➦ Reinout van Rees • 11 years ago

I know, I can habdle everything with sockets ;) by this is not the problem.

Thanks for piston.

^ | ˅ • Share ›

lifewithryan ➦ Łukasz Ka • 11 years ago

Yeah, not sure you actually NEED any sort of library to consume rest. You just use HTTP as it was originally intended. There are libraries that help you produce RESTful webservice but consuming them is as simple as making an HTTP GET call.

Having said that, depending on what gets returned from that call, you may need a library to interpret it, (e.g. JSON, XML, plain text, html even -- imagine that *wink*).

^ | ˅ • Share ›

Matthew Pitts ➦ lifewithryan • 10 years ago

Yeah, and that part is the pain in the ass. I personally think that a lot of programmers (not me) are turned off to REST because of the final part of parsing the xml and incorporating it into something more useful than a list of values. Working with raw XML is a nightmare (IMO). To make matters worse, some web services are not even returning XML anymore, only JSON objects. (Some newbie turns to me and says WTF is JSON?) Just like I said WTF is REST!

With SOAP/WSDL, there was a roadmap that helped you determine what types were being returned from the service. Also, vendor support is amazing for consumption of SOAP services (bypassing the complexity of working with raw XML and instead working with strongly typed data). Even the purest of RESTafarians would agree that WSDL is useful (hence the support for WADL as a possible standard for REST).

Still, REST is awesome. It is conceptually simple. Mountains of data are within easy reach. But for those of us who have been mostly shielded from the complexity of working directly with XML, it can be quite a learning curve. I much prefer programming against strongly typed data.

1 ^ | ˅ 1 • Share ›

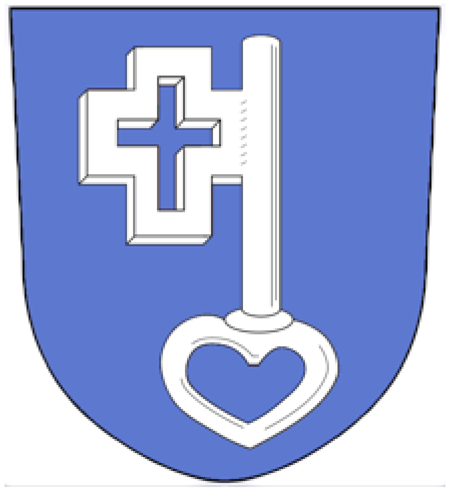
Łukasz Ka ➦ lifewithryan • 11 years ago

Of course there is a need for high level libraries, it is so obvious I will not explain.

^ | ˅ • Share ›

Sombriks ➦ Łukasz Ka • 11 years ago

for java there's Jersey:



About me

My name is Reinout van Rees and I work a lot with Python (programming language) and Django (website framework). I live in The Netherlands and I'm happily married to Annie van Rees-Kooiman.

Weblog feeds

Most of my website content is in my weblog. You can keep up to date by subscribing to the automatic feeds (for instance with Google reader):

- Atom feed of my full weblog
- Atom feed, filtered to just the Python/Django content.