REST Vs SOAP, The Difference Between Soap And Rest

Someone asked me a question today “Why would anyone choose SOAP (Simple Object Access Protocol) instead of REST (Representational State Transfer)?” My response: “The general rule of thumb I’ve always heard is ‘Unless you have a definitive reason to use SOAP use REST’”. He asked “what’s one reason?”

I thought about it for a minute and honestly answered that I haven't ever come across a reason. My background is building great internet companies.

While he seemed satisfied, I wasn't very happy with that answer, I did some homework and here's my summary on REST versus SOAP, the difference between SOAP and REST and why anyone would choose SOAP. As usual, with competing technologies both have value, the challenge is to know when to use each one (spoiler: luckily the answer is almost always REST).

I'm clearly boiling down a somewhat so please don't flame me for simplifying things, but feel free to provide any corrections you feel are necessary.

Definitions

REST

RESTs sweet spot is when you are exposing a public API over the internet to handle CRUD operations on data. REST is focused on accessing named resources through a single consistent interface.

SOAP

SOAP brings it's own protocol and focuses on exposing pieces of application logic (not data) as services. SOAP exposes operations. SOAP is focused on accessing named operations, each implement some business logic through different interfaces.

Though SOAP is commonly referred to as "web services" this is a misnomer. SOAP has very little if anything to do with the Web. REST provides true "Web services" based on URIs and HTTP.

By way of illustration here are few calls and their appropriate home with commentary.

```
getUser(User);
```

This is a rest operation as you are accessing a resource (data).

```
switchCategory(User, OldCategory, NewCategory)
```

This is a SOAP operation as you are performing an operation.

Yes, either could be done in either SOAP or REST. The purpose is to illustrate the conceptual difference.

Why REST?

Here are a few reasons why REST is almost always the right answer.

Since REST uses standard HTTP it is much simpler in just about ever way. Creating clients, developing APIs, the documentation is much easier to understand and there aren't very many things that REST doesn't do easier/better than SOAP.

REST permits many different data formats where as SOAP only permits XML. While this may seem like it adds complexity to REST because you need to handle multiple formats, in my experience it has actually been quite beneficial. JSON usually is a better fit for data and parses much faster. REST allows better support for browser clients due to it's support for JSON.

REST has better performance and scalability. REST reads can be cached, SOAP based reads cannot be cached.

It's a bad argument (by authority), but it's worth mentioning that Yahoo uses REST

for all their services including Flickr and del.icio.us. Amazon and Ebay provide both though Amazon's internal usage is nearly all REST source . Google used to provide only SOAP for all their services, but in 2006 they deprecated in favor of REST source . It's interesting how there has been an internal battle between rest vs soap at amazon. For the most part REST dominates their architecture for web services.

Why SOAP?

Here are a few reasons you may want to use SOAP.

WS-Security

While SOAP supports SSL (just like REST) it also supports WS-Security which adds some enterprise security features. Supports identity through intermediaries, not just point to point (SSL). It also provides a standard implementation of data integrity and data privacy. Calling it "Enterprise" isn't to say it's more secure, it simply supports some security tools that typical internet services have no need for, in fact they are really only needed in a few "enterprise" scenarios.

WS-AtomicTransaction

Need ACID Transactions over a service, you're going to need SOAP. While REST supports transactions, it isn't as comprehensive and isn't ACID compliant. Fortunately ACID transactions almost never make sense over the internet. REST is limited by HTTP itself which can't provide two-phase commit across distributed transactional resources, but SOAP can. Internet apps generally don't need this level of transactional reliability, enterprise apps sometimes do.

WS-ReliableMessaging

Rest doesn't have a standard messaging system and expects clients to deal with communication failures by retrying. SOAP has successful/retry logic built in and

provides end-to-end reliability even through SOAP intermediaries.

Summary

In Summary, SOAP is clearly useful, and important. For instance, if I was writing an iPhone application to interface with my bank I would definitely need to use SOAP. All three features above are required for banking transactions. For example, if I was transferring money from one account to the other, I would need to be certain that it completed. Retrying it could be catastrophic if it succeed the first time, but the response failed.

External Resources

- Why Soap Sucks
- SOAP or REST? it's about your priorities!
- Goodbye, Google SOAP search API
- SOAP vs REST

Fri Jan 15, 2010
900 Words

Read In About 4 Min

Architecture

Development

Systems

#api #remote procedure call #representational state transfer #rest #RPC #soap #software architecture #web service #web services #web standards #ws security

Using the right keys

My Online Business Card (Vcard)

About The Author:

Steve Francia has been responsible for two of the largest commercial open source projects as the current Chief Operator of the Docker Project and the former Chief Developer Advocate of MongoDB. Steve has also run some of the top community-based open source projects: spf13-vim, Hugo, Cobra & Viper

He loves open source and is thrilled to be able to work on it full-time and then some. In writing, Steve tweets as @spf13, codes on GitHub, blogs at spf13.com, connects on LinkedIn, ocassionally posts on Google+ and has written a few books for O'Reilly.

In person, he enjoys giving talks and workshops and spending time with the developer community around the world. When not coding, he is usually having fun outdoors with his wife and four children.

Comments

as defined by the client, or use PUT followed by a verification GET until both work. Generally, the first solution is much better, because recievers are much better positioned to be responsible for reliable delivery. Atom with archiving demonstrates good reliable delivery.

WS-AtomicTransactions. Distributed transactions are an antipattern. Ban them. If eBay can make payments between accounts at their auction site and paypal without them, so can you. Distributed transactions destroy your scalability, and they force you to give up either availability or network fault tolerance (CAP theorem). Use compensating transactions instead. If you need consistency, define a single source of truth and do ACID transactions within it.

WS-Security. This protocol is about achieving end-to-end security by using encryption and cryptographic signing to assure that messages may pass through intermediaries of lower trust without fear of tampering or disclosure. A better way to do this is to create an encrypted transport channel all the way from one endpoint to the other. This is called SSL. If you had to, you could make your representations contain encrypted and/or signed content using your favorite technologies for this. Only a middleware vendor would see any value in coupling this to your transport mechanism, though.

35 ^ | v • Reply • Share ›

OfficerX • 5 years ago

I'm not convinced by your "financial transactions require SOAP" argument. Here is my counter argument: how is it that I do financial transactions (as millions of other people do) using my bank website from my web browser, without the help from SOAP and all the WS technologies you mention?

14 ^ | v • Reply • Share ›

Maxx Velocity ➚ OfficerX • 3 years ago

When you use a bank website, you're not directly hitting the bank's processing API. You're submitting a form to a server, and once you submit it, it's no longer operable in the client. Native mobile apps on the other hand operate in real time, so client-state management becomes an issue.

12 ^ | v • Reply • Share ›

Craig Monroe ➚ Maxx Velocity • 3 years ago

Agreed with Maxx. You have no control over the services in the back-end once you've done your POST in your browser on the client side.

4 ^ | v • Reply • Share ›

Vincent Pazeller ↗ Craig Monroe • 9 months ago

I agree with OfficerX: this example does not require SOAP IMHO! You can use a single Rest POST call and provide all the data. E.g.:
POST https://mybank.com/api/transac...
{
"sourceAccount": "SJDVSASDA",
"destAccount": "WDKWDDJS",
"amount": 2500
}

Then, if you use a transactional DBMS on the backend, then I don't see where is the problem...

 • Reply • Share ›

Charlie ↗ Vincent Pazeller • 4 months ago

I think the issue is just that you have to check your database that the amount has indeed changed. Because you don't know if your server has received the request. If it hasn't then you need to get some sort of response verifying that the database has indeed changed. And if your first post didn't work, then you expect a user to try again after getting a notification that their deposit failed.

Overall I think it's just a lot of work for something that SOAP seems to do out of the box.

I'm not sure though. Fairly new to understanding SOAP, and I'm a novice at REST.

 • Reply • Share ›

Stuart ↗ Charlie • 3 months ago

Nobody should rely on a SOAP retry mechanism to prevent accidental duplicate transfers.

Instead, the banking API should require a unique (random) key for each transaction request. On the client side, this ID would be generated when loading the "Make New Transfer" page (web app or native, it doesn't matter). If someone successfully hits Submit twice on that page, two transaction requests are sent to the bank with identical keys. The bank can recognise and ignore the duplicates.

This is easily accomplished using REST.

1 · Reply · Share ›

wingi · 5 years ago

Point for SOAP: There is a defined interface included (as WSDL) and any client can use it without problems while updating the service. The usage of a REST interface is described mostly on a wiki page.

Point for REST: No problems with incompatibility with different SOAP-client/server.

But you got the other point: SOAP address methods, REST address resources.

8 · Reply · Share ›

OfficerX ↗ wingi · 5 years ago

REST/HTTP has a defined interface: it is the HTTP spec. You'll find the definitions of methods, status code and so on. Conent exchanged over HTTP requests/responses isusualy either specified in additional standard ocuments (i.e., XHTML, ATOM, etc.) or in your own doc (e.g., XML schema, etc.)

4 · Reply · Share ›

NelsonMinar · 4 years ago

This post has delivered a consistently steady trickle of links to my blog ("Why SOAP sucks") over the past two years. I assume people search Google for "SOAP vs REST", get sent here, then read far enough to click the links. I'm a little surprised this topic is so popular.

My own opinion is SOAP has no purpose in the world now except as a legacy technology. XML is wrong; use JSON. And the SOAP efforts at packaging communications are a failure. You're quite polite in highlighting the advanced WS-* features as being useful for solving problems. I think they are good attempts, but I wouldn't trust either the design or implementation of any of them for anything important.

9 · Reply · Share ›

white.thief · 5 years ago

Great post!

I would only like to point out that, even when almost all the REST architectures our there use HTTP as the underlying protocol, REST is an architectural style, so it is not bound by default to HTTP; quoting the Wikipedia: "REST was initially described in the context of HTTP, but is not limited to that protocol"[1].

[1] http://en.wikipedia.org/wiki/R...

4 · Reply · Share ›

Steve Francia Owner ➚ white.thief · 5 years ago

Yup, great point. Thanks for commenting!

· Reply · Share ›

Byterocky · 2 years ago

Thank you for the great quick comparison between REST and SOAP. Also sadly noticed that someone on StackOverflow is using your exact article as if their own and even linking to their site, please see http://stackoverflow.com/quest...

6 · Reply · Share ›

taxilian · 5 years ago

I think what it comes down to is simply that with all other considerations being the same, simple is better than complex.

Good to see this summarized in a coherent manner, though. Thanks!

3 · Reply · Share ›

patrick99 · 7 months ago

SOAP and REST are like comparing cows and trains. SOAP is a protocol, REST is an architectural style. REST can and does sometimes use SOAP. Further more, the vast majority of "RESTful" services are not even in fact restful, as they are not hypertext driven. In addition blindly mapping CRUD operations to HTTP methods is simply wrong, they have very well defined behaviours that have nothing to do with CRUD.

The author also seems to have a poor understanding of SOAP. I in addition strongly disagree with the authors conclusions (possibly due to his lack of understanding of SOAP and what RESTful services actually are), and having worked many years with enterprise apps and integration, in various industries, SOAP based web services are almost always the correct type of web services to be used.

2 · Reply · Share ›

Wilf Smith ➚ patrick99 · 4 months ago

Agreed totally. My rule of thumb is opposite to the original article - unless you have a good reason use SOAP (far more robust).

1 · Reply · Share ›

How is soap far more robust?

• Reply • Share ›

Richie • 9 months ago

SOAP stands for Simple Object Access Protocol. REST stands for REpresentational State Transfer.

SOAP is a XML based messaging protocol and REST is not a protocol but an architectural style.

SOAP has a standard specification but there is none for REST.

Whole of the web works based on REST style architecture. Consider a shared resource repository and consumers access the resources.

Even SOAP based web services can be implemented in RESTful style. REST is a concept that does not tie with any protocols.

SOAP is distributed computing style and REST is web style (web is also a distributed computing model).

---

see more

2 • Reply • Share ›

zzzxtreme • 3 years ago

most REST services are not REST-ful, so back to RPC, which SOAP serves the purpose. but too complex.
so my choice ? use JSON wrapped with XML-RPC

2 • Reply • Share ›

TechZilla • a year ago

Working in the financial industry now, I'm a Systems Engineer, there is one reason to use soap.... Objects, and or other granular components, which are not addressable with urls. Now you could likely come up with a scheme to RESTify the required SOAP benefits, but it would not be an out of box interface. Essentially ....You wan't to turn Java objects over HTTP, but at the same time in a launguage agnostic and verifiable schema? That;s what SOAP is for! For more Basic web services, yea they don't need all the structure SOAP requires... O and Json is no XML replacement... I know, blasphemy....

Christopher Koch · a year ago

It might be interesting to note, as an example, that the G2S (Game to System) API which is used by modern slot machines to communicate with casino services uses SOAP specifically for its security and ACID compliance because of the extremely strict nature of laws and regulations governing casino gambling.

1 · Reply · Share ›

JasonSilvestri · a year ago

Gotta love the end to the first paragraph, "I don't know... My background is building great
internet companies." Show us 3 of them and then tell us more about how you became a programmer over night. Reason 101 why it's best to get programming advice from a programmer.

1 · Reply · Share ›

Ashish Ramteke · 2 years ago

Nice article.

I Think both REST and SOAP can be used to implement similar functionality, but in general SOAP should be used when a particular feature of SOAP is needed, and the advantages of REST make it generally the best option otherwise. However, both REST and SOAP are often termed "Web services," and one is often used in place of the other, but they are totally different approaches. REST is an architectural style for building client-server applications. SOAP is a protocol specification for exchanging data between two endpoints.

http://www.csharptutorial.in/?...

1 · Reply · Share ›

StevenWung · 2 years ago

Still not clear about the differences

1 · Reply · Share ›

Chris Harrison ↗ StevenWung · 10 months ago

Just use REST ;)

1 · Reply · Share ›

sathyanarayana sastry chamarth · 3 years ago

Good article. Thanks for the details

1 · Reply · Share ›

Maxx Velocity · 3 years ago

Thanks for the article. I've never used SOAP, and after reading this, I'll be perfectly happy to continue avoiding it.

1 · Reply · Share ›

Ahmed · 5 years ago

I was just searching for this, many thanks for your efforts, keep up good work please.

1 · Reply · Share ›

Steve Francia Owner → Ahmed · 5 years ago

Thanks for the encouragement!

1 · Reply · Share ›

unni mana · a year ago

It is a late comment.

I was just thinking about to find the reason to use REST and SOAP until I read your article. Please correct me my understanding about these two.

1. In service oriented architecture in enterprise scenario, for example, getting an account details of a customer, which is not supposed to be changed by the calling entity, I would use SOAP kind of service. This will be exposed as SOAP service. I can integrate this service from different platform/technologies or scenarios.

2. REST will stand one step below from SOAP because of its strong relationship with HTTP protocol, so I will use REST to create a basic data set for customer object.

So the basic REST spec points to a simple CRUD operation along with its representational capabilities.

· Reply · Share ›

someone · a year ago

you are the best

· Reply · Share ›

Amudha · 2 years ago

Good post on REST and SOAP

prateek patil · 3 years ago

Never ending battle SOAP vs. REST @ http://spf13.com/post/soap-vs-...

Reply · Share ›

Krunoslav Hrnjak · 3 years ago

Really great article. SOAP seems like a little bit lost protocol since there already was a CORBA as binary standard for interoperability. Maybe there is a problem with REST since date exchange is rather arbitrary with now strict rules for validation but we love it for its simplicity. Thanx Steve

Reply · Share ›

adel kraiem · 4 years ago

Great Blog. Thx

Reply · Share ›

Sirajudheen Abdul Rahiman · 4 years ago

Good

Reply · Share ›

Tasman Hayes · 4 years ago

I've used SOAP from a website to a web coupon published to pull coupons. It worked great.

I've also used SOAP between a web app (PHP) and billing app (MS Basic) to push orders and pull back billing amounts. It works well for this app too.

SOAP can work really well. I've had no issues with it.

Reading your article and "Why Soap Sucks?" as well certainly gave me pause. :-)

SOAP hint: Cache the service definition (WSDL) for performance if you're doing a high volume of calls. This is configurable in PHP's SOAP implementation.

Reply · Share ›

E S · 4 years ago

A good summary of some important points, but I don't know that I really agree that the web doesn't care much about ACID transactions. Many applications do things that require ACID transactions such as updating multiple items at once, moving items from one group to another, or cascading deletes. And web

applications are just applications like any other so why should that factor in? Maybe you can explain further regarding what parts of ACID don't apply to the web?

Reply · Share ›

Igory Lr · 4 years ago

I case of "methods vs. resources" REST also wins. Just encode method calls to URIs. You have the benefit of not being tied to XML. Sure you can transfer anything in XML too but you need to encode/decode it to/from base64.

Reply · Share ›

orliesaurus · 5 years ago

Nice article, really liked it!
I was researching into WEB 2.0 and ended up reading your article for my essay!
Keep em coming

Reply · Share ›

Steve Francia Owner → orliesaurus · 5 years ago

Glad you enjoyed it.

Reply · Share ›

Chris · 6 years ago

Reading through this, I was thinking ahead to your conclusion - the combination of security and transaction/messaging management makes me think of financial services. Is there anything else? Is the takeaway that SOAP is useful in areas where data security/integrity is critical?

Thanks for looking up the answer - it's been nagging at me but REST's ascendancy has increasingly led me to believe this may not be an issue in the long term.

c

Reply · Share ›

Steve Francia Owner → Chris · 6 years ago

A good read is Why SOAP Sucks by Nelson Minar linked above. He was the guy that implemented Google's SOAP interface, which was often heralded as the example of SOAP done right. He also gives insight into why SOAP is no longer in use at google.

SOAP provides ACID compliant transactions, REST provides transactions, but they aren't ACID compliant. There are some key differences, but for most cases the extra bits that are required to be

ACID compliant are unnecessary especially on the web.

REST can do just about anything SOAP can, but without established standards. It's not that REST can't pass along messages, it does this job fine, but without standards, so each implementation can be different. This in itself isn't necessarily a positive or a negative, it allows flexibility at the expense of potentially more complicated integration.

REST makes a lot of sense for internet applications. It easily integrates with Javascript and flash. It follows the same paradigm of internet transmission of retry when the connection fails. I can't really think of any reason for an internet service to be anything but REST based. All the big

---