Can you explain briefly why you prefer either an SDK or direct HTTP?

(50 responses)

Easier to use from multiple client languages, no classpath issues.

SDK would handle all the errors and exceptions and well as input methods to the api which would be otherwise built by the client and waste time and resources.

HTTP leaves me free to implement client in most idiomatic way for my context

HTTP is trivial. Any SDK is bloat. An API that requires an SDK because direct HTTP is too difficult is probably very poorly engineered. Also, pure HTTP gives more flexibility. Note that his is not a strong preference. It also very much depends on the particular API.

I prefer not to maintain code I don't need to maintain. If the SDK is crap, I can always build my own client.

I want the SDK to guide me in good HTTP practise. I don't want to have to know HTTP spec inside out to work out how I should use the API.

As long as the API is simple to onboard and use, using an HTTP client gives me more flexibility to build an abstraction around the API better suited to my specific use case

Quicker to integrate, and some
Non-functionals like error handling are usually provided

Universal.

most of the time, an sdk gets in the way. You have to make choices in an sdk as to how to address the api and which constructs in the language to choose. Inevitably some of those choices will not be choices I would make.

For statically typed languages an SDK provides less friction.

Can focus on productivity rather than writing boilerplate. That said, if the SDK is poor or inflexible then HTTP is preferred.

It allows me to customise how I deserialize the response

The SDK will be wrong for whatever my context. It will be an additional dependency I don't want. I'd always take sample implementations though

Though an SDK usually allows quicker development. the Http way allows you to consume only what you need, so more flexibility and simplicity.

Less hassle and less error handling.

Language agnostic, strong tool support, cachable, easy to debug.

I don't want to depend on an SDK as it brings in more integration, dependency, chances of breakage....etc

Prefer not to bother with the plumbing and get to the data as fast as possible (dev time wise).

To easy the development time, considering its a robust and well tested SDK

I don't want to be tied to the producer's SDK release cycle. If I am using http I can be more free with the implementation of the client as it is in my control and I am not coupling the use of the API with their implementation of the SDK.

HTTP is entirely platform and language agnostic

I am not tied into the library and have the flexibility of dealing with changes to a well written API as and when required

The SDK can do all the HTTP stuff for me, it limits what I need to learn.

less boilerplate when using HTTP

Interoperability

Common language amongst developers.

HTTP is tech agnostic leaving me free to use any tech that I wish; a SDK is not

Because it can be better integrated into the clients own software stack

prefer Http as it gives more flexibility

Simplicity, flexibility

Every language supports HTTP. Unless there's a very complex data representation that needs to be normalized, plain HTTP is better suited.

I like to have the freedom to define my client behaviours

Easier, also can do testing in the browser very quickly

One less dependency to worry about.

reuse of existing tools and well known libraries for http. device & operating system agnostic makes for the easiest reuse. plus the developer of the api doesn't have to maintain, and support, a whole new set of code for accessing the api

Not really fussed. The SDK would obviously make it a lot easier to integrate into projects without having to worry about constructing the HTTP message.

Makes the API easier to consume

I assume you mean some language http library over a vendor sdk? In which case rest is a good abstraction itself and don't feel the need for it to be hidden from me. Though some syntactic sugar is always nice

Upgrades to direct HTTP is much more easy than use of SDK.

Language agnostic

HTTP is an independent standard and thus desirable.

It gives me more control and I'm less likely to encounter issues with vanilla HTTP - unless of course the API is so complex that the SDK is a neccesity

When committed to using SDKs, more API integrations mean more of someone else's code in our codebase. And that means more dependencies, which means more chances for conflicts.
An API should be very straightforward and meaningful with published possible error codes. If it's tough to use with a basic HTTP library, that's a problem.

HTTP an established standard; imo that's crucial in a world where systems can use different programming languages. Using HTTP provides an external commonality as opposed to shoehorning language specific SDKs together.

Easy to access and test

I could have chosen HTTP but having an SDK for the language I use would easy out and speedup my development process.

Direct HTTP allows for simple testing and discovery using curl or postman. It's then trivial to create your client in whichever language you like. Secondly, a SDK often turns out to be another layer where bugs may occur.

The sdk would abstract the api from the client and allow for changes in http codes, verbs or hateoas without requiring changes to the client code (ideally)

I already know how to use HTTP and will not have to learn how to use a vendor specific SDK