Ryan Tomayko

How I Explained REST to My Wife

Sunday, December 12, 2004

Translations of the following dialog available in Japanese, French, Vietnamese, Italian, Spanish, Portuguese, and Chinese. Huge thanks to YAMAMOTO Yohei, Karl Dubost, jishin, Barbz, Tordek, Edgard Arakaki, keven lw, respectively. If you know of additional translations, please leave a comment with the location.

Wife: Who is "Roy Fielding"?

Ryan: Some guy. He's smart.

Wife: Oh? What did he do?

Ryan: He helped write the first web servers and then did a ton of research explaining why the web works the way it does. His name is on the specification for the protocol that is used to get pages from servers to your browser.

Wife: How does it work?

Ryan: The web?

Wife: Yeah.

Ryan: Hmm. Well, it's all pretty amazing really. And the funny thing is that it's all very undervalued. The protocol I was talking about, HTTP, it's capable of all sorts of neat stuff that people ignore for some reason.

Wife: You mean "http" like the beginning of what I type into the browser?

Ryan: Yeah. That first part tells the browser what protocol to use. That stuff you type in there is one of the most important breakthroughs in the history of computing.

Wife: Why?

Ryan: Because it is capable of describing the location of something anywhere in the world *from* anywhere in the world. It's the foundation of the web. You can think of it like GPS coordinates for knowledge and information.

Wife: For web pages?

Ryan: For anything point to in that research I was talking about. The web is built on an architectural style called "REST". REST provides a definition of a "resource", which is what those things point to.

Wife: A web page is a resource?

Ryan: Kind of. A web page is a "representation" of a resource. Resources are just concepts. URLs--those things that you type into the browser...

Wife: I know what a URL is..

Ryan: Oh, right. Those tell the browser that there's a concept somewhere. A browser can then go ask for a specific representation of the concept. Specifically, the browser asks for the web page representation of the concept.

Wife: What other kinds of representations are there?

Ryan: Actually, representations is one of these things that doesn't get used a lot. In most cases, a resource has only a single representation. But we're hoping that representations will be used more in the future because there's a bunch of new formats popping up all over the place.

Wife: Like what?

Ryan: Hmm. Well, there's this concept that people are calling "Web Services". It means a lot of different things to a lot of different people but the basic concept is that machines could use the web just like people do.

Wife: Is this another robot thing?

Ryan: No, not really. I don't mean that machines will be sitting down at the desk and browsing the web. But computers can use those same protocols to send messages back and forth to each other. We've been doing that for a long time but none of the techniques we use today work well when you need to be able to talk to all of the machines in the entire world.

Wife: Why not?

Ryan: Because they weren't designed to be used like that. When Fielding and his buddies started building the web, being able to talk to any machine anywhere in the world was a primary concern. Most of the techniques we use at work to get computers to talk to each other didn't have those requirements. You just needed to talk to a small group of machines.

Wife: And now you need to talk to all the machines?

Ryan: Yes - and more. We need to be able to talk to all machines about all the stuff that's on all the other machines. So we need some way of having one machine tell another machine

about a resource that might be on yet another machine.

Wife: What?

Ryan: Let's say you're talking to your sister and she wants to borrow the sweeper or something. But you don't have it - your Mom has it. So you tell your sister to get it from your Mom instead. This happens all the time in real life and it happens all the time when machines start talking too.

Wife: So how do the machines tell each other where things are?

Ryan: The URL, of course. If everything that machines need to talk about has a corresponding URL, you've created *the machine equivalent of a noun*. That you and I and the rest of the world have agreed on talking about nouns in a certain way is pretty important, eh?

Wife: Yeah.

Ryan: Machines don't have a universal noun - that's why they suck. Every programming language, database, or other kind of system has a different way of talking about nouns. That's why the URL is so important. It let's all of these systems tell each other about each other's nouns.

Wife: But when I'm looking at a web page, I don't think of it like that.

Ryan: Nobody does. Except Fielding and handful of other people. That's why machines still suck.

Wife: What about verbs and pronouns and adjectives?

Ryan: Funny you asked because that's another big aspect of REST. Well, verbs are anyway.

Wife: I was just joking.

Ryan: It was a funny joke but it's actually not a joke at all. Verbs are important. There's a powerful concept in programming and CS theory called “polymorphism”. That's a geeky way of saying that different nouns can have the same verb applied to them.

Wife: I don't get it.

Ryan: Well.. Look at the coffee table. What are the nouns? Cup, tray, newspaper, remote. Now, what are some things you can do to all of these things?

Wife: I don't get it...

Ryan: You can *get* them, right? You can pick them up. You can knock them over. You can burn them. You can apply those same exact verbs to any of the objects sitting there.

Wife: Okay... so?

Ryan: Well, that's important. What if instead of me being able to say to you, "get the cup," and "get the newspaper," and "get the remote"; what if instead we needed to come up with different verbs for e[illegible] had to think up a new word for each verb/noun combination.

Wife: Wow! That's weird.

Ryan: Yes, it is. Our brains are somehow smart enough to know that the same verbs can be applied to many different nouns. Some verbs are more specific than others and apply only to a small set of nouns. For instance, I can't drive a cup and I can't drink a car. But some verbs are almost universal like `GET`, `PUT`, and `DELETE`.

Wife: You can't `DELETE` a cup.

Ryan: Well, okay, but you can throw it away. That was another joke, right?

Wife: Yeah.

Ryan: So anyway, HTTP--this protocol Fielding and his friends created--is all about applying verbs to nouns. For instance, when you go to a web page, the browser does an HTTP `GET` on the URL you type in and back comes a web page.

Web pages usually have images, right? Those are separate resources. The web page just specifies the URLs to the images and the browser goes and does more HTTP GETs on them until all the resources are obtained and the web page is displayed. But the important thing here is that very different kinds of nouns can be treated the same. Whether the noun is an image, text, video, an mp3, a slideshow, whatever. I can GET all of those things the same way given a URL.

Wife: Sounds like GET is a pretty important verb.

Ryan: It is. Especially when you're using a web browser because browsers pretty much just `GET` stuff. They don't do a lot of other types of interaction with resources. This is a problem because it has led many people to assume that HTTP is just for `GET`ing. But HTTP is actually a *general purpose protocol for applying verbs to nouns*.

Wife: Cool. But I still don't see how this changes anything. What kinds of nouns and verbs do you want?

Ryan: Well the nouns are there but not in the right format.

Think about when you're browsing around amazon.com looking for things to buy me for Christmas. Imagine each of the products as being nouns. Now, if they were available in a representation that a machine could understand, you could do a lot of neat things.

Wife: Why can't a machine understand a normal web page?

Ryan: Because web pages are designed to be understood by people. A machine doesn't care about layout and styling. Machines basically just need the data. Ideally, every URL

would have a human readable and a machine readable representation. When a machine GETs the resource, it will ask for the machine readable one. When a browser GETs a resource for a human, it will ...

Wife: So people would have to make machine formats for all their pages?

Ryan: If it were valuable.

Look, we've been talking about this with a lot of abstraction. How about we take a real example. You're a teacher - at school I bet you have a big computer system, or three or four computer systems more likely, that let you manage students: what classes they're in, what grades they're getting, emergency contacts, information about the books you teach out of, etc. If the systems are web-based, then there's probably a URL for each of the nouns involved here: student, teacher, class, book, room, etc. Right now, getting the URL through the browser gives you a web page. If there were a machine readable representation for each URL, then it would be trivial to latch new tools onto the system because all of that information would be consumable in a standard way. It would also make it quite a bit easier for each of the systems to talk to each other. Or, you could build a state or country-wide system that was able to talk to each of the individual school systems to collect testing scores. The possibilities are endless.

Each of the systems would get information from each other using a simple HTTP `GET`. If one system needs to add something to another system, it would use an HTTP `POST`. If a system wants to update something in another system, it uses an HTTP `PUT`. The only thing left to figure out is what the data should look like.

Wife: So this is what you and all the computer people are working on now? Deciding what the data should look like?

Ryan: Sadly, no. Instead, the large majority are busy writing layers of complex specifications for doing this stuff in a different way that isn't nearly as useful or eloquent. Nouns aren't universal and verbs aren't polymorphic. We're throwing out decades of real field usage and proven technique and starting over with something that looks a lot like other systems that have failed in the past. We're using HTTP but only because it helps us talk to our network and security people less. We're trading simplicity for flashy tools and wizards.

Wife: Why?

Ryan: I have no idea.

Wife: Why don't you say something?

Ryan: Maybe I will.
