Website uses browser cookies to give you the best possible experience.
To make Website better our cookies collect data about how you use Website, which pages you visit and what you do on it. By continuing to use Website, you agree to our Privacy Policy. Accept

REST in the new SOAP

Kristian Köehntopp (kris@koehntopp.de)

Update: Many people have asked me what the alternative to REST is. I have written a follow-up to this rant that lays out the alternatives I see: Protocol Buffers and gRPC and GraphQL and Thrift and Cap'n Proto and Flatbuffers and SOAP.

Introduction

REST is popular. Its Wikipedia entry is extensive and long. Its Google Ngram shows it as the most popular way to talk about web services, surpassing other contenders for the throne such as SOAP.

Google Ngram for "REST" and "SOAP" and "CORBA" and "XML-RPC" and "JSON-RPC" and "Thrift" and "Protocol Buffers" and "Cap'n Proto" and "Flatbuffers" and "Avro" and "MQTT" and "GRPC"

REST is a set of rules that describe how to define and address such services, with a few decisions left for implementers.

REST is a set of rules that describe how to address a resource and it's instances, with a few decisions left for implementers.

Not a new thing
Roy Fielding's PhD thesis.

Fielding's description of the architecture of the world wide web.

REST was been formalized as HATEOAS extension of HTTP/1.1 in RFC 2616 (Hypertext Transfer Protocol).

Roy Fielding's rant: Web APIs must be hypertext driven.

What's the problem with REST?

It takes a length to make it big enough. And to a length still. And big enough to be recognized as a problem.

```
GET /service/foo/bar HTTP/1.1
Host: example.com
Accept: application/vnd.foo.v1+text

HTTP/1.1 200 OK
Date: Wed, 23 Jan 2013 23:59:59 GMT
Content-Type: application/vnd.foo.v1+text
Content-Length: 1234

{"id": 1, "name": "Foo", "status": "active", "created": "2013-01-15", "updated": "2013-01-20", "owner": {"id": 42, "name": "Bar", "email": "bar@example.com"}, "tags": ["alpha", "beta", "gamma"], "meta": {"version": 3, "checksum": "a1b2c3d4", "cache_ttl": 300}, "items": [{"id": 101, "name": "Item One", "status": "pending"}, {"id": 102, "name": "Item Two", "status": "shipped"}], "pagination": {"page": 1, "per_page": 25, "total": 2, "total_pages": 1}}
```

GET /service/foo/bar/{id} HTTP/1.1
Host: example.com
Accept: application/vnd.foo.v1+text

HTTP/1.1 200 OK
Date: Wed, 23 Jan 2013 23:59:59 GMT
Content-Type: application/vnd.foo.v1+text
Content-Length: 1234

{"id": 1, "name": "Foo", "status": "active", "created": "2013-01-15", "updated": "2013-01-20", "owner": {"id": 42, "name": "Bar", "email": "bar@example.com"}, "tags": ["alpha", "beta", "gamma"], "meta": {"version": 3, "checksum": "a1b2c3d4", "cache_ttl": 300}, "items": [{"id": 101, "name": "Item One", "status": "pending"}, {"id": 102, "name": "Item Two", "status": "shipped"}], "pagination": {"page": 1, "per_segment": 25, "total": 2, "total_pages": 1}}

Let's talk about the Content-Type header.
Let's talk about the Accept header.
Let's talk about a Content-Type header that is not a MIME type.
Let's talk about a Content-Type header that is a MIME type.

Let's talk about how Accept-Encoding: gzip is a MIME type.
Let's talk about how Content-Encoding: gzip is a MIME type.
Let's talk about how Transfer-Encoding: chunked is a MIME type.
Let's talk about how Accept-Encoding: chunked is a MIME type.

Content-Type is a type like this or a type with a parameter like this.
Let's talk about type SOAPAction.

Let's talk about Content-Type: text/plain; charset=utf-8.
Let's talk about Accept: text/plain; charset=utf-8.
Let's talk about Content-Type: text/plain; charset=utf-8; foo=bar.
Let's talk about Accept: text/plain; charset=utf-8; foo=bar.

Let's talk about Content-Type: text/plain; charset=utf-8; foo=bar; baz=qux.
Let's talk about Accept: text/plain; charset=utf-8; foo=bar; baz=qux.

Let's talk about Content-Type: text/plain; charset=utf-8; foo=bar; baz=qux; gzip=true.
Let's talk about Accept: text/plain; charset=utf-8; foo=bar; baz=qux; gzip=true.

Let's talk about Content-Type: text/plain; charset=utf-8; foo=bar; baz=qux; gzip=true; chunked=true.
Let's talk about Accept: text/plain; charset=utf-8; foo=bar; baz=qux; gzip=true; chunked=true.

NO! LOOK, WHY DOES EVERYONE KEEP DOING THIS? WHAT IS WRONG WITH YOU?

Let's talk. Please. Let's talk about Accept-Ranges.

The joy of REST verbs

Let's talk about the joy of REST verbs.

REST defines a set of verbs that can be used with URIs to define actions on resources.

REST defines GET as idempotent, safe, cacheable, and HEAD as the same without the response body.

REST defines PUT as idempotent, but not safe. PUT is used to create or update a resource identified by a URI. PUT is used to create or update a resource identified by a URI. PUT is used to create or update a resource identified by a URI