# DISCLAIMER

Knowledge is not power; Implementation is power.

The opinions expressed in this presentation are my own and developed separate from my current employer.

All software code is mine and was developed personally and not in conjunction with my current employer.  They do not have rights to this code.

I have placed this in the public domain for educational purposes and the betterment of the community.

However, future I may create variations of this code later for use in software projects for my employer at the time or in contract work.

Everything is located at this public repository:

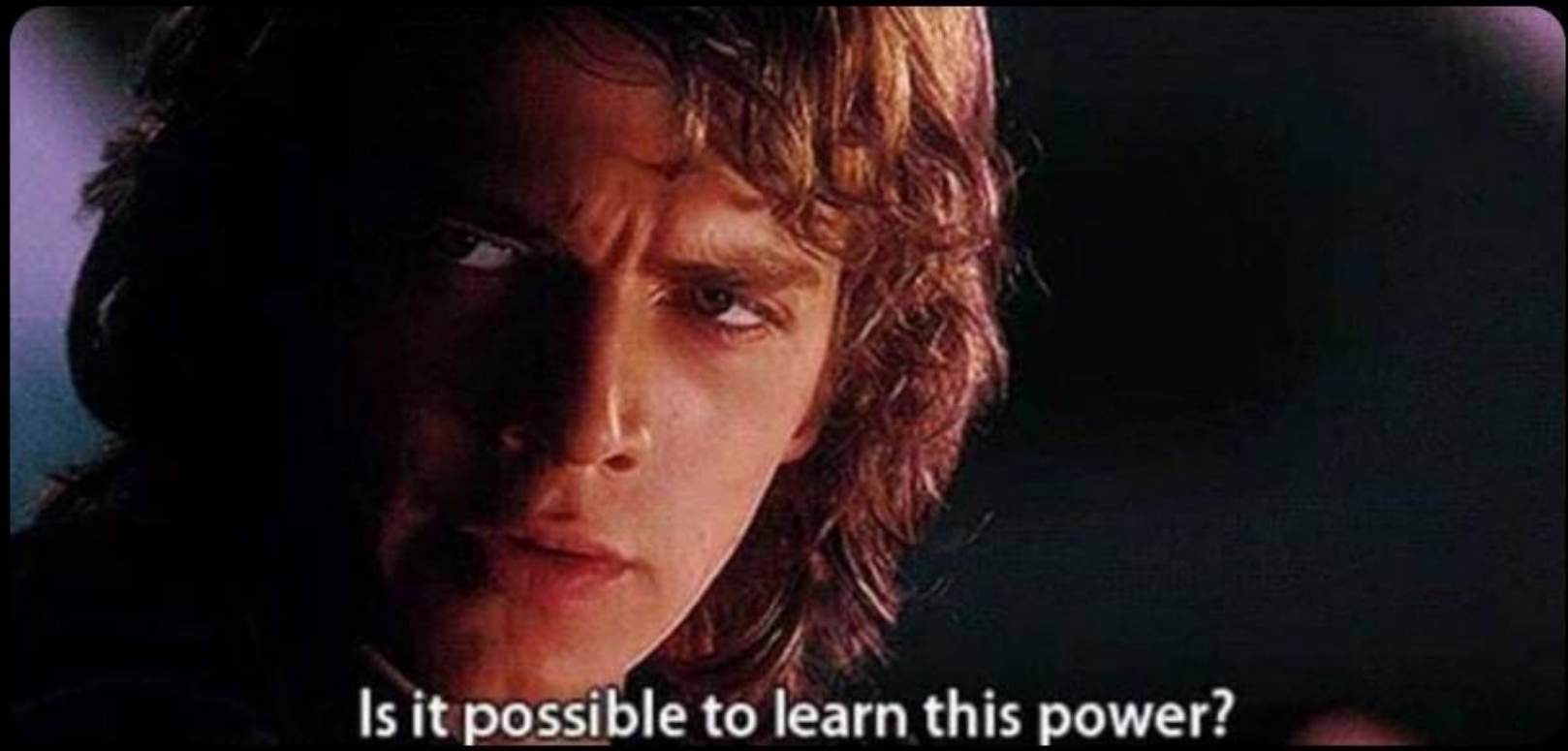
RonGarlit/RESTfulApiDemo (github.com)

# RESTFUL API THE HATEOAS WAY

Everything you wanted to know and didn't

By: Ronald Garlit

# HATEOAS



# A LITTLE HISTORY FIRST

- So about 20 years ago Roy Thomas Fielding submitted his PhD dissertation paper on "Architectural Styles and the Design of Network-based Software Architectures". There is much controversy today about this. But it is generally accepted that this is the foundational document of what we call REST.
- Roy was not just some PhD student in 1994, he began working at and for the World Wide Web Consortium.
- He co-authored the HTTP 1.0 specification in 1996.
- Whose other claim to fame is having co-founded the Apache web server project.

# "I AM GETTING FRUSTRATED"

In 2008 Roy was shouting from the roof tops that "REST APIs must be hypertext-driven". The state of REST (Representational State Transfer) was being perverted by those who just misunderstood what he had meant.

To be clear:

"In other words, if the engine of application state (and hence the API) is not being driven by hypertext, then it cannot be RESTful and cannot be a REST API. Period." – Roy Fielding, October 20, 2008

He then laid out the "rules before calling your creation a REST API".

(Which to many were as clear as mud.)

# RULES ARE NOT STANDARDS

In a nutshell you need to understand that REST is an Architecture, NOT a Standard.

There are many "Standards" out there for various aspects of software design that are used in REST.

But REST itself is an architecture.

My point is this.

- REST can be implemented following the guidelines Roy outlined.
- It is today generally implemented on top of the HTTP "Standard".
- But REST itself is not a standard, but rather it is an architectural style that provides constraints for the design.

# RESTFUL DEFINED?

It should be noted that MANY APIs do not conform to every element of REST.

These are called ‘RESTful” APIs.

But they are really HTTP APIs.

Most people today assume that REST and RESTful are the same things.

GRAY AREA ALERT!

They are the same, but only in the way that a skyscraper is the same as a residential single-family home.

- They are building.
- They provide have similar features and functionality.
- But they are different.

# THE FIGHT FOR SANITY

Opinions are varied and strong.

Developer A: It's REST.
Developer B: No RESTful.
Developer A: They are NOT the same!
Developer B: Yes they are!
…

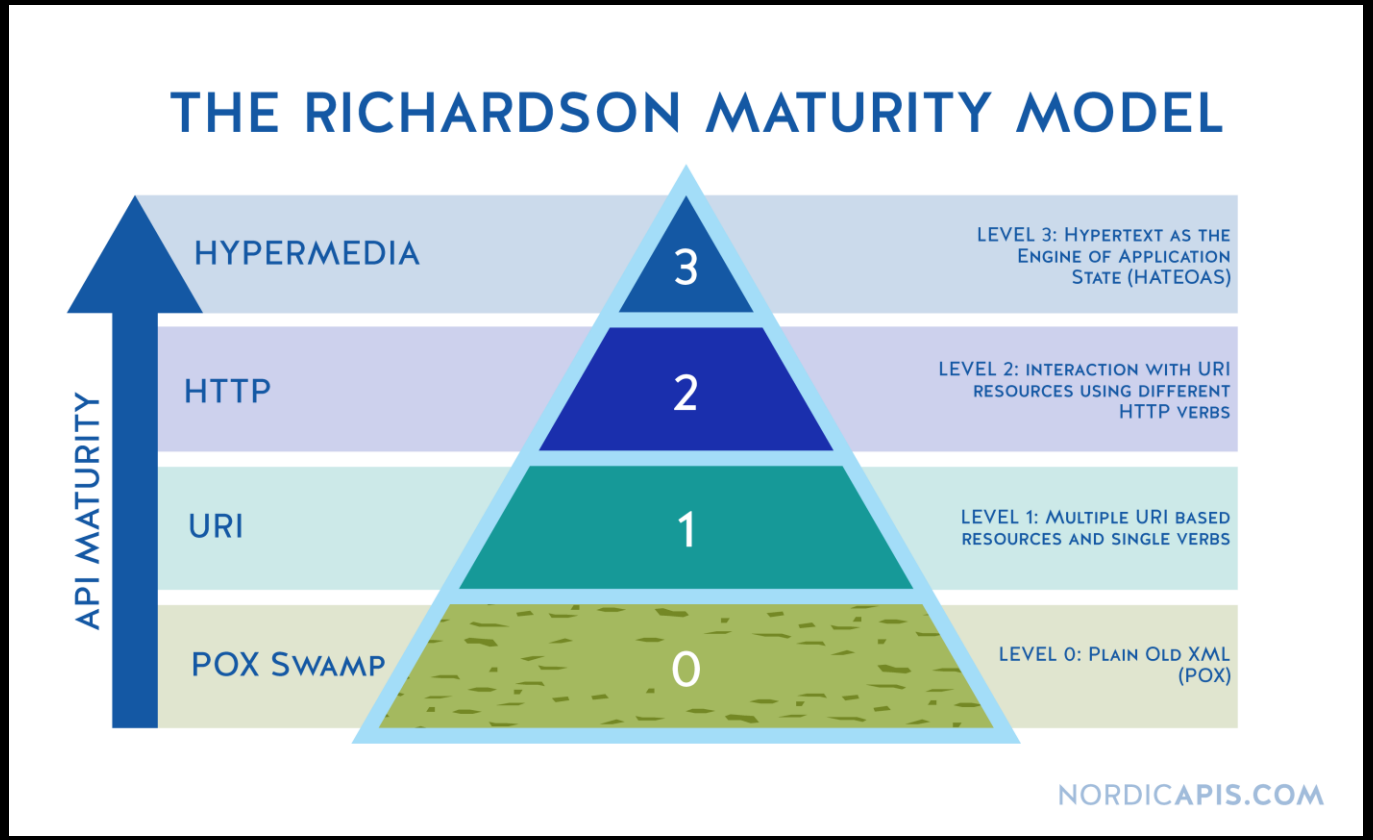
An that argument continues even to this day.

But, meanwhile in 2008 while Roy Feilding was having a fit.

Leonard Richardson defined a "Maturity Model" that classifies Web APIs based on their adherence and conformity to each of the model's four levels.

# LEONARD TO THE RESCUE!

That is when the "Richardson Maturity Model" (RMM) was born.



# SIMPLY PUT

Leonard Richardson created this for “Web APIs” based on the HTTP Protocol.

There are only four levels (for now)

- Level 0: Swamp of POX
- Level 1: Resources
- Level 2: HTTP verbs
- Level 3: Hypermedia controls

# WELCOME TO THE SWAMP

Level 0: Swamp of POX

These are typical RPC POX and many SOAP web services.

It exposes only one universal endpoint as the entry point (URI) and one kind of method which in HTTP, this normally is the POST method.

# UNIFORM RESOURCE IDENTIFIERS

Level 1: Resources

APIs that can distinguish between different resources are level 1.

This level uses multiple URIs.

Every URI is the entry point to a specific resource.

Examples are:

http://example.org/articles
http://example.org/article/1
http://example.org/article/2

But this level uses only one single method like POST.

# ACTIONS

Level 2: HTTP verbs

Yes you would be right to think Roy Feilding would have a fit.
Richardson said that at this level 2 that the API MUST use HTTP verbs.

But REST is an architectural style that should be completely protocol agnostic.
So if you want to use a different protocol, your API can still be RESTful.

# HYPERMEDIA CONTROLS

Level 3: Hypermedia controls

The highest level, uses HATEOAS to deal with discovering the possibilities of your API towards the clients.

An that my friends is what we are here for.

# HATEOAS

Typically, when we perform a REST request, we only get the data and not any actions around it.

This is where HATEOAS comes in the fill in the gap.

A HATEOAS request allows you to not only send the data but also specify the related actions.

See the HAL-Hypertext Application Language for detailed examples.

# ENOUGH TALK!!

It is time for the walkthrough and demo.

# ANY QUESTIONS

# WHERE TO GET THE GOODIES

Everything is located at this public repository:

RonGarlit/RESTfulApiDemo (github.com)