A bit of REST (Representational State Transfer)
Roy T. Fielding | Senior Principal Scientist, Adobe
Ph.D. Dissertation
Why talk about REST?

Because

REST

has become a

BUZZWORD

There's nothing particularly wrong with that... unless you happen to be me... or working with me

What is REST Anyway?

- **Representational State Transfer (REST)** is a style of software architecture for *distributed hypermedia systems* such as the World Wide Web

- Roy Fielding looked at the Web and saw that it was *good*
A bit of context: REST also began 20 years ago
A bit of context: REST also began 20 years ago
A bit of context: REST also began 20 years ago
A bit of context: REST also began 20 years ago

- Using XMosaic
- libwww-perl
- Relative URLs
- HTML 2.0
- Conditional GET
- MOMspider
- www.stat
- www.ics.uci.edu
- HTTP editor
- SJ IETF
- 2nd WWW
- 1st WWW
- 10,022
- 2,738
- 623
- 130
- Jun 93
- Dec 93
- Jun 94
- Dec 94
- Jun 95

Jul 2015: 849,602,745 (36,127x)
Adobe ColdFusion Documentation [Sep 2014]:

ColdFusion 10 lets you create and publish REST (Representational State Transfer) services that can be consumed by clients over HTTP/HTTPS request.

What is REST

The following URL takes you to the Java Tutorial that provides conceptual information on REST: http://download.oracle.com/javaee/6/tutorial/doc/gijqy.html
We all know about REST in ColdFusion, right?

Adobe ColdFusion Documentation [Sep 2014]:

ColdFusion 10 lets you create and publish REST (Representational State Transfer) services that can be consumed by clients over HTTP/HTTPS request.

*What is REST*

The following URL takes you to the Java Tutorial that provides conceptual information on REST:

**JAX-RS (Jersey)**

Java API for RESTful Web Services
Am I going to talk about ColdFusion's implementation of REST APIs and API Management?

No,

REST is NOT an implementation
Three (very different) perspectives of the Web
Web Implementation (origin view)

- Intermediary
- Proxy Cache
- Application Servers
- Dynamic Content
- Centralized Data
  - RDBMS, NFS, SAN

- User Agents
- Webservers/Gateways
  - Accelerator Cache
Web Architecture

Architecture is a vertical abstraction on implementation

User Agents -> Proxies -> Gateways -> Origin Servers

© 2015 Adobe Systems Incorporated. All Rights Reserved.
Web protocols define that vertical abstraction on implementation
So, is REST the Web Architecture?

No, REST is NOT an architecture!
Original proposal for the World Wide Web

[Berners-Lee, 1989]
The Web is an application integration system

- Hyper Card
- ENQUIRE
- Hyperlinked information
- Computer conferencing
- VAX/NOTES
- A Proposal "Mesh"
- CERNDOC
- IBM GroupTalk
- uucp News
- Hierarchical systems

For example:
- unifies
- describes

[Berners-Lee, 1989]
The Web is an application integration system

-Hypertext
-Linked information
-Hypermedia
-CERNDOC
-ENQUIRE
-IBM GroupTalk
-uucp News
-VAX/NOTES
-Hierarchical systems

HyperCard

for example

ENQUIRE

for example

VAX/NOTES

unifies

A Proposal "Mesh"

describes

CERNDOC

[Berners-Lee, 1989]
Network-based Applications

• Application
  • short for “applying a computer to accomplish a given purpose”
  • examples: finding a document, managing a bank account, or buying a travel ticket

• Network-based
  • operating over the network with full knowledge of the user
  • i.e., unlike distributed, which intentionally hides the network
Architectural Styles

- **A horizontal abstraction** across multiple architectures (vertical abstractions)
  - names a repeated architectural pattern
  - defined by its design constraints
  - chosen for the properties they induce

- **REST is an architectural style**
  - for network-based applications
  - to induce a specific set of architectural properties
  - desired for the World Wide Web
REST is an accumulation of design constraints that induce architectural properties.
REST is an accumulation of design constraints that induce architectural properties
REST is an accumulation of design constraints that induce architectural properties.
REST’s Five Uniform Interface Constraints

- All important resources are identified by one resource identifier mechanism
  - induces simple, visible, reusable, stateless communication

- Access methods have the same semantics for all resources
  - induces visible, scalable, available through layered system, cacheable, and shared caches

- Resources are manipulated through the exchange of representations
  - induces simple, visible, reusable, cacheable, and evolvable (information hiding)

- Representations are exchanged via self-descriptive messages
  - induces visible, scalable, available through layered system, cacheable, and shared caches
  - induces evolvable via extensible communication

- Hypertext as the engine of application state
  - induces simple, visible, reusable, and cacheable through data-oriented integration
  - induces evolvable (loose coupling) via late binding of application transitions
Why the hypermedia constraint?

Hypertext as the Engine of Application State

- each state can be dynamic
- each transition can be redirected
The client only needs to know one state and its transitions!

Follow Your Nose
The client only needs to know one state and its transitions!

Follow Your Nose
The client only needs to know one state and its transitions!

Follow Your Nose
The client only needs to know one state and its transitions!

Follow Your Nose
REST emphasizes evolvability
to sustain an uncontrollable system

If you think you have control over the system
or aren’t interested in evolvability,
don’t waste your time arguing about REST
What is the most common question about REST?

So, where is your ... 

REST API?
An API that provides network-based access to resources via a uniform interface of self-descriptive messages containing hypertext to indicate potential state transitions might be part of an overall system that is a RESTful application.
Some tips for building an API for RESTful applications

- Identify all of the resources
  - few resources are atomic; most are collections or views of other resources
  - don't confuse identity (naming) with containment (storage)
  - use access control, not obscurity, to control publication
  - resources have more in common with stored procedures than they do with records or files

- Iteratively develop resources and state transitions (use cases)
  - don't try to do everything at once
  - don't make any assumptions about received content, order, versioning, etc.

- Be flexible regarding media types and access protocols
  - start by prototyping in HTML and exploring with browsers and spiders
  - if you need to publish JSON, use a profile that defines hypertext semantics
  - use relative URLs wherever possible (to save space and improve portability)
Don’t over-think the problem space

a RESTful API is just a website for users with a limited vocabulary (machine to machine interaction)
Don’t under-think the problem space

building a good website is not easy (but it has been done before)
So, what does that mean for ColdFusion?

Why are we using an API designed by Sun/Oracle to build a website?
So, what does that mean for ColdFusion?

Why are we using an API designed by Sun/Oracle to build a website?

Wouldn't it be better to use a language for rapid application development that could automatically select its output serialization to match the media type in which it is embedded?