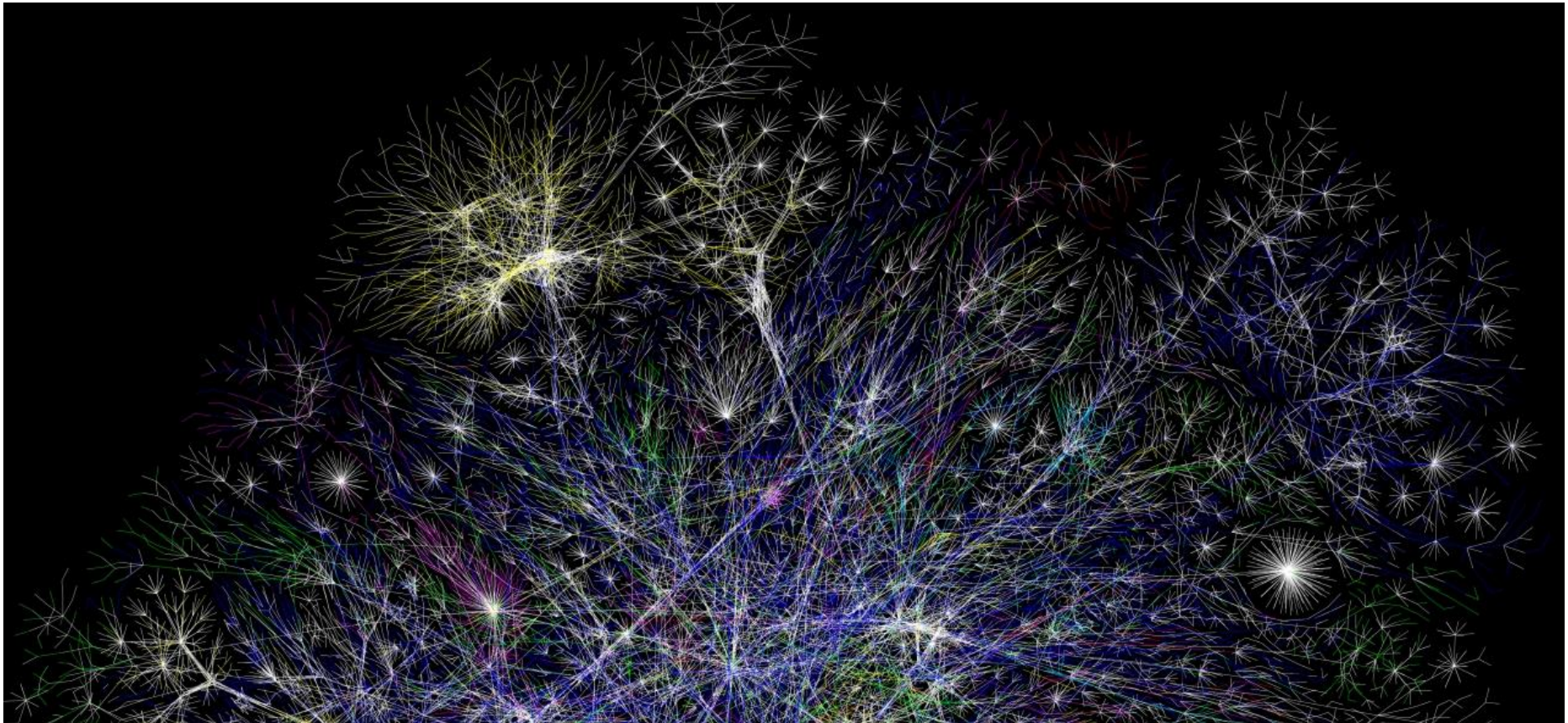


API Design Methodology


Mike Amundsen
API Academy / CA
@mamund



APIs for the Web



meth·od·ol·o·gy

/ˌmeTHəˈdäləjē/ 

noun

a system of methods used in a particular area of study or activity.
"a methodology for investigating the concept of focal points"



Actually, we have a methodology
already...



Design Guidelines

Craft [good/pretty/usable/stable] URIs



Term	Description
Authority	A URI component that identifies the party with jurisdiction over the namespace defined by the remainder of the URI.
Collection	A resource archetype used to model a server-managed <i>directory</i> of resources.
Controller	A resource archetype used to model a procedural concept.
CRUD	An acronym that stands for the four classic storage-oriented functions: create, retrieve, update, and delete.
Developer portal	A Web-based graphical user interface that helps a REST API acquire new clients.
Docrout	A resource that is the hierarchical ancestor of all other resources within a REST API's model. This resource's URI should be the advertised entry point.
Document	A resource archetype used to model a singular concept.
Forward slash separator (/)	Used within the URI path component to separate hierarchically related resources.
Opacity of URIs	An axiom, originally described by Tim Berners-Lee, that governs the visibility of a resource identifier's composition.
Parent resource	The document, collection, or store that governs a given subordinate concept by preceding it within a URI's hierarchical path.
Query	A URI component that comes after the path and before the optional fragment.
Resource archetypes	A set of four intrinsic concepts (document, collection, store, and controller) that may be used to help describe a REST API's model.
Store	A resource archetype used to model a client-managed resource repository.
URI path segment	Part of a resource identifier that represents a single node within a larger, hierarchical resource model.
URI template	A resource identifier syntax that includes variables that must be substituted before resolution.

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Map domain actions to HTTP methods (CRUD)



Term	Description
DELETE	HTTP request method used to remove its parent.
GET	HTTP request method used to retrieve a representation of a resource's state.
HEAD	HTTP request method used to retrieve the metadata associated with the resource's state.
OPTIONS	HTTP request method used to retrieve metadata that describes a resource's available interactions.
POST	HTTP request method used to create a new resource within a collection or execute a controller.
PUT	HTTP request method used to insert a new resource into a store or update a mutable resource.
Request-Line	RFC 2616 defines its syntax as Method SP Request-URI SP HTTP-Version CRLF
Request method	Indicates the desired action to be performed on the request message's identified resource.
Response status code	A three-digit numeric value that is communicated by a server to indicate the result of a client's request.
Status-Line	RFC 2616 defines its syntax as: HTTP-Version SP Status-Code SP Reason-Phrase CRLF
Tunneling	An abuse of HTTP that masks or misrepresents a message's intent and undermines the protocol's transparency.

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Map domain actions to HTTP methods (CRUD)

Use the proper HTTP Status Codes



Code	Name	Meaning
400	Bad Request	Indicates a nonspecific client error
401	Unauthorized	Sent when the client either provided invalid credentials or forgot to send them
402	Forbidden	Sent to deny access to a protected resource
404	Not Found	Sent when the client tried to interact with a URI that the REST API could not map to a resource
405	Method Not Allowed	Sent when the client tried to interact using an unsupported HTTP method
406	Not Acceptable	Sent when the client tried to request data in an unsupported media type format
409	Conflict	Indicates that the client attempted to violate resource state
412	Precondition Failed	Tells the client that one of its preconditions was not met
415	Unsupported Media Type	Sent when the client submitted data in an unsupported media type format
500	Internal Server Error	Tells the client that the API is having problems of its own

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Map domain actions to HTTP methods (CRUD)

Use the proper HTTP Status Codes

Document serialized objects as HTTP bodies



Term	Description
Field	A named slot with some associated information that is stored in its value.
Form	A structured representation that consists of the fields and links, which are defined by an associated schema.
Format	Describes a form's presentation apart from its schematic.
Link	An actionable reference to a resource.
Link formula	A boolean expression that may serve as HATEOAS calculator's input in order to determine the availability of state-sensitive hypermedia within a form.
Link relation	Describes a connection between two resources.
Schema	Describes a representational form's structure independent of its format.
State fact	A Boolean variable that communicates a condition that is relevant to some state-sensitive hypermedia.

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Map domain actions to HTTP methods (CRUD)

Use the proper HTTP Status Codes

Document serialized objects as HTTP bodies

Use HTTP headers responsibly



Code	Purpose
Content-Type	Identifies the entity body's media type
Content-Length	The size (in bytes) of the entity body
Last-Modified	The date-time of last resource representation's change
ETag	Indicates the version of the response message's entity
Cache-Control	A TTL-based caching value (in seconds)
Location	Provides the URI of a resource

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Map domain actions to HTTP methods (CRUD)

Use the proper HTTP Status Codes

Document serialized objects as HTTP bodies

Use HTTP headers responsibly

Describe edge cases (async, errors, authN/Z)



Response

HTTP/1.1 202 Accepted ❶

Content-Type: application/xml; charset=UTF-8

Content-Location: http://www.example.org/images/task/1

Date: Sun, 13 Sep 2009 01:49:27 GMT

```
<status xmlns:atom="http://www.w3.org/2005/Atom">
```

```
  <state>pending</state>
```

```
  <atom:link href="http://www.example.org/images/task/1" rel="self"/>
```

```
  <message>Your request has been accepted for processing.</message>
```

```
  <ping-after>2009-09-13T01:59:27Z</ping-after> ❷
```

```
</status>
```

Response

HTTP/1.1 409 Conflict

Content-Type: application/xml; charset=UTF-8

Content-Language: en

Date: Wed, 14 Oct 2009 10:16:54 GMT

Link: <http://www.example.org/errors/limits.html>; rel="help"

<error xmlns:atom="http://www.w3.org/2005/Atom">

<message>Account limit exceeded. We cannot complete the transfer due to
insufficient funds in your accounts</message>

<error-id>321-553-495</error-id>

<account-from>urn:example:account:1234</account-from>

<account-to>urn:example:account:5678</account-to>

<atom:link href="http://example.org/account/1234"
rel="http://example.org/rels/transfer/from"/>

<atom:link href="http://example.org/account/5678"
rel="http://example.org/rels/transfer/to"/>

</error>

Request to obtain a request token

POST /request_token HTTP/1.1 ❶

Host: www.example.org

Authorization: OAuth realm="http://www.example.com/photos",
 oauth_consumer_key=a1191fd420e0164c2f9aeac32ed35d23,
 oauth_nonce=109843dea839120a,
 oauth_signature=d8e19bb988110380a72f6ca33b2ba5903272fe1,
 oauth_signature_method=HMAC-SHA1,
 oauth_timestamp=1258308730,
 oauth_version=1.0 ❷

Content-Length: 0

Response containing a request token and a secret

HTTP/1.1 200 OK

Content-Type: application/x-www-form-urlencoded

oauth_token=0e713d524f290676de8aff4073b1bb52e37f065c

&oauth_token_secret=394bc633d4c93f79aa0539fd554937760f05987c ❸

Designing Consistent RESTful Web Service Interfaces



REST API

Design Rulebook

O'REILLY®

Mark Massé

Solutions for Improving Scalability and Simplicity



O'REILLY®

YAHOO! PRESS

Subbu Allamaraju

But there's a problem here...




Those are not *design* guidelines..



They are *implementation* guidelines!



im·ple·men·ta·tion

/,ɪmpləmənˈtāSHən/ 

noun

the process of putting a decision or plan into effect; execution.
"she was responsible for the implementation of the plan"



OREILLY



RESTful
Web Clients

Enabling REST through HyPaaS

Mike Amundsen

Ok, so what is a *design methodology*, then?



de·sign

/dəˈzɪn/ 

noun

1. a plan or drawing produced to show the look and function or workings of a building, garment, or other object before it is built or made.
"he has just unveiled his design for the new museum"
synonyms: plan, blueprint, drawing, sketch, outline, map, plot, diagram, draft, representation, scheme, model [More](#)
2. purpose, planning, or intention that exists or is thought to exist behind an action, fact, or material object.
"the appearance of design in the universe"
synonyms: intention, aim, purpose, plan, intent, objective, object, goal, end, target; [More](#)



Services for a Changing World

RESTful Web APIs



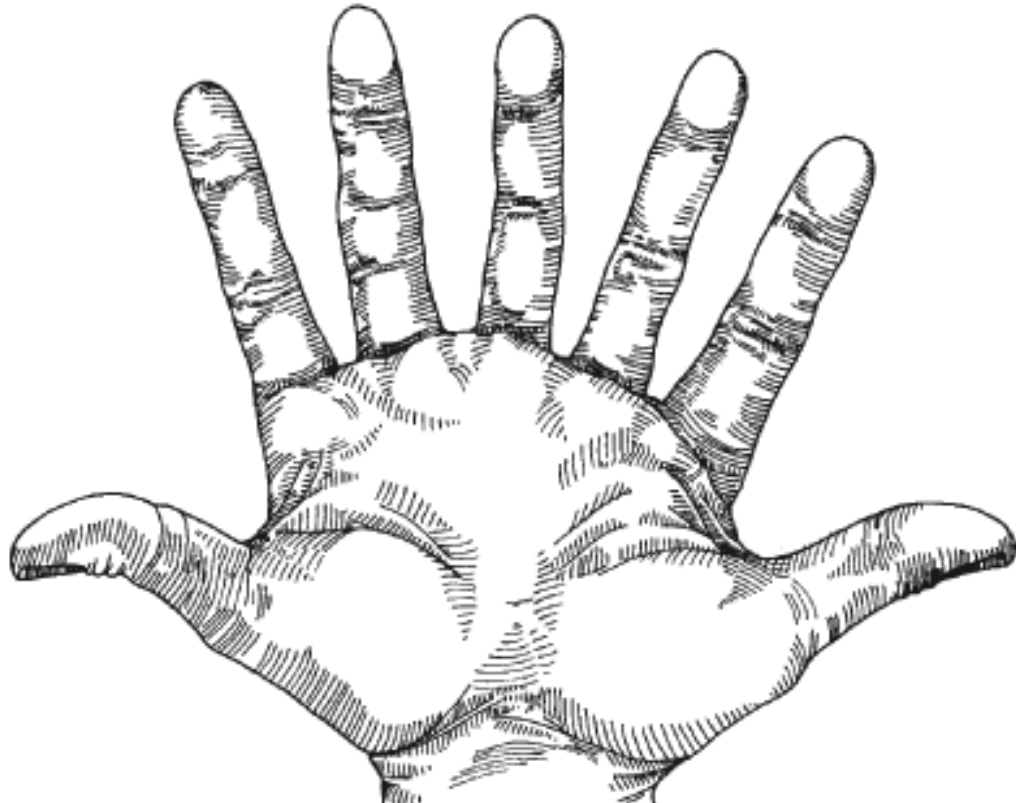
O'REILLY®

*Leonard Richardson,
Mike Amundsen & Sam Ruby*

Here's a simple seven-step procedure...



Yep, seven steps.



1. List the Semantic Descriptors



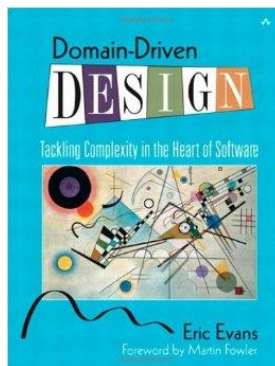
1. List the Semantic Descriptors *(the what?)*



Ubiquitous Language

- *Ubiquitous Language is the term Eric Evans uses in **Domain Driven Design** for the practice of building up a common, rigorous language between developers and users. This language should be based on the Domain Model used in the software - hence the need for it to be rigorous, since software doesn't cope well with ambiguity.*

-- Martin Fowler, 2006



1. List the Semantic Descriptors

id (unique string)

title (string)

completed (true/false)

CRUD (create, read, update, delete)

listAll tasks

listActive (completed=false)

listCompleted (completed=true)

findByName (partial match)



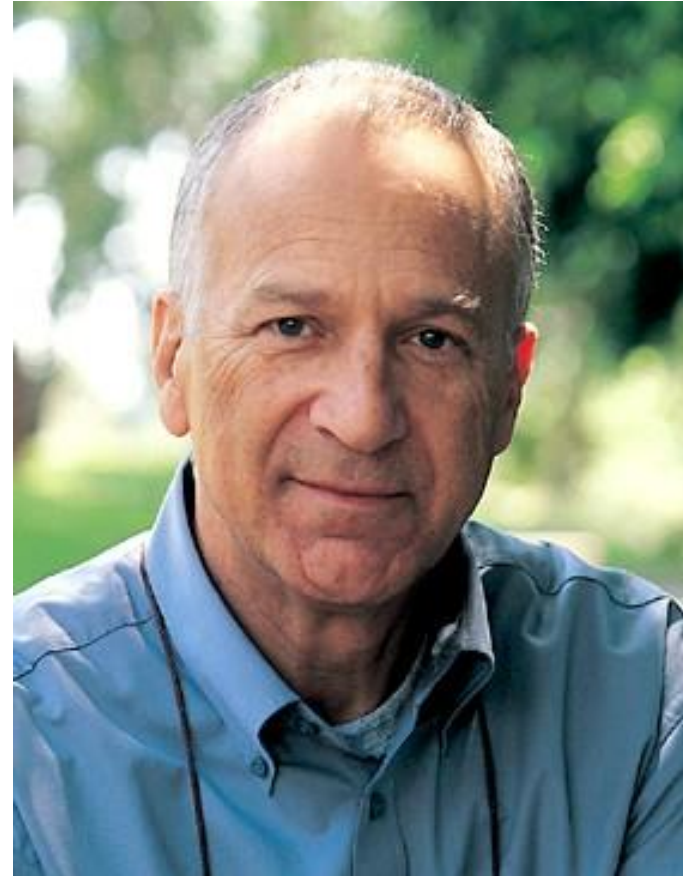
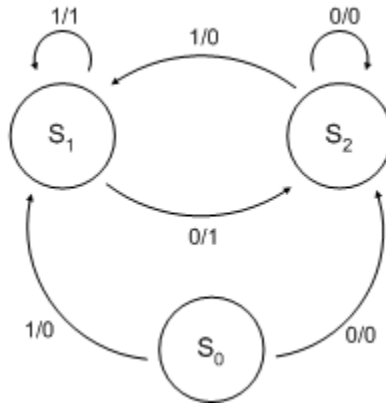
2. Draw a State Diagram

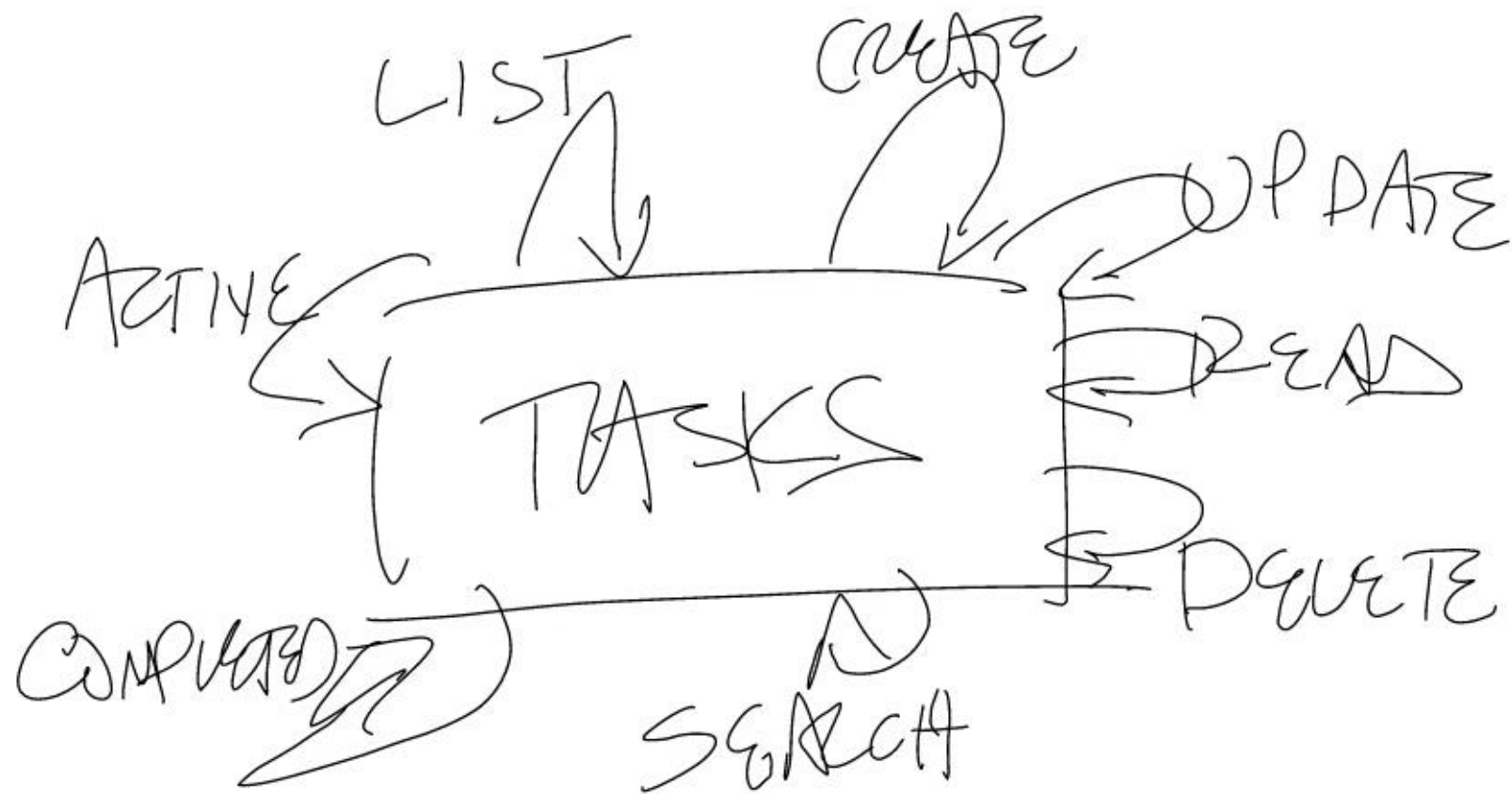


State Charts

- *State Charts are compact and expressive – small diagrams can express complex behavior – as well as compositional and modular.*

-- David Harel, 2002





3. Reconcile Names



3. Reconcile Names

IANA Link Relation Values

schema.org

microformats

Dublin Core

Activity Streams



3. Reconcile Names

id

title

completed

Create, read, update, delete

listAll

listActive

listCompleted

findByName



3. Reconcile Names

- id (data)
 - title (data)
 - completed (data)
- Create, read, update, delete
listAll – IANA collection
listActive – IANA collection
listCompleted IANA collection
findByName – search



OK, that was the design part...



But I still need to implement it, right?



4. Choose a Media Type



4. Choose a Media Type

Use application/json, application/xml

Collection type: Atom, OData, Collection+JSON

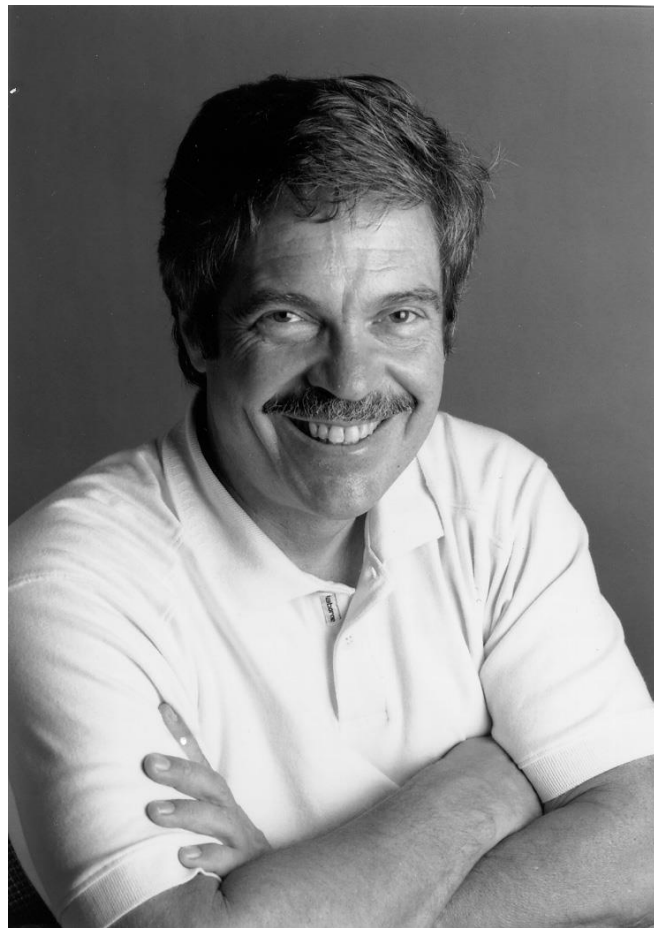
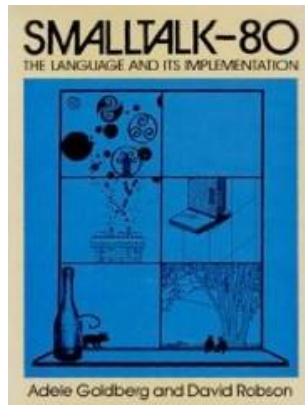
Free-form: HTML, Siren, HAL, JSON-LD

Invent your own semantic type



Message-Passing

- *The big idea is "messaging" - that is what the kernal of Smalltalk/Squeak is all about The key in making great and grow-able systems is much more to design how its modules communicate rather than what their internal properties and behaviors should be.*
- -- Alan Kay, 1998



4. Choose a Media Type

Use application/json, application/xml

Collection type: Atom, OData, Collection+JSON

Free-form: HTML, Siren, HAL, JSON-LD

Invent your own semantic type



4. Choose a Media Type

- ~~Use application/json, application/xml~~
Collection type: Atom, OData, Collection+JSON
Free-form: HTML, Siren, HAL, JSON-LD
Invent your own semantic type



4. Choose a Media Type

- ~~Use application/json, application/xml~~
Collection type: Atom, OData, Collection+JSON
Free-form: HTML, Siren, HAL, JSON-LD
- ~~Invent your own semantic type~~



4. Choose a Media Type

- ~~Use application/json, application/xml~~
Collection type: Atom, OData, Collection+JSON
- ~~Free form: HTML, Siren, HAL, JSON-LD~~
- ~~Invent your own semantic type~~



4. Choose a Media Type

- ~~Use application/json, application/xml~~
Collection type: Atom, OData, **Collection+JSON**
- ~~Free form: HTML, Siren, HAL, JSON-LD~~
- ~~Invent your own semantic type~~



Media Types

Last Updated

2014-06-10

Registration Procedure(s)

Expert Review for Vendor and Personal Trees.

Expert(s)

Ned Freed, primary; Mark Baker, secondary; Bjoern Hoehrmann, secondary

Reference

[\[RFC6838\]](#)[\[RFC4855\]](#)

Note

Per Section 3.1 of [\[RFC6838\]](#), Standards Tree requests made through IETF documents will be reviewed and approved by the IESG, while requests made by other recognized standards organizations will be reviewed by the Designated Expert in accordance with the Specification Required policy. IANA will verify that this organization is recognized as a standards organization by the IESG.

Note

[\[RFC2046\]](#) specifies that Media Types (formerly known as MIME types) and Media Subtypes will be assigned and listed by the IANA.

Name	Template	Reference
1d-interleaved-parityfec	application/1d-interleaved-parityfec	[RFC6015]
3gpdash-qoe-report+xml	application/3gpdash-qoe-report+xml	[Ozgur_Qyman][ThreeGPP]
3gpp-ims+xml	application/3gpp-ims+xml	[John_M_Meredith]
activemessage	application/activemessage	[Ehud_Shapiro]
activemessage	application/activemessage	[Ehud_Shapiro]
alto-costmap+json	application/alto-costmap+json	[RFC-ietf-alto-protocol-27]
alto-costmapfilter+json	application/alto-costmapfilter+json	[RFC-ietf-alto-protocol-27]
alto-directory+json	application/alto-directory+json	[RFC-ietf-alto-protocol-27]
alto-endpointprop+json	application/alto-endpointprop+json	[RFC-ietf-alto-protocol-27]
alto-endpointpropparams+json	application/alto-endpointpropparams+json	[RFC-ietf-alto-protocol-27]
alto-endpointcost+json	application/alto-endpointcost+json	[RFC-ietf-alto-protocol-27]
alto-endpointcostparams+json	application/alto-endpointcostparams+json	[RFC-ietf-alto-protocol-27]
alto-error+json	application/alto-error+json	[RFC-ietf-alto-protocol-27]
alto-networkmapfilter+json	application/alto-networkmapfilter+json	[RFC-ietf-alto-protocol-27]
alto-networkmap+json	application/alto-networkmap+json	[RFC-ietf-alto-protocol-27]
andrew-inset	application/andrew-inset	[Nathaniel_Borenstein]
applefile	application/applefile	[Patrik_Faltstrom]
atom+xml	application/atom+xml	[RFC4287][RFC5023]
atomcat+xml	application/atomcat+xml	[RFC5023]
atomdeleted+xml	application/atomdeleted+xml	[RFC6721]
atomicmail	application/atomicmail	[Nathaniel_Borenstein]
atomsvc+xml	application/atomsvc+xml	[RFC5023]
auth-policy+xml	application/auth-policy+xml	[RFC4745]
bacnet-xdd+zip	application/bacnet-xdd+zip	[ASHRAE][Dave_Robin]
batch-SMTP	application/batch-SMTP	[RFC2442]
beep+xml	application/beep+xml	[RFC3080]
calendar+json	application/calendar+json	[RFC7265]
calendar+xml	application/calendar+xml	[RFC6321]
call-completion	application/call-completion	[RFC6910]
cals-1840	application/cals-1840	[RFC1895]
cbor	application/cbor	[RFC7049]
ccmp+xml	application/ccmp+xml	[RFC6503]

Collection+JSON

```
{  
  "collection" : {  
    "version" : "1.0",  
    "href" : "http://amundsen.com/examples/mazes/2d/five-by-five/4:south",  
    "links" : [  
      {"rel" : "current", "href" : "http://amundsen.com/examples/mazes/2d/five-by-five/4:south"},  
      {"rel" : "north", "href" : "http://amundsen.com/examples/mazes/2d/five-by-five/3:north"},  
      {"rel" : "east", "href" : "http://amundsen.com/examples/mazes/2d/five-by-five/9:east"}  
    ]  
  }  
}
```



5. Write a Profile



Profiles RFC

- *A profile allows clients to learn about additional semantics that are associated with the resource.*

-- Erik Wilde, 2013



Independent Submission
Request for Comments: 6906
Category: Informational
ISSN: 2070-1721

E. Wilde
EMC Corporation
March 2013

The 'profile' Link Relation Type

Abstract

This specification defines the 'profile' link relation type that allows resource representations to indicate that they are following one or more profiles. A profile is defined not to alter the semantics of the resource representation itself, but to allow clients to learn about additional semantics (constraints, conventions, extensions) that are associated with the resource representation, in addition to those defined by the media type and possibly other mechanisms.

```
1 <alps>
2   <doc>
3     Sample ALPS document for TASK APP
4   </doc>
5
6   <!-- data elements -->
7   <descriptor id="id" type="semantic" />
8   <descriptor id="title" type="semantic" />
9   <descriptor id="completed" type="semantic" />
10
11  <!-- CRUD actions -->
12  <descriptor id="createTask" type="unsafe">
13    <descriptor href="#title" />
14    <descriptor href="#completed" />
15  </descriptor>
16
17  <descriptor id="readTask" type="safe">
18    <descriptor href="#id" />
19  </descriptor>
20
21  <descriptor id="updateTask" type="idempotent">
22    <descriptor href="#id" />
23    <descriptor href="#title" />
24    <descriptor href="#completed" />
25  </descriptor>
26
```

6. Implementation



6. Implementation

ta-da!



And now we have running code!



Wait, what's step seven?



7. Publication



7. Publication

Publish your "billboard" URL

Publish your profile

Register new rel values and/or media types

Publish the documentation

Consider "well-known" URIs



Now, you're done!



Seven Simple Steps

List the Semantic Descriptors

Draw a State Diagram

Reconcile Names

Write a Profile

Select a Media Type

Implementation

Publication



API Design Methodology

