API Design Methodology

Mike Amundsen,
CA / Layer7
@mamund

Introduction

LAYER 7

TECHNOLOGIES

www.apiacademy.co

API ACADEMY
LAYER 7

Search API Academy

API Strategy ▾   API Design ▾   API Management ▾   Resources ▾   About ▾

Register   Sign In

Window Snip

Your Guide to API Design & Implementation Best Practices

API Academy delivers free online lessons and in-person consulting services covering essential API techniques and tools for business managers, interface designers and enterprise architects

API Academy Overview

LAYER 7
API ACADEMY

Your Guide to API Design & Implementation Best Practices

0:00 / 1:15   YouTube

What is an API?

Get an overview of what an API is and what it does, to help you realize the business value of APIs

API Design Basics

Understand the API architecture process and learn basic design and implementation best practices

Web API Architectural Styles

Get a detailed overview of the main architectural styles for Web and mobile API design

Choosing a Solution

Choose between the various solutions that offer the basic components for enterprise API Management

Creating Evolvable Hypermedia Applications

Building

Hypermedia APIs with HTML5 & Node

Mike Amundsen

Services for a Changing World

RESTful Web APIs

Leonard Richardson,
Mike Amundsen & Sam Ruby

O'REILLY®

O'REILLY®

Designing APIs for the Web

Mike Amundsen

VIDEO

meth·od·ol·o·gy

/ˌmeTHəˈdäləjē/ 🔊

noun

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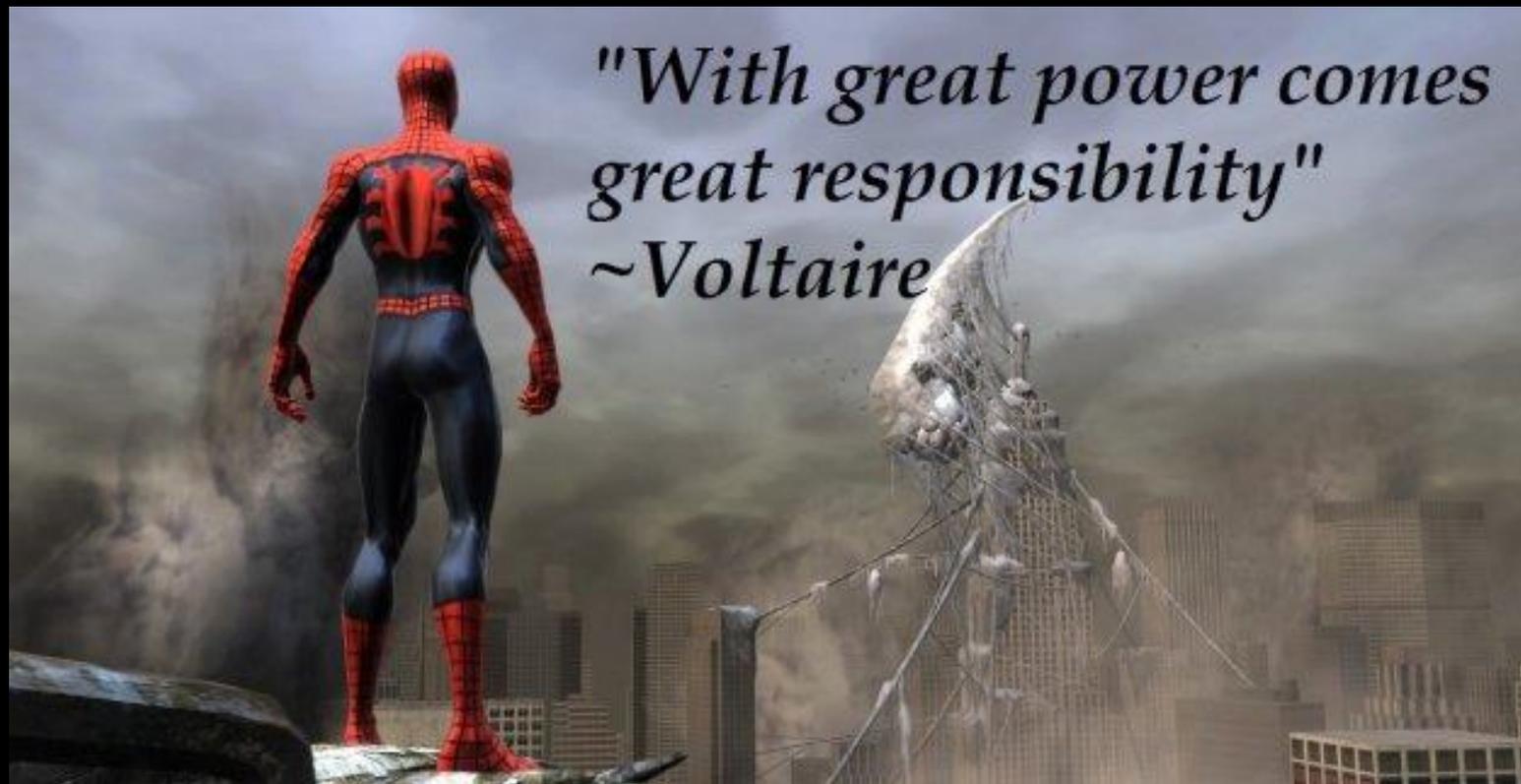"a methodology for investigating the concept of focal points"

It's so fine and yet so terrible to stand in front of a blank canvas.

Paul Cezanne

I enjoy the freedom of the blank page.

Irvine Welsh

"With great power comes great responsibility"
~Voltaire

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 *directory* 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.
Docroot	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
```

REST API

Design Rulebook

Mark Massé

Solutions for Improving Scalability and Simplicity

RESTful
Web Services
Cookbook

But there's a problem here...

Those are not *design* guidelines..

They are *implementation* guidelines!

im·ple·men·ta·tion

/ˌimpləmənˈtāSHən/ 🔊

noun

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

Ok, so what is a
design methodology, then?

de·sign

/dəˈzīn/ 🔊

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

Leonard Richardson,
Mike Amundsen & Sam Ruby

O'REILLY®

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

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

Let's design a Maze game API

1. List the Semantic Descriptors

1. List the Semantic Descriptors
(the what?)

1. List the Semantic Descriptors *(the what?)*
You know, the *stuff!*

1. List the Semantic Descriptors

- A maze
- A maze cell
- A switch
- Switch position ("up" or "down")
- The title of a maze cell
- A doorway connecting to cells
- An exit from the maze
- A list of mazes

- A list of mazes
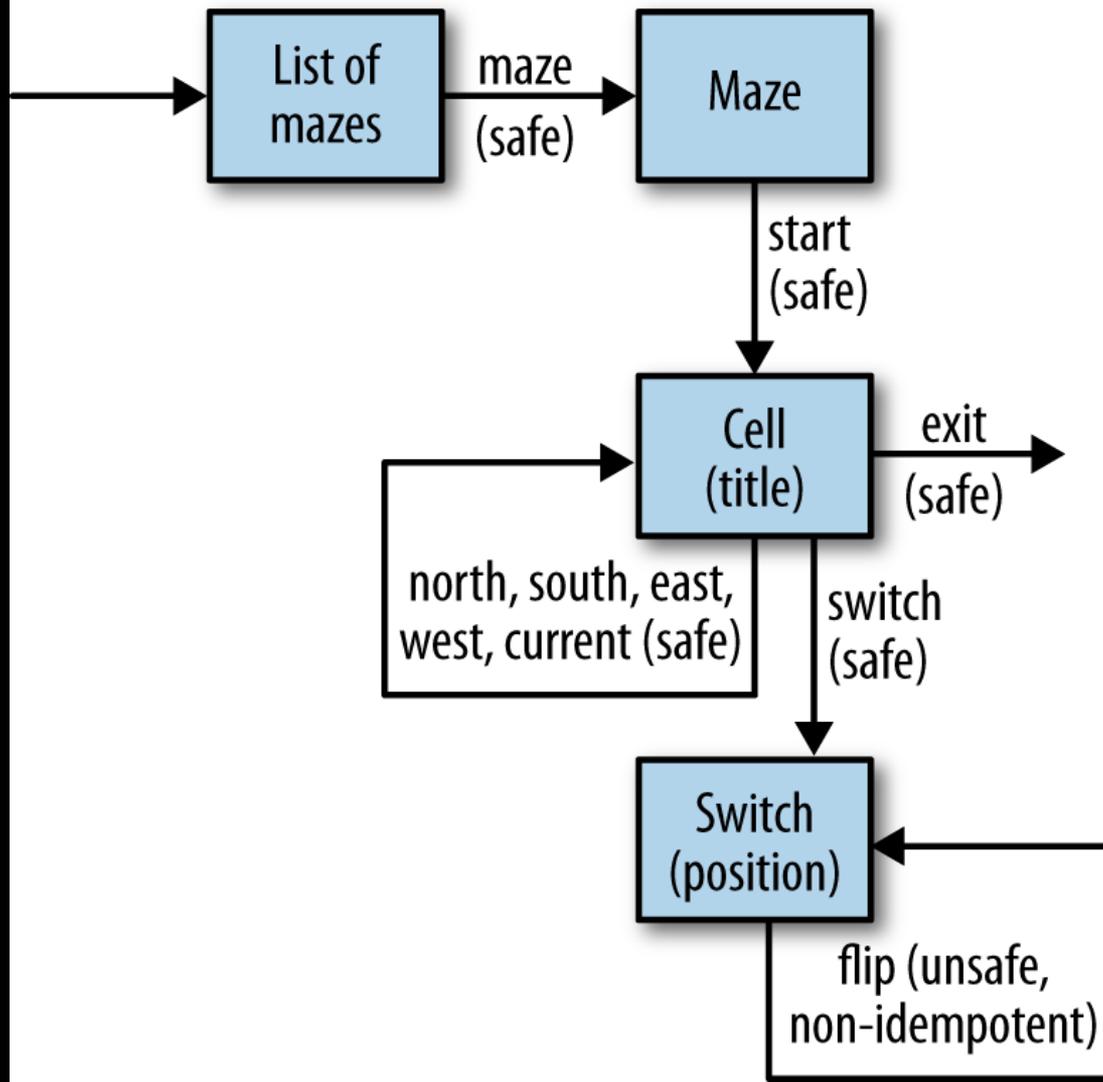 - A maze
    - A maze cell
      - A title
      - A doorway to another cell
      - An exit from the current maze
      - A switch
        - A position ("up" or "down")

2. Draw a State Diagram

3. Reconcile Names

3. Reconcile Names

- IANA Link Relation Values
- schema.org
- microformats
- Dublin Core
- Activity Streams

3. Reconcile Names

- maze
- start
- current
- exit
- north, south, east, west
- switch
- flip

3. Reconcile Names

- maze
- start (IANA)
- current (IANA)
- exit
- north, south, east, west
- switch
- flip

3. Reconcile Names

- maze
- start (IANA)
- current (IANA)
- exit (microformats)
- north, south, east, west (microformats)
- switch
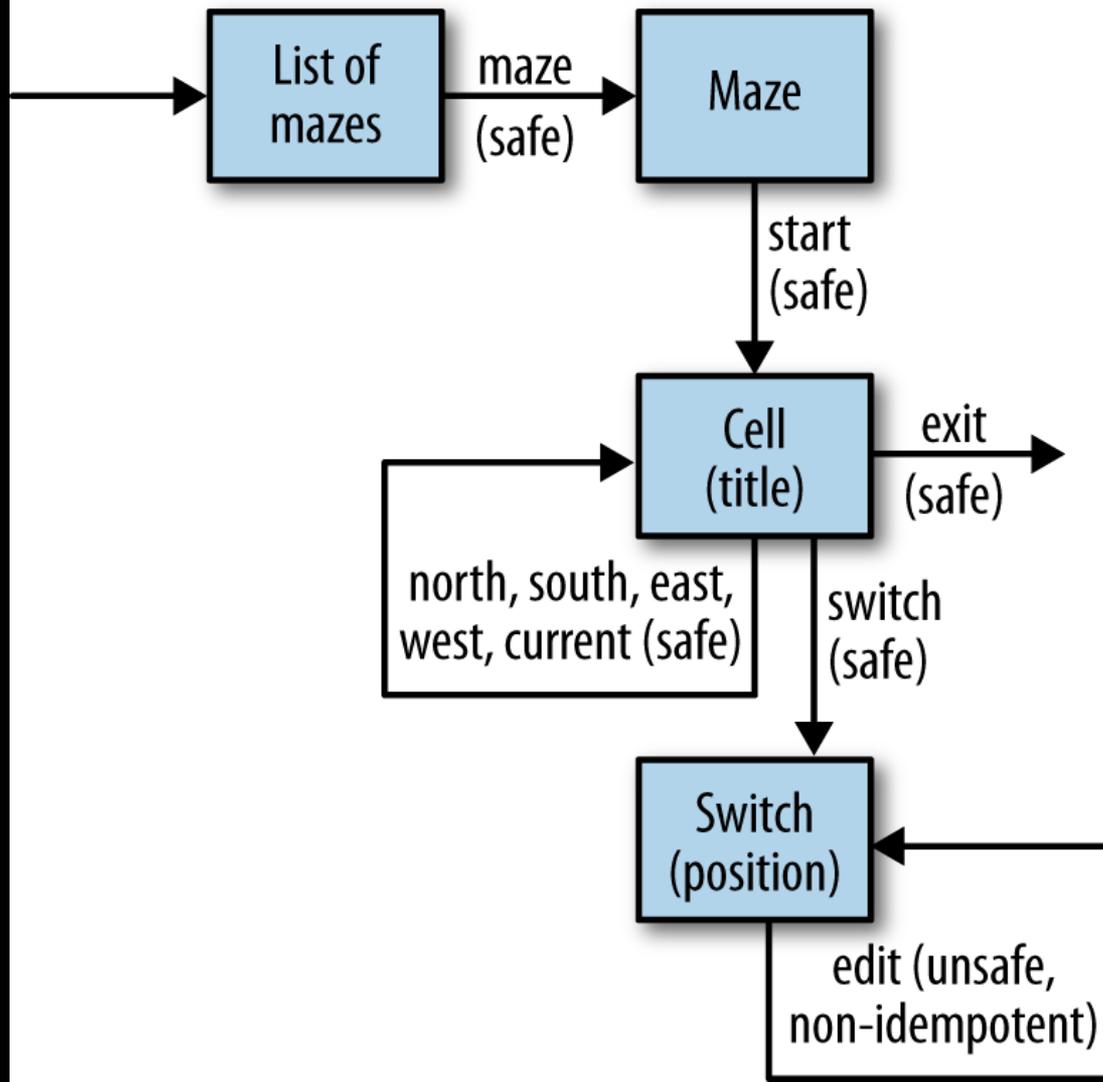- flip

3. Reconcile Names

- maze
- start (IANA)
- current (IANA)
- exit (microformats)
- north, south, east, west (microformats)
- switch
- ~~flip~~ edit (IANA)

3. Reconcile Names

- http://mamund.com/rels/maze (RFC5988)
- start (IANA)
- current (IANA)
- exit (microformats)
- north, south, east, west (microformats)
- http://mamund.com.rels/switch (RFC5988)
- ~~flip~~ edit (IANA)

3. Reconcile Names

- IANA
  - edit
  - start
  - current
- microformats
  - exit
  - north, south, east, west
- RFC5988
  - http://mamund.com/rels/switch
  - http://mamund.com/rels/maze

```
┌──────────┐   maze        ┌──────────┐
│ List of  │   (safe)      │   Maze   │
│  mazes   │ ───────────▶  │          │
└──────────┘               └──────────┘
                                 │ start
                                 │ (safe)
                                 ▼
                           ┌──────────┐   exit
          ┌──────────────▶ │   Cell   │ ─────────▶
          │                │ (title)  │   (safe)
          │                └──────────┘
          │                     │ switch
  north, south, east,           │ (safe)
  west, current (safe)          ▼
                           ┌──────────┐
                           │  Switch  │ ◀──────┐
                           │(position)│        │
                           └──────────┘        │
                                │ edit (unsafe,│
                                │ non-idempotent)
                                └─────────────┘
```

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

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_Oyman][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]

5. Write a Profile

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.

```xml
<alps version="1.0">
  <link href="http://amundsen.com/media-types/maze/" rel="help" />

  <!-- semantic descriptors -->
  <descriptor id="maze" type="safe" def="RFC5988"/>
  <descriptor id="switch" type="safe" def="RFC5988" />

  <descriptor id="edit" type="safe" def="http://www.iana.org/assignments/link-relations/"/>
  <descriptor id="start" type="safe" def="http://www.iana.org/assignments/link-relations/"/>
  <descriptor id="current" type="safe" def="http://www.iana.org/assignments/link-relations/"/>

  <descriptor id="exit" type="safe" def="http://microformats.org/wiki/existing-rel-values" />
  <descriptor id="north" type="safe" def="http://microformats.org/wiki/existing-rel-values" />
  <descriptor id="south" type="safe" def="http://microformats.org/wiki/existing-rel-values" />
  <descriptor id="west" type="safe" def="http://microformats.org/wiki/existing-rel-values" />
  <descriptor id="east" type="safe" def="http://microformats.org/wiki/existing-rel-values" />
</alps>
```

6. Implementation

6. Implementation

ta-da!

```csharp
public void ProcessRequest(HttpContext context)
{
  ctx = context;
  string method = ctx.Request.HttpMethod.ToLower();

  try
  {
    switch (method)
    {
      case "get":
        Get();
        break;
      case "head":
        Get(true);
        break;
      case "post":
        Post();
        break;
      case "options":
        Options();
        break;
      default:
        Options(405);
        break;
    }
  }
  catch (HttpException hex)
  {
    ctx.Response.ContentType = "text/plain";
    ctx.Response.StatusCode = hex.GetHttpCode();
    ctx.Response.StatusDescription = hex.Message;
    ctx.Response.Write(string.Format("{0} - {1}", hex.GetHttpCode(), hex.Message));
  }
  catch (Exception ex)
  {
    ctx.Response.ContentType = "text/plain";
    ctx.Response.StatusCode = 500;
    ctx.Response.StatusDescription = ex.Message;
    ctx.Response.Write(string.Format("{0} - {1}", 500, (debug==true?ex.StackTrace:ex.Message)));
  }
}
```

```javascript
// handle CORS OPTIONS call
if(req.method==='OPTIONS') {
    var body = JSON.stringify(headers);
    showResponse(req, res, body, 200);
}


// only accept GETs
if(req.method!=='GET') {
    showError(req, res, 'Method Not Allowed', 405);
}
else {
    // route to handle requests
    switch(parts.length) {
        case 0:
            showCollection(req, res);
            break;
        case 1:
            showMaze(req, res, parts[0]);
            break;
        case 2:
            showCell(req, res, parts[0], parts[1]);
            break;
        default:
            showError(req, res, 'Not Found', 404);
            break;
    }
}
```

```javascript
// check for exit link
href = findLink(links,'exit');
if(href) {
    m.done = true;
    console.log(m.winner.replace('{m}',m.moves));
    return;
}

// check for entrance link
if(m.start===false) {
    href = findLink(links, 'start');
    if(href) {
        flag = true;
        m.start = true;
        m.facing = 'north';
        console.log(m.moves++ + ': I\'m starting the ' + room + '. My first move is:' + href)
    }
    // ok, see if we can find a maze link
    if(href===undefined) {
        href = findLink(links, 'maze');
    }
    // well, is there a collection link?
    if(href===undefined) {
        href = findLink(links, 'collection');
    }
    // ok, i give up!
    if(href===undefined) {
        console.log(m.quitter);
        return;
```

The Game

```
12: Congratulations! you've made it out of the maze!
11:east
10:south
9:south
8:west
7:south
6:east
5:east
4:south
3:east
2:east
1:start
```

You are in the: **The Exit**.
You have the following options: **maze, clear**

What would you like to do?

[   ] Go

```
33: *** Done! http://amundsen.com/examples/mazes/2d/five-by-five/999:east
32: http://amundsen.com/examples/mazes/2d/five-by-five/24:east
31: http://amundsen.com/examples/mazes/2d/five-by-five/19:east
30: http://amundsen.com/examples/mazes/2d/five-by-five/14:south
29: http://amundsen.com/examples/mazes/2d/five-by-five/13:east
28: http://amundsen.com/examples/mazes/2d/five-by-five/8:north
27: http://amundsen.com/examples/mazes/2d/five-by-five/9:east
26: http://amundsen.com/examples/mazes/2d/five-by-five/4:south
25: http://amundsen.com/examples/mazes/2d/five-by-five/3:south
24: http://amundsen.com/examples/mazes/2d/five-by-five/2:west
23: http://amundsen.com/examples/mazes/2d/five-by-five/7:south
22: http://amundsen.com/examples/mazes/2d/five-by-five/6:east
21: http://amundsen.com/examples/mazes/2d/five-by-five/1:west
20: http://amundsen.com/examples/mazes/2d/five-by-five/6:north
19: http://amundsen.com/examples/mazes/2d/five-by-five/7:west
18: http://amundsen.com/examples/mazes/2d/five-by-five/12:east
17: http://amundsen.com/examples/mazes/2d/five-by-five/7:east
16: http://amundsen.com/examples/mazes/2d/five-by-five/2:north
15: http://amundsen.com/examples/mazes/2d/five-by-five/3:north
14: http://amundsen.com/examples/mazes/2d/five-by-five/4:west
13: http://amundsen.com/examples/mazes/2d/five-by-five/9:south
12: http://amundsen.com/examples/mazes/2d/five-by-five/8:west
11: http://amundsen.com/examples/mazes/2d/five-by-five/13:north
10: http://amundsen.com/examples/mazes/2d/five-by-five/14:west
9: http://amundsen.com/examples/mazes/2d/five-by-five/19:south
8: http://amundsen.com/examples/mazes/2d/five-by-five/18:south
7: http://amundsen.com/examples/mazes/2d/five-by-five/17:west
6: http://amundsen.com/examples/mazes/2d/five-by-five/22:south
5: http://amundsen.com/examples/mazes/2d/five-by-five/21:east
4: http://amundsen.com/examples/mazes/2d/five-by-five/16:east
3: http://amundsen.com/examples/mazes/2d/five-by-five/11:south
2: http://amundsen.com/examples/mazes/2d/five-by-five/10:east
1: http://amundsen.com/examples/mazes/2d/five-by-five/5:east
0: http://amundsen.com/examples/mazes/2d/five-by-five/0:north
```

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
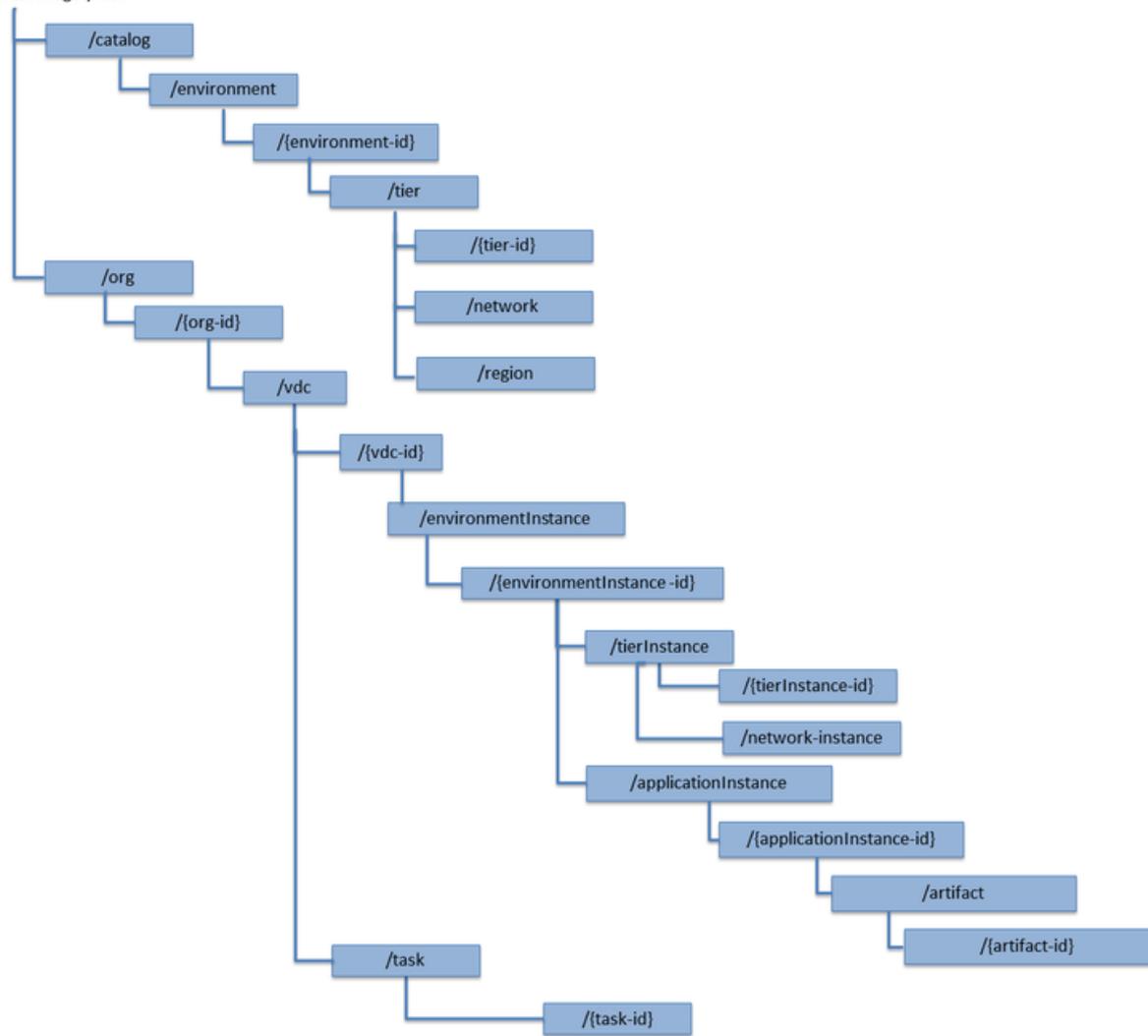
Now, *you're done!*

Seven Simple Steps

1. List the Semantic Descriptors
2. Draw a State Diagram
3. Reconcile Names
4. Write a Profile
5. Select a Media Type
6. Implementation
7. Publication

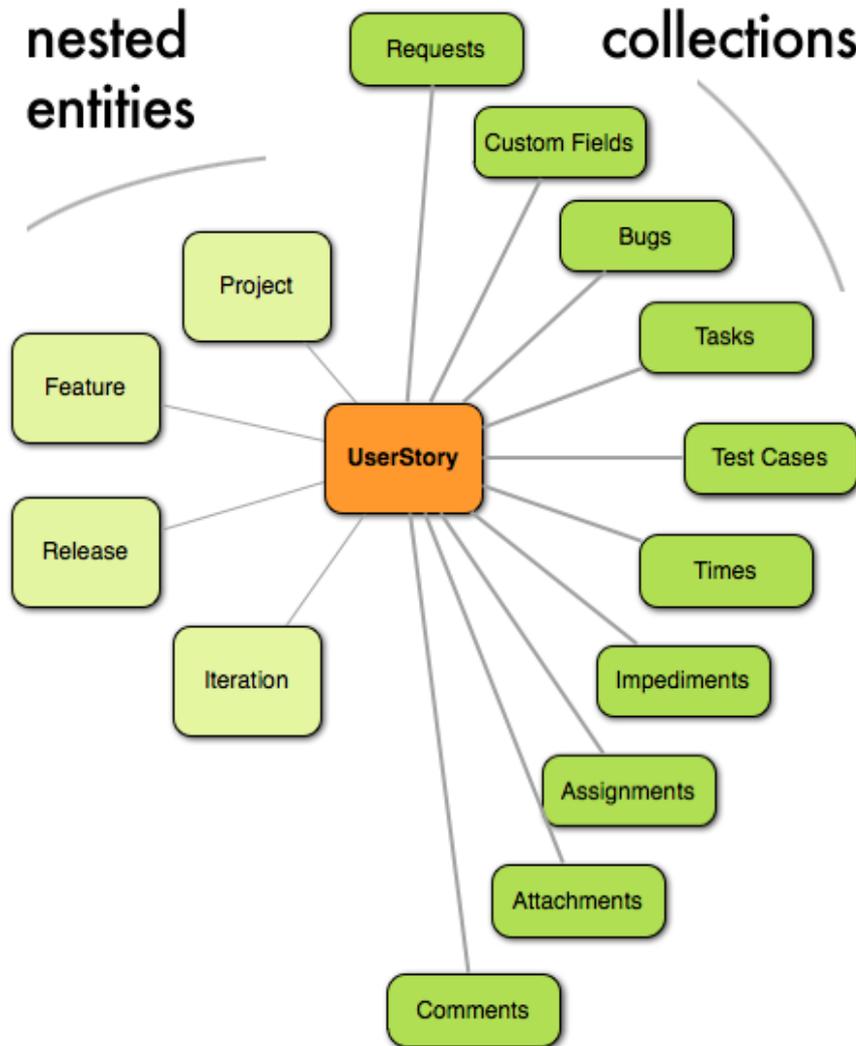# Some Final Advice

Resources are an implementation detail

Don't fall into the collection trap

nested entities

collections

Requests

Custom Fields

Bugs

Project

Tasks

Feature

UserStory

Test Cases

Release

Times

Iteration

Impediments

Assignments

Attachments

Comments

Don't start with the representation format

```json
{ "collection" :
  {
    "version" : "1.0",
    "href" : "http://example.org/friends/",

    "links" : [
      {"rel" : "feed", "href" : "http://example.org/friends/rss"},
      {"rel" : "queries", "href" : "http://example.org/friends/?queries"},
      {"rel" : "template", "href" : "http://example.org/friends/?template"}
    ],

    "items" : [
      {
        "href" : "http://example.org/friends/jdoe",
        "data" : [
          {"name" : "full-name", "value" : "J. Doe", "prompt" : "Full Name"},
          {"name" : "email", "value" : "jdoe@example.org", "prompt" : "Email"}
        ],
        "links" : [
          {"rel" : "blog", "href" : "http://examples.org/blogs/jdoe", "prompt" : "Blog"},
          {"rel" : "avatar", "href" : "http://examples.org/images/jdoe", "prompt" : "Avatar", "render" : "image"}
        ]
      }
    ]
  }
}
```

URL design doesn't matter

```
http://services.odata.org/OData/OData.svc
_____/
                    |
            service root URL


http://services.odata.org/OData/OData.svc/Category(1)/Products?$top=2&$orderby=name

_____/ _____/ _____/
                 |                            |                    |
         service root URL              resource path        query options
```

Standard names are probably better than yours.

Don't keep all the hypermedia in one place

```xml
<?xml version="1.0" encoding="
<definitions name="AktienKurs":
  targetNamespace="http://loca
  xmlns:xsd="http://schemas.xmlsoap.or
  xmlns="http://schemas.xmlsoap.org/wsd
  <service name="AktienKurs">
    <port name="AktienSoapPort" binding
      <soap:address location="http://loc
    </port>
    <message name="Aktie.HoleWert">
      <part name="body" element="xsd:Tra
    </message>
    …
  </service>
</definitions>
```

WSDL

```
Blog entry {
         _id: 1
        _rev: 2-36d81850033da710262c21462f293703
        body: JSON Schema (application/schema+json) has several purposes, one
              instance validation. The validation process may be interactive o
              interactive. For instance, applications may use JSON Schema to b
              interface enabling interactive content generation in addition to
              checking, or validate data retrieved from various sources. This
              describes schema keywords dedicated to validation purposes.
      schema: blog
       title: JSON Schema
      author:  Author  Mike
    comments: [
                 Comment   comment  {
                    author:  Author  Mike
                        id: 2
                     title: json
                 }
               ]
        date: Invalid Date
        tags: [
                 Tag  json
                 Tag  jsonschema
```

Some Final Advice

- Resources are implementation details
- Don't fall into the collection trap
- Don't start w/ the representation format
- URL design doesn't matter
- Standard names are probably better than yours
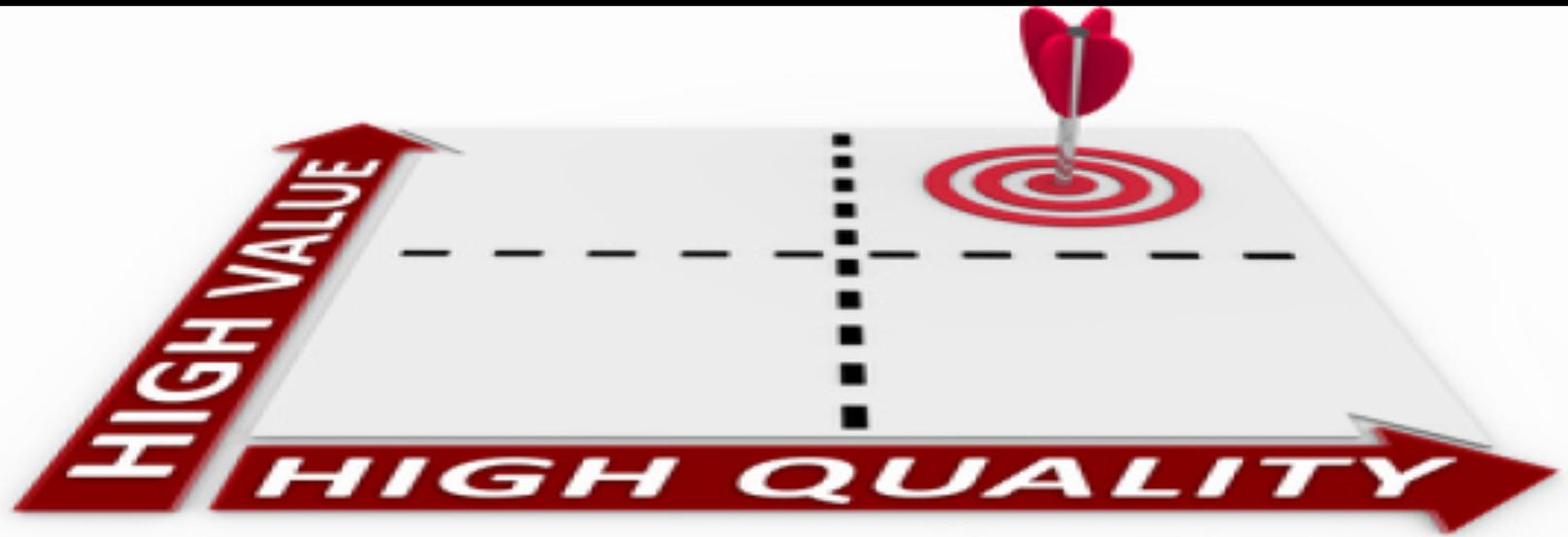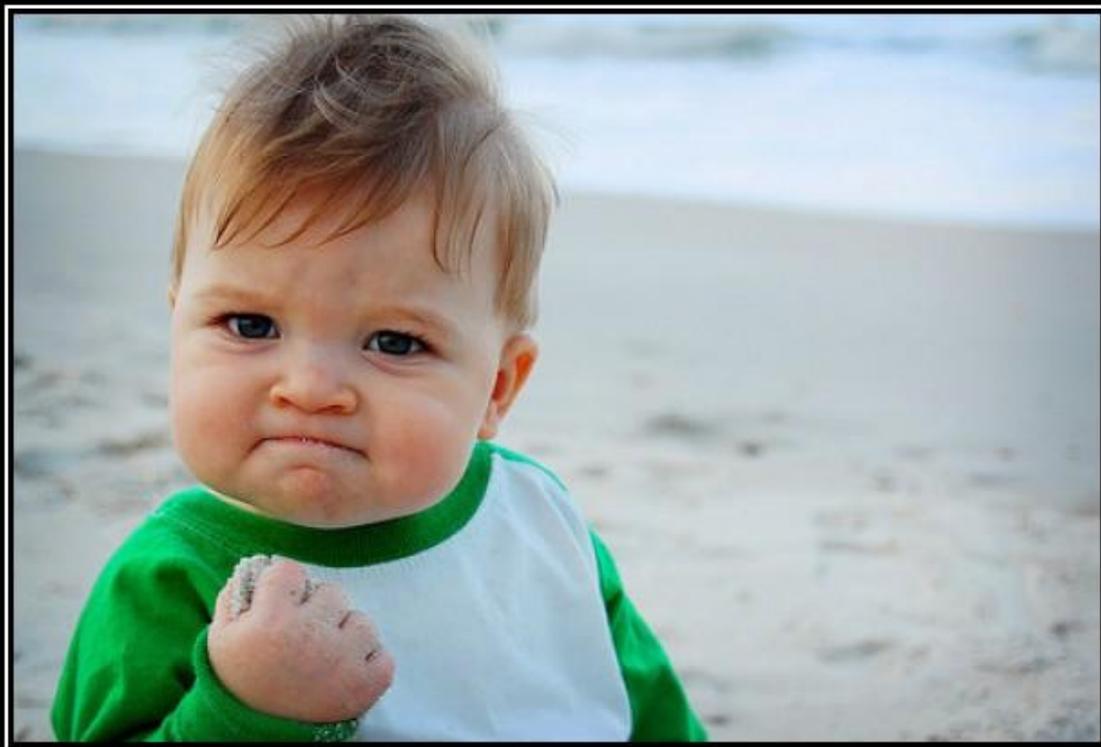- Don't keep all the hypermedia in one place

In Conclusion...

STOP

WRONG
WAY

DO NOT
ENTER

SUCCESS

Because you too can own this face of pure accomplishment

In Conclusion...

- Don't confuse implementation w/ design
- Design is the hard part (high value)
- Implementation is the easy part (high speed)
- Avoid common design mistakes
- Go out and make lots of APIs!

API Design Methodology

Mike Amundsen,
CA / Layer7
@mamund