RESTful API Design
Second Edition

Brian Mulloy
@landlessness

Apigee
@apigee

11.03.11 @ 11:03:11 PST
Dial-in or VoIP
See Details in Chat Window
API Craft

Webinar: RESTful API Design, Second Edition

Brian Mulloy View profile.
This thread is for follow-up questions to the November 3rd webinar on RESTful API design.
(webinar details here: http://co/32GUAn7)

Members

About this group
Edit my membership
Group settings
Management tasks
Invite members

Sponsored links

Create a group - Google Groups - Google Home - Terms of Service - Privacy Policy
©2011 Google
youtube.com/apigee
To be a RESTafarian or a Pragmatist?
What is a RESTafarian?
Nov 12th, 2006 | Programming, Web

A RESTifarian is a zealous proponent of the REST software architectural style as defined by Roy T. Fielding in Chapter 5 of his PhD. dissertation at UC Irvine. You can find RESTifarians in the wild on the REST-discuss mailing list. But be careful, RESTifarians can be extremely meticulous when discussing the finer points of REST, as I learned recently while participating on the list. ;)

Tags: people, rest, rayfielding, webservices
Application developers are *raison d'être* for APIs.
Be pragmatic.

For the benefit of application developers.
Pragmatic RESTful APIs are a design problem.
Paul Mijksenaar Design for Function Award 2011
Let’s look at puppies.
... 
/getAllDogs 
/locationVerify 
/foodNeeded 
/createRecurringDogWalk 
/giveDirectOrder 
/healthCheck 
/getRecurringDogWalkSchedule 
/getLocation 
/getDog 
/massDogParty 
/getNewDogsSince 
/getRedDogs 
/getSittingDogs 
/dogStateChangesSearch 
/replaceSittingDogsWithRunningDogs 
/saveDog 
...
<table>
<thead>
<tr>
<th>API</th>
<th>NVP</th>
<th>SOAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressVerify</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>BillOutstandingAmount</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>Callback</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>CreateRecurringPaymentsProfile</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoAuthorization</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoCapture</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoDirectPayment</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoExpressCheckoutPayment</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoNonReferencedCredit</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoReauthorization</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoReferenceTransaction</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>DoVoid</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>GetBalance</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>GetBillingAgreementCustomerDetails</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>GetExpressCheckoutDetails</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>GetRecurringPaymentsProfileDetails</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>GetTransactionDetails</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>ManageRecurringPaymentsProfileStatus</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>ManagePendingTransactionStatus</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>MassPayment</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>RefundTransaction</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>SetCustomerBillingAgreement</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>SetExpressCheckout</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>TransactionSearch</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
<tr>
<td>UpdateRecurringPaymentsProfile</td>
<td>NVP</td>
<td>SOAP</td>
</tr>
</tbody>
</table>
A puppy’s world is big.
/getAllDogs
/verifyLocation
/feedNeeded
/createRecurringWakeUp
/giveDirectOrder
/checkHealth
/getRecurringWakeUpSchedule
/getLocation
/getDog
/newDog
/getNewDogsSince
/getRedDogs
/getSittingDogs
/setDogStateTo
/replaceSittingDogsWithRunningDogs
/saveDog
...

/getAllLeashedDogs
/verifyVeterinarianLocation
/feedNeededFood
/createRecurringMedication
/doDirectOwnerDiscipline
/doExpressCheckupWithVeterinarian
/getRecurringFeedingSchedule
/getHungerLevel
/getSquirrelsChasingPuppies
/newDogForOwner
/getNewDogsAtKennelSince
/getRedDogsWithoutSiblings
/getSittingDogsAtPark
/setLeashedDogStateTo
/replaceParkSittingDogsWithRunningDogs
/saveMommaDogsPuppies
...
We are on a slippery slope.
Keep the simple things simple.
We only need two base URLs per resource.
The first is for a collection.
The second is for an element.
/dogs/1234
POST
GET
PUT
DELETE
CREATE
READ
UPDATE
DELETE
<table>
<thead>
<tr>
<th>Resource</th>
<th>POST</th>
<th>GET</th>
<th>PUT</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dogs</td>
<td>create</td>
<td>list dogs</td>
<td>replace</td>
<td>delete all</td>
</tr>
<tr>
<td></td>
<td>a new dog</td>
<td>dogs with</td>
<td>dogs with</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>new dogs</td>
<td>new dogs</td>
<td>dogs</td>
</tr>
<tr>
<td>/dogs/1234</td>
<td></td>
<td>show Bo</td>
<td>if exists</td>
<td>delete Bo</td>
</tr>
<tr>
<td></td>
<td>treat as a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>collection</td>
<td></td>
<td>update Bo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>create new</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dog in it</td>
<td></td>
<td>if not create Bo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>POST</td>
<td>GET</td>
<td>PUT</td>
<td>DELETE</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>/dogs</td>
<td>create a new dog</td>
<td>list dogs</td>
<td>replace dogs with new dogs</td>
<td>delete all dogs</td>
</tr>
<tr>
<td>/dogs/1234</td>
<td>treat as a collection create new dog in it</td>
<td>show Bo</td>
<td>if exists update Bo if not create Bo</td>
<td>delete Bo</td>
</tr>
<tr>
<td>Resource</td>
<td>POST</td>
<td>GET</td>
<td>PUT</td>
<td>DELETE</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>/dogs</td>
<td>create a new dog</td>
<td>list dogs</td>
<td>bulk update dogs</td>
<td>delete all dogs</td>
</tr>
<tr>
<td>/dogs/1234</td>
<td>error</td>
<td>show Bo</td>
<td>if exists update Bo if not error</td>
<td>delete Bo</td>
</tr>
</tbody>
</table>
Verbs are bad.
Nouns are good.
Plurals or singulars?
Foursquare
/checkins

GroupOn
/deals

Zappos
/Product
Plurals are better.

/dogs
Abstract or concrete naming?
<table>
<thead>
<tr>
<th>Level</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super High</td>
<td>/things</td>
</tr>
<tr>
<td>High</td>
<td>/animals</td>
</tr>
<tr>
<td>Medium</td>
<td>/dogs</td>
</tr>
<tr>
<td>Low</td>
<td>/beagles</td>
</tr>
</tbody>
</table>
Concrete is better than abstract.

/dogs
What about associations?
GET /owners/5678/dogs

POST /owners/5678/dogs
What about complex variations?
Sweep variations under the ‘?’
/dogs?color=red&state=running&location=park
What about errors?
Facebook

HTTP Status Code: 200

{"type":"OAuthException","message":"(#803) Some of the aliases you requested do not exist: foo.bar"}

Twilio

HTTP Status Code: 401


SimpleGeo

HTTP Status Code: 401

{"code":401,"message":"Authentication Required"}
## Code for code

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
</tr>
</tbody>
</table>


## Message for people

```json
{
    "message": "Verbose, plain language description of the problem with hints about how to fix it."
    "more_info": "http://dev.teachdogrest.com/errors/12345"
}
```
What about versioning?
Twilio

/2010-04-01/Accounts/

salesforce.com

/services/data/v20.0/sobjects/Account

Facebook

?v=1.0
/v1/dogs
Please give me exactly what I need.
LinkedIn

/people:(id,first-name,last-name,industry)

Facebook

/joe.smith/friends?fields=id,name,picture

Google (partial response)

?fields=title,media:group(media:thumbnail)
What about pagination?
Facebook

offset
limit

Twitter

page
rpp

LinkedIn

start
count
/dogs?limit=25&offset=50
What about formats?
Google Data

?alt=json

Foursquare

/venue.json

Digg*

Accept: application/json

?type=json

* The type argument, if present, overrides the Accept header.
What about defaults?
Format

json

Pagination (depends on data size)

limit=10&offset=0
What about attribute names?
Twitter

"created_at": "Thu Nov 03 05:19:38 +0000 2011"

Bing

"DateTime": "2011-10-29T09:35:00Z"

Foursquare

"createdAt": 1320296464
JSON is for JavaScript Objects

```javascript
var myObject = JSON.parse(response);
```

These looks funny in JavaScript

```javascript
timing = myObject.created_at;
timing = myObject.DateTime;
```
JavaScript Convention

"createdAt": 1320296464

timing = myObject.createdAt;
What about non-resource-y stuff?
Calculate
Translate
Convert
Use verbs not nouns

/convert?from=EUR&to=CNY&amount=100
What about searching?
Global

/search?q=fluffy+fur

Scoped

/owners/5678/dogs/search?q=fluffy+fur

Formatted

/search.xml?q=fluffy+fur
What about counts?
/dogs/count
What about the rest of the URL?
<table>
<thead>
<tr>
<th>Platform</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Facebook</strong></td>
<td><code>graph.facebook.com</code></td>
</tr>
<tr>
<td></td>
<td><code>api.facebook.com</code></td>
</tr>
<tr>
<td></td>
<td><code>developers.facebook.com</code></td>
</tr>
<tr>
<td><strong>Foursquare</strong></td>
<td><code>api.foursquare.com</code></td>
</tr>
<tr>
<td></td>
<td><code>developers.foursquare.com</code></td>
</tr>
<tr>
<td><strong>Twitter</strong></td>
<td><code>api.twitter.com</code></td>
</tr>
<tr>
<td></td>
<td><code>search.twitter.com</code></td>
</tr>
<tr>
<td></td>
<td><code>stream.twitter.com</code></td>
</tr>
<tr>
<td></td>
<td><code>dev.twitter.com</code></td>
</tr>
</tbody>
</table>
API gateway

api.teachdogrest.com

Developer connection

developers.teachdogrest.com

Do web redirects

api → developers (if from browser)

dev → developers

developer → developers
What about exceptional stuff?
Client intercepts HTTP error codes
Twitter

/public_timelines.json?
suppress_response_codes=true

HTTP Status Code: 200

{"error" : "Could not authenticate you." }
Always returns **OK**

```bash
/dogs?suppress_response_codes=true
```

**Code for **code** ignoring**

```
200 - OK
```

**Message for people & code**

```json
{"response_code": "401", "message": "Verbose, plain language description of the problem with hints about how to fix it."
"more_info": "http://dev.teachdogrest.com/ errors/12345", "code": 12345}
```
Client supports limited HTTP methods
Method Parameter

create
/dogs?method=post

read
/dogs

update
/dogs/1234?method=put&location=park

delete
/dogs/1234?method=delete
What about authentication?
PayPal
Permissions Service API

Facebook
OAuth 2.0

Twitter
OAuth 1.0a
Use latest and greatest OAuth

OAuth 2.0

Don’t do something close, but different.
How do application developers use the API?
What about chatty applications?
First be complete & RESTful.

Then provide shortcuts.
Partial response syntax can help.

/owners/5678?fields=name,dogs(name)
What about when building an UI requires a lot of domain knowledge?
firesideint @cdellaverson Felt that, loud and clear. We’re in Cap Haitien. House (really the whole world) shook for about 30 seconds.

From Haiti
Complement your API with code libraries and SDK.
Really? All of this? And iterate it?
API Virtualization Layer

Application

API
API
API
Be RESTful
Only 2 URLs
No verbs
Use nouns as plurals
Concrete over abstract
For JSON follow JavaScript conventions
Sweep complexity behind the ‘?’
Borrow from leading APIs
Account for exceptional clients
Add virtualization layer
Questions?
THANK YOU

Subscribe to API webinars at:
youtube.com/apigee
THANK YOU

Questions and ideas to:

groups.google.com/group/api-craft
THANK YOU

Contact me at:
@landlessness
brian@apigee.com