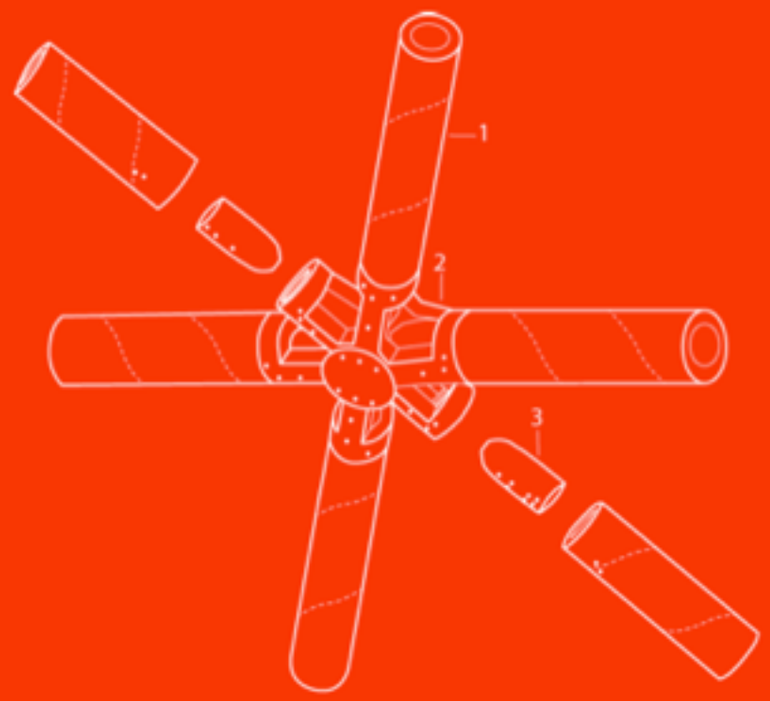
Web API Design

Crafting Interfaces that Developers Love



2014-06-20 mindwind

Pragmatic REST

- REST is an architectural style, not a strict standard.
- The success of an API design is measured by how quickly developers can get up to speed and start enjoying using your API.
- Design communicates how something will be used.

1. Nouns are good, verbs are bad

- Keep base URL simple and intuitive
  - /dogs /dogs/1234
- Keep verbs out of base URL

2. Use HTTP verbs to operate on the collections and elements

Resource	POST create	GET read	PUT update	DELETE delete
/dogs	Create a new dog	List dogs	Bulk update dogs	Delete all dogs
/dogs/1234	Error	Show Bo	If exists update Bo If not error	Delete Bo

3. plural nouns and concrete names

- It is more intuitive to use plural nouns.
- Avoid a mixed model in which you use singular for some resources and plural for others.
- Concrete names are better than abstract.
  - /dogs (concrete) ~~/animals (abstract)~~

4. Simplify association - sweep complexity under the '?'

- To get all the dogs belonging to a specific owner.
  - ~~GET /owners/5678/dogs (bad)~~
  - GET /dogs?owner=5678 (good)

5. Handling 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

{"status" : "401", "message":"Authenticate","code": 20003, "more
info": "http://www.twilio.com/docs/errors/20003"}
```

SimpleGeo

```
HTTP Status Code: 401

{"code" : 401, "message": "Authentication Required"}
```

5. Handling errors

- Use HTTP Status Code.
  - success 200 - OK
  - client error 400 - Bad Request
  - server error 500 - Internal Server Error
- Make message returned as verbose as possible.

6. Tips for versioning

- Never release an API without a version and make the version mandatory.
  - Twilio /2014-04-01/Accounts/ (Confusing)
  - Salesforce /services/v20.0/sobjects/Account (change frequently)
  - Facebook ?v=1.0 (optional not mandatory)
  - recommend /v1/dogs /v2/dogs

7. pagination and partial response

- Use limit and offset to make it familiar to developer.
  - /dogs?limit=25&offset=50
- Support partial response by adding optional fields in a comma delimited list.
  - /dogs?fields=name,color,location

8. What about responses that don't involve resources?

- Use verbs not nouns.
  - /convert?from=EUR&to=CNY&amount=1000
- Make it clear in API documentation that these "non-resource" scenarios are different.

9. Supporting multiple formats

- Use JSON as default format.
- It is friendly for javascript and JSON is the closet thing we have to universal language.
- /dogs.json /dogs/1234.json

10. What about attribute names?

- Follow javascript convention for naming attributes.
  - CamelCase

11. Tips for search

- Simple search could be modeled as a resourceful API
  - /dogs?q=red
- Global search follow the Google model
  - /search?q=red+fluffy

12. Consolidate API requests in one subdomain

- Consolidate all API requests under one subdomain
  - api.yourdomain.com
  - only one is cleaner, easier and more intuitive for developers.

13. Complement with an SDK

- Simplify integration effort required to work with your API.

- An SDK can help reduce bad or inefficient code.