API
Design and Contracts

Nikolaos Anastopoulos

About Me

I am a programmer from Athens, Greece. I like developing modern web applications in Ruby and its Rails framework.

I am working with GaggleAMP being a member of the product development team. GaggleAMP makes brand advocacy software, specializing in social media engagement and reach amplification.

I recently designed the API of our mobile app product...

Overview

- Why REST is so trendy?
- What's the fuzz with JSON API?
- API Contracts? Do I need an API lawyer?
- Open API or API Blueprint? Chocolate or strawberry?
- Any tips on how to implement the above concepts?

REST vs RPC

- The design principles acronyms REST and RPC mean Representational State Transfer and Remote Procedurally Call respectively.
- REST is used to design based on actions on resources, while RPC designs are based on operations.

REST vs RPC

- Data ownership is a good way to decide on the design principle, i.e. if API exposed operates on its own data it is quite effective to expose them as resources, while if an API operates on data given during the call, it might be more efficient to expose the procedures.

- There are many principles, more or less descriptive, but the main concepts revolve around resource vs process.

REST vs RPC Examples

Example	REST	RPC
Get tasks list of a user	GET /users/1/tasks	GET /getTasks?user_id=1
User sign in	POST /user/sessions	POST /sign_in

JSON vs JSON API

- Acronym JSON stands for JavaScript Object Notation and is used to serialize data structures.
- The JSON API specification tries to establish some conventions on JSON responses format.
- It goes a little bit further trying to generalize both the data representation and the response metadata in order to take advantage of broader tooling.

JSON vs JSON API

- However Ruby on Rails implies some kind of conventions and these are used by many people.
- It really matters who will be the consumer of the data and the interoperation levels required.

JSON vs JSON API Examples

JSON

```
Link: <http://example.com/articles?page=2>; rel="next",
<http://example.com/articles?page=10>; rel="last"
```

```
[{
  "id": "1",
  "title": "JSON API paints my bikeshed!"
  "author": {
    "id": "9",
    "first-name": "Dan",
    "last-name": "Gebhardt"
   }
}]
```

JSON API

```
{
  "links": {
    "self": "http://example.com/articles",
    "next": "http://example.com/articles?page[offset]=2",
    "last": "http://example.com/articles?page[offset]=10"
  },
  "data": [{
    "type": "articles",
    "id": "1",
    "attributes": {
      "title": "JSON API paints my bikeshed!"
    },
    "relationships": {
      "author": {
        "links": {
          "self": "http://example.com/articles/1/
relationships/author",
          "related": "http://example.com/articles/1/author"
        },
        "data": { "type": "people", "id": "9" }
      }
    },
    "links": {
      "self": "http://example.com/articles/1"
    }
  }],
  "included": [{
    "type": "people",
    "id": "9",
    "attributes": {
      "first-name": "Dan",
      "last-name": "Gebhardt"   },
    "links": {
      "self": "http://example.com/people/9"
    }
 }]
}
```

API Contract and Documentation

- API contract is similar to tests in test-driven development.
- The standards represent API contracts in a language-agnostic interface which allow both humans and computers to discover and understand the capabilities of a service (Open API Specification Introduction, v3.0.0).
- Settling on a contract before API is developed tends to lead to better API designs (API Blueprint Brochure, 2017).

API Contract and Documentation

- It greatly facilitates cross-functional teams, like back-end developers (implementing the service provider) and mobile app developers (implementing the service consumer).
- API contracts may be used as formal documentation of the service they describe.

Open API vs API Blueprint

- SmartBear in conjunction with some other large companies formed the Open API Initiative under the Linux Foundation donating its Swagger specification and renaming it to Open API.
- API Blueprint is released under MIT License by Apiary, now acquired by Oracle.

Open API vs API Blueprint

- Open API contracts are written in YAML and JSON, while API Blueprint contracts are written in MSON (Markdown Syntax for Object Notation).

- Both of these specifications are oriented towards the resource-oriented architecture.

Open API vs API Blueprint Examples

Open API

```
swagger: '2.0'
info:
  version: '1.0'
  title: The Simplest API
  description: >-
    This is one of the simplest APIs
    written in the **Open API**.
paths:
  /message:
    get:
      produces:
        - text/plain
      responses:
        '200':
          description: It is required
          schema:
            type: string
          examples:
            text/plain: >-
              Hello World!
```

API Blueprint

```
FORMAT: 1A

# The Simplest API
This is one of the simplest APIs
written in the **API Blueprint**.


# GET /message
+ Response 200 (text/plain)

        Hello World!
```

Tips on How to Write the API Contract

- Better use YAML or MSON for being easier to read and edit.
- A good idea would be to version track the API contract together with the implementation code.

Tips on How to Write the API Contract

- Might want to leverage ERB to better integrate the API contract:

```
swagger: '2.0'
info:
  title: Some subscription related API
  version: '1.0'
paths: ...
definitions:
  Subscription:
    type: object
    properties:
      ...
      tier:
        type: string
        title: Subscription Tier
        default: <%= Subscription.tiers.keys.first %>
        enum: <%= Subscription.tiers.keys %>
```

Tips on How to Serve the API Contract

- In case of API Blueprint the related MIME type should be registered in `config/initializers/mime_types.rb`:

```
Mime::Type.register "text/vnd.apiblueprint", :apib
```

Tips on How to Serve the API Contract

- A nice touch would be to use a RoR metal controller to convert Open API YAML to JSON:

```
class API::SpecificationsController < ActionController::Metal
  # The path to the API specification file in Open API format.
  SPECIFICATION_PATH = Rails.root.join('config', 'api.yml').freeze

  # GET /api/swagger.json
  def show
    self.content_type  = Mime[:json]
    self.response_body = JSON.pretty_generate \
      YAML.load(ERB.new(specification_pathname.read).result)
  end

  private

  def specification_pathname
    Pathname.new SPECIFICATION_PATH
  end
end
```

- Might also want to add some caching to the above on production environments.

Tips on How to Test against the API Contract

- For API Blueprint there is the Dredd project, a language-agnostic API testing tool. It also features dredd_hooks, a Ruby binding gem.

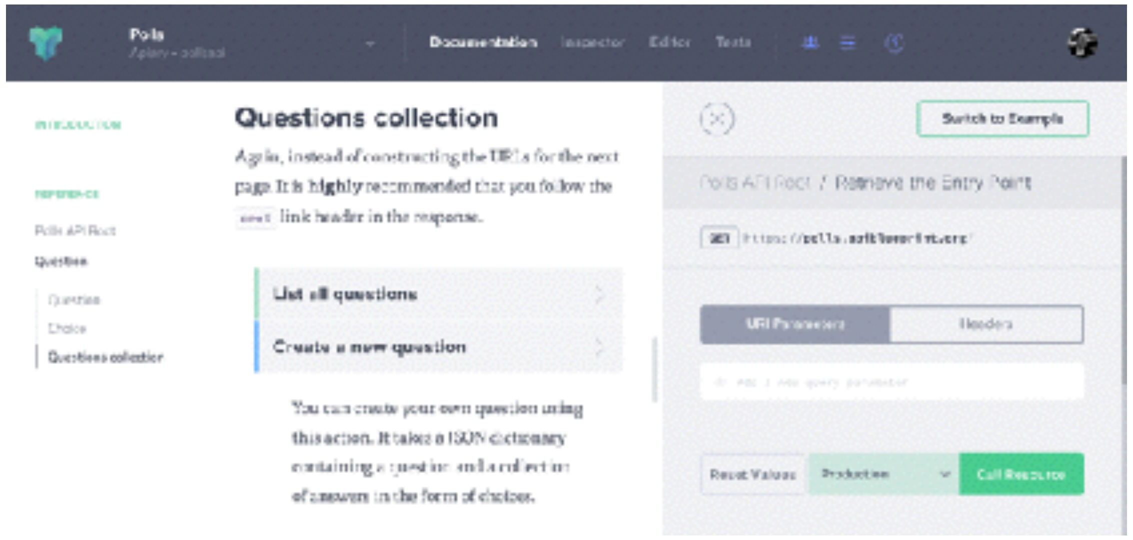
Tips on How to Test against the API Contract

- Respectively, for Open API, gem apivore extends RSpec to validate the implementation:

```
require 'spec_helper'

RSpec.describe 'The Simplest API', type: :apivore,
order: :defined do
  subject { Apivore::SwaggerChecker.instance_for('/swagger.json')
}

  context 'has valid paths' do
    it { is_expected.to validate(:get, '/message', 200, {}) }
  end

  context 'and' do
    it 'tests all documented routes' do
      expect(subject).to validate_all_paths
    end
  end
end
```

Tips on How to Read the API Documentation

- The great benefit of API contracts is that there are many tools to support each phase of the lifecycle.
- These API contracts should also be used as the API documentation.
- In case of API Blueprint Apiary works great and for Open API the Swagger UI is very good.

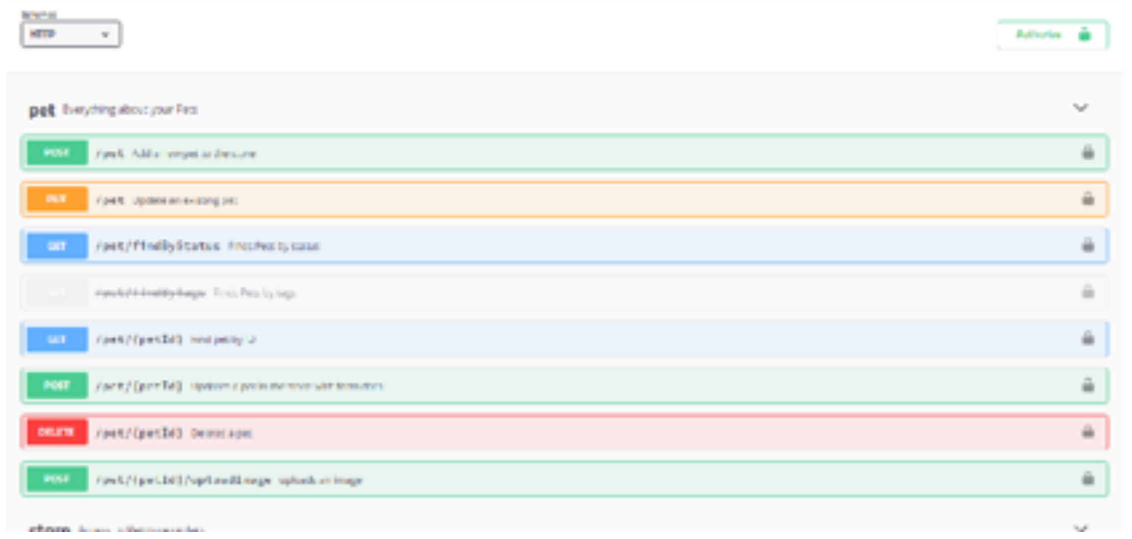


Questions collection

Again, instead of constructing the URLs for the next page. It is highly recommended that you follow the next link header in the response.

You can create your own question using this action. It takes a JSON dictionary containing a question and a collection of answers in the form of choices.

GET https://polls.apiblueprint.org/



pet Everything about your Pets

POST /pet Add a new pet to the store
PUT /pet Update an existing pet
GET /pet/findByStatus Finds Pets by status
GET /pet/findByTags Finds Pets by tags
GET /pet/{petId} Find pet by ID
POST /pet/{petId} Updates a pet in the store with form data
DELETE /pet/{petId} Deletes a pet
POST /pet/{petId}/uploadImage uploads an image

store

Thank You

Any questions?