

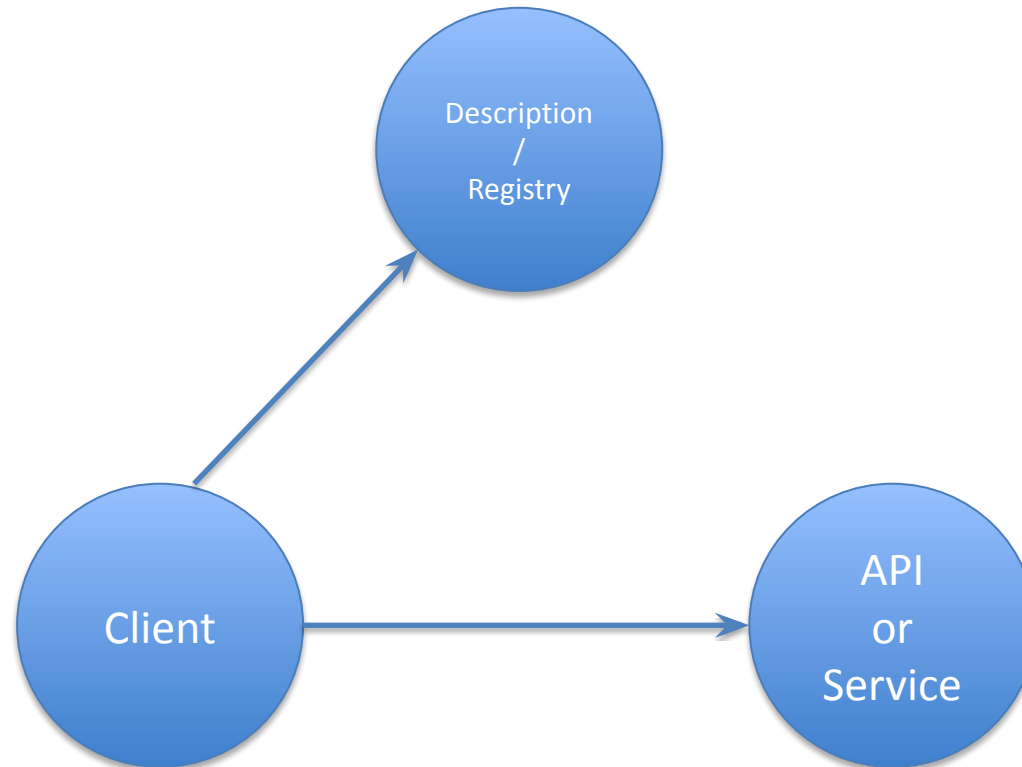
# REST description

Oxford University  
Software Engineering Programme  
April 2021



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# SOA contracts



# SOAP + WSDL

- Enabled a lot of tooling
- Very powerful
- Built in from day 1
- Not perfect
  - WSDL 1.1 vs 2.0, doc/lit, rpc/lit, rpc/encoded
  - Not always interoperable



# REST description

- Do we need it?



# 2007

**Q:** Does REST need a description document?

**A:** No.

Joe Gregorio, REST proponent, [bitworking.org](http://bitworking.org)



# More info

## Debate: Does REST Need a Description Language?

by **Arnon Rotem-Gal-Oz** on Jun 06, 2007 | **7** Discuss

Share  |      

Following up on the debate of REST vs. WS-\* that has [yet again, been discussed here last week](#) , it is interesting to note a debate on the topic of contracts for RESTful services that has been picking up pace over the last few days. The question that emerged was whether REST needs contracts (a la WSDLs) and even more fundamentally whether REST with contracts was still REST. It started when [Aristotle Pagaltzis asked "Does REST need a service description language"](#): To which [Paul Mueller responded](#) [Mark Baker followed up](#), agreeing with Aristotle that

"The other day, someone posted on rest-discuss to ask whether there was a standard way to describe a REST service. This question resurfaces from time to time (and usually draws a pointer to the [WADL](#) effort as a response), which seems to indicate a popular misunderstanding of REST. I think describing a RESTful service with a WSDL-like language is a contradiction in terms..."

<https://www.infoq.com/news/2007/06/rest-description-language>

RELATE

The R  
to Bui

The S  
Micro

Creati  
Swag

Spring  
Hvner



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# The world has moved on

- WADL (pretty much dead)
- RAML
  - [www.raml.org](http://www.raml.org)
- Swagger (aka OpenAPI)
  - [www.swagger.io](http://www.swagger.io)
  - <https://github.com/OAI/OpenAPI-Specification>
- API Blueprint
  - [www.apiblueprint.org](http://www.apiblueprint.org)



# Top down or bottom up

- Just like WSDL, we can use these for design as well as description
- Language neutral way of creating prototype interfaces
- More in the API management section





# RAML



For every API, start by defining which version of RAML you are using, and then document basic characteristics of your API - the title, baseURI, and version.



Create and pull in namespaced, reusable libraries containing data types, traits, resource types, schemas, examples, & more.



Annotations let you add vendor specific functionality without compromising your spec



Traits and resourceTypes let you take advantage of code reuse and design patterns



Easily define resources and methods, then add as much detail as you want. Apply traits and other patterns, or add parameters and other details specific to each call.



Describe expected responses for multiple media types and specify data types or call in pre-defined schemas and examples. Schemas and examples can be defined via a data type, in-line, or externalized with !include.



Write human-readable, markdown-formatted descriptions throughout your RAML spec, or include entire markdown documentation sections at the root.

```
1 title: World Music API
2 baseUri: http://example.api.com/{version}
3 version: v1
4
5
6 uses:
7   Songs: !include libraries/songs.raml
8
9 annotationTypes:
10   monitoringInterval:
11     parameters:
12       value: integer
13
14 traits:
15   secured: !include secured/accessToken.raml
16
17 /songs:
18   is: secured
19   get:
20     (monitoringInterval): 30
21     queryParameters:
22       genre:
23         description: filter the songs by genre
24   post:
25     /{songId}:
26       get:
27         responses:
28           200:
29             body:
30               application/json:
31                 type: Songs.Song
32               application/xml:
33                 schema: !include schemas/songs.xml
34                 example: !include examples/songs.xml
```

## Songs Library

```
1 #%RAML 1.0 Library
2 types:
3   Song:
4     properties:
5       title: string
6       length: number
7   Album:
8     properties:
9       title: string
10      songs: Song[]
11   Musician:
12     properties:
13       name: string
14       discography: (Song | Album)[]
```

## songs.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3   elementFormDefault="qualified" attributeFormDefault="unqualified">
4   <xs:element name="song">
5     <xs:complexType>
6       <xs:sequence>
7         <xs:element name="title" type="xs:string"/>
8       </xs:sequence>
9     </xs:complexType>
10   </xs:element>
11   <xs:element name="artist" type="xs:string"/>
12 </xs:schema>
```

# RAML

- YAML language
  - Easy
- Industry support
  - Basically a Mule project, with some supporters
- API console, tooling, etc
  - All good



# Swagger

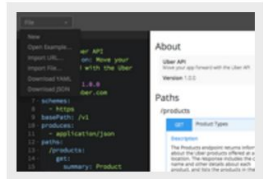
## SWAGGER TOOLS & RESOURCES

### TOOLS



#### SWAGGER UI

Use a Swagger specification to drive your API documentation. [Demo](#) and [Download](#).



#### SWAGGER EDITOR

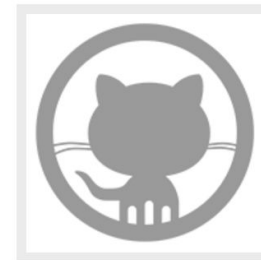
An editor for designing Swagger specifications from scratch, using a simple YAML structure. [Demo](#) and [Source](#).



#### SDK GENERATORS

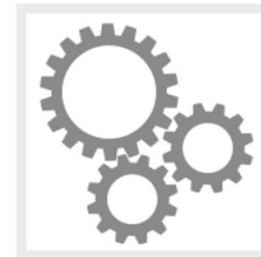
Turn an API spec into client SDKs or server-side code with [Swagger Codegen](#).

### RESOURCES



#### SERVER INTEGRATIONS

Dozens of integration options for putting Swagger in your API, from both SmartBear and the community. Pick your language, framework and [get started!](#).



#### SERVICES

From API management to Platform as a Service, there are a number of services tightly integrated with Swagger. [See more](#).

# Swagger

- Swagger is now officially “OpenAPI”
- JSON format is the standard
- Now supports YAML too
- Much wider industry support
- Nice editor for design first

<https://swagger.io/specification/>



# Swagger Editor

File Preferences Generate Server Generate Client Help

```
1 swagger: '2.0'
2 info:
3   version: 0.0.0
4   title: Course Booking Service
5 paths:
6   /courseInstances:
7     get:
8       description: |
9         Gets all `CourseInstance` objects for all courses.
10        Optional query param of size determines
11        size of returned array. Typically you would use
12        GET `/course/{courseId}/courseInstances` instead to get
13        the list of instances for a particular course
14      parameters:
15        - name: size
16          in: query
17          description: Size of array
18          required: false
19          type: number
20          format: double
21        - name: expand
22          in: query
23          description: expand objects or not
24          required: false
25          type: boolean
26      responses:
27        '200':
28          description: Successful response
29          schema:
30            title: ArrayOfCourseInstances
31            type: array
32            items:
33              $ref: '#/definitions/CourseInstance'
34      post:
35        description: Creates a new `CourseInstance` object
36        parameters:
37          - name: courseInstance
38            in: body
39            required: true
40            schema:
41              $ref: '#/definitions/CourseInstance'
42          - name: expand
43            in: query
```

Processed with no error

## Course Booking Service

Version 0.0.0

### Paths

/courseInstances

GET /courseInstances

#### Description

Gets all **CourseInstance** objects for all courses. Optional query param of **size** determines size of returned array. Typically you would use GET **/course/{courseId}/courseInstances** instead to get the list of instances for a particular course

#### Parameters

Name	Located in	Description	Required	Schema
size	query	Size of array	No	⇔ <b>number</b> (double)
expand	query	expand objects or not	No	⇔ <b>boolean</b>

#### Responses

Code	Description	Schema
200	Successful response	⇔ <b>ArrayOfCourseInstances</b> [ <b>CourseInstance</b> { } ]

Try this operation

POST /courseInstances




# Editor features

- YAML based
- Allows you to dynamically design APIs / RESTful services
- Creates skeleton projects in many languages
  - including Node and JAX-RS
- Creates clients in many languages
  - Too many to list
- Import/export JSON



# Swagger UI

 **swagger**

http://petstore.swagger.io/v2/swagger.json

Authorize

Explore

## Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [irc.freenode.net, #swagger](irc://freenode.net/#swagger). For this sample, you can use the api key `special-key` to test the authorization filters.

Find out more about Swagger

<http://swagger.io>  
[Contact the developer](#)  
[Apache 2.0](#)

### pet : Everything about your Pets

Show/Hide | List Operations | Expand Operations

POST /pet Add a new pet to the store

#### Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	(required) <div></div> <div>Parameter content type: application/json</div>	Pet object that needs to be added to the store	body	<div>Model</div> <div>Example Value</div> <pre>{   "id": 0,   "category": {     "id": 0,     "name": "string"   },   "name": "doggie",   "photoUrls": [     "string"   ],   "tags": [     { </pre>



# SwaggerHub

The screenshot shows the SwaggerHub web application interface. The browser address bar displays `https://app.swaggerhub.com/home?loggedInWithGitHub=true`. The page header includes the SwaggerHub logo and the user's name 'pzfreo'. The main content area is titled 'MY hub' and features a search bar and a list of APIs. The APIs are sorted by 'Recently updated'.

API Name	Visibility	Description	Tags
test	PUBLIC   UNPUBLISHED	This is a simple API	API
hello	PUBLIC   UNPUBLISHED	hello - [1.0.0]	API, OAS3
ecommerce	PRIVATE   UNPUBLISHED	The API is organized around REST, and is designed to have predictable, re... Endpoint All requests to the API must be made over HTTPS to the following endpoint	

SHOWING 1-3 OF 3





# Questions?



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>