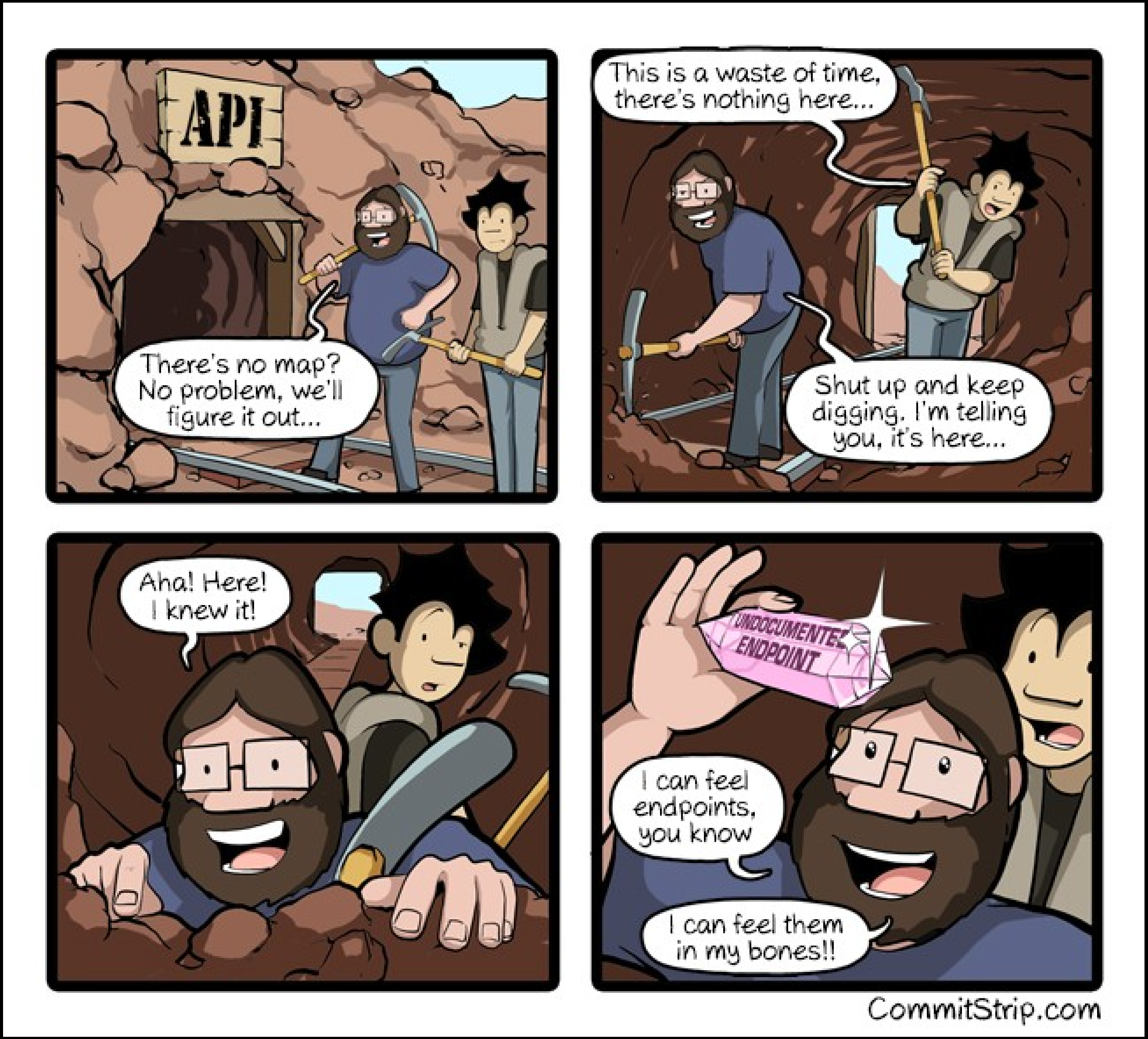
REST Services Description

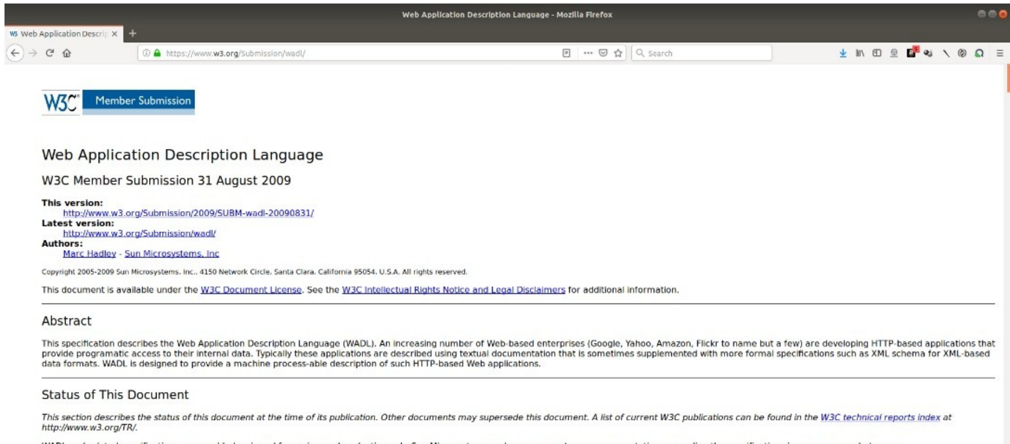

API
There's no map? No problem, we'll figure it out...
This is a waste of time, there's nothing here...
Shut up and keep digging. I'm telling you, it's here...
Aha! Here! I knew it!
UNDOCUMENTED ENDPOINT
I can feel endpoints, you know
I can feel them in my bones!!
CommitStrip.com

2010s: Numerous Competing Proposals

- Formats to describe a service:
  - 2006: WADL
  - 2011: Swagger
  - 2013: API Blueprint, RAML, JSON:API
  - …
- And tools:
  - to assist in writing, templates, syntax checking
  - to translate from one format to another
  - to generate code (client and/or server)

WADL

- *Web Application Description Language*
- inspired by WSDL (description of SOAP services)
- XML description of a set of resources
- submitted by Sun to the W3C in 2009 but will not be standardized (too verbose? stopped by Oracle's takeover?).
- WADL-to-code and code-to-WADL tools


Web Application Description Language - Mozilla Firefox
Web Application Descrip
https://www.w3.org/Submission/wadl/
Search
W3C Member Submission

Web Application Description Language

W3C Member Submission 31 August 2009

This version:
http://www.w3.org/Submission/2009/SUBM-wadl-20090831/
Latest version:
http://www.w3.org/Submission/wadl/
Authors:
Marc Hadley - Sun Microsystems, Inc

Copyright 2005-2009 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

This document is available under the W3C Document License. See the W3C Intellectual Rights Notice and Legal Disclaimers for additional information.

Abstract

This specification describes the Web Application Description Language (WADL). An increasing number of Web-based enterprises (Google, Yahoo, Amazon, Flickr to name but a few) are developing HTTP-based applications that provide programatic access to their internal data. Typically these applications are described using textual documentation that is sometimes supplemented with more formal specifications such as XML schema for XML-based data formats. WADL is designed to provide a machine process-able description of such HTTP-based Web applications.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications can be found in the W3C technical reports index at http://www.w3.org/TR/.

OpenAPI

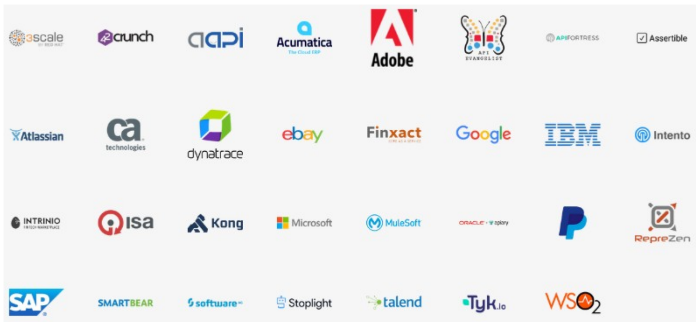
2010 SWAGGER SMARTBEAR

description JSON → interactive documentation
+ generation of SDK for clients

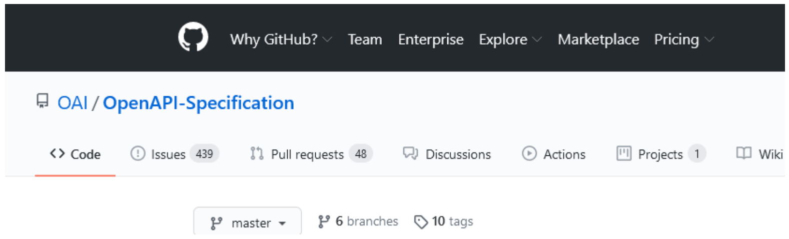
⇨ more and more code-to-JSON
⇨ Formalization of the JSON schema

Swagger focuses on tools

2015 OPENAPI INITIATIVE



2016: renamed to OpenAPI, on GitHub


Why GitHub? Team Enterprise Explore Marketplace Pricing

OAI / **OpenAPI-Specification**

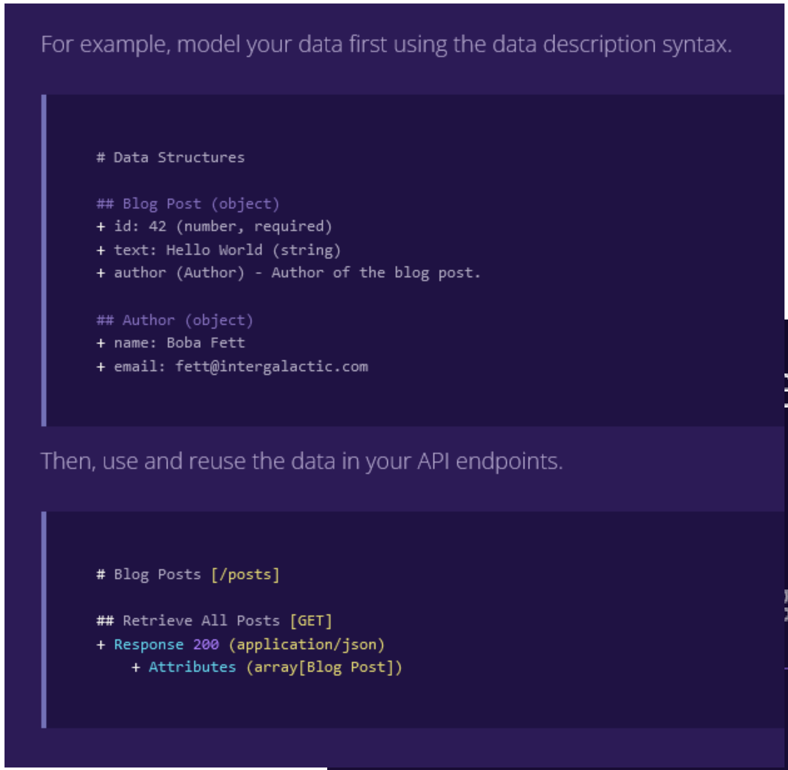
Code | Issues 439 | Pull requests 48 | Discussions | Actions | Projects 1 | Wiki

master | 6 branches | 10 tags

current version: 3.1.0

API Blueprint

- Markdown description
- developed in 2013 by Apiary (acquired by Oracle)

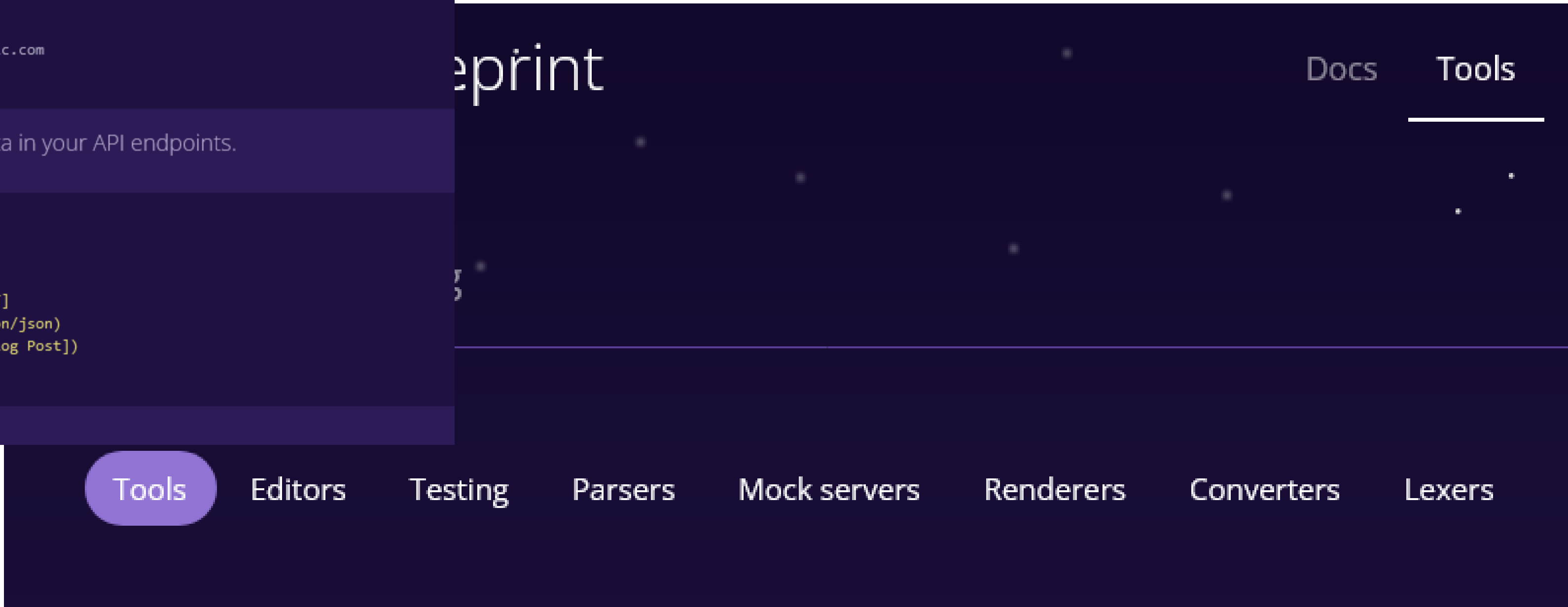


For example, model your data first using the data description syntax.

```
# Data Structures

## Blog Post (object)
+ id: 42 (number, required)
+ text: Hello World (string)
+ author (Author) - Author of the blog post.

## Author (object)
+ name: Boba Fett
+ email: fett@intergalactic.com
```

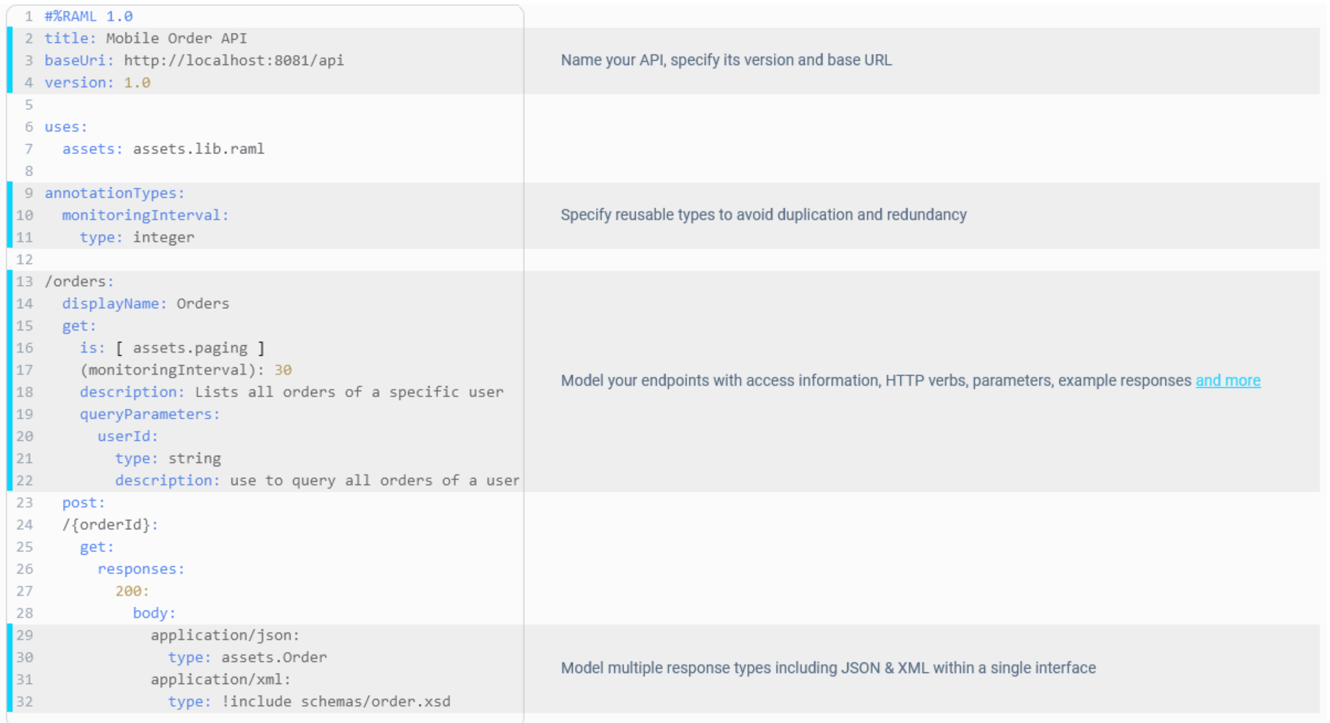
Then, use and reuse the data in your API endpoints.

```
# Blog Posts [/posts]

## Retrieve All Posts [GET]
+ Response 200 (application/json)
    + Attributes (array[Blog Post])
```



RAML

- *RESTful API Modeling Language*
- description in YAML
- developed in 2013 by MuleSoft (acquired by Salesforce)



```yaml
#%RAML 1.0
title: Mobile Order API
baseUri: http://localhost:8081/api
version: 1.0

uses:
  assets: assets.lib.raml

annotationTypes:
  monitoringInterval:
    type: integer

/orders:
  displayName: Orders
  get:
    is: [ assets.paging ]
    (monitoringInterval): 30
    description: Lists all orders of a specific user
    queryParameters:
      userId:
        type: string
        description: use to query all orders of a user
  post:
  /{orderId}:
    get:
      responses:
        200:
          body:
            application/json:
              type: assets.Order
            application/xml:
              type: !include schemas/order.xsd
```

Name your API, specify its version and base URL

Specify reusable types to avoid duplication and redundancy

Model your endpoints with access information, HTTP verbs, parameters, example responses and more

Model multiple response types including JSON & XML within a single interface

- design
- build
- test
- document
- share & support

JSON:API (jsonapi.org)

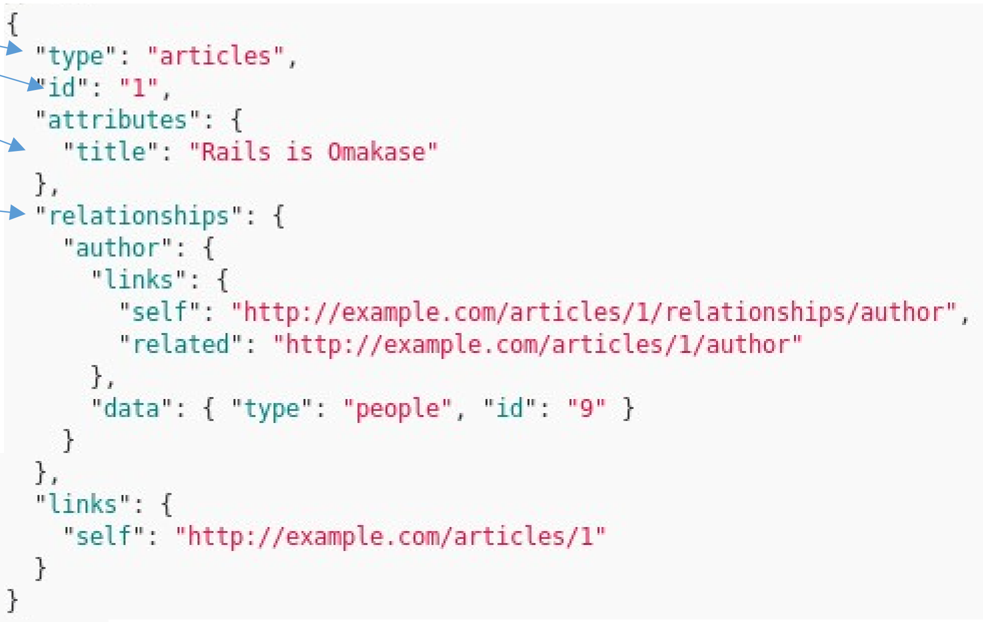
- Drafted in 2013 at tilde.io, latest version 09/2022
- Slightly different goal: to structure APIs: helps API authors by offering well-thought-out patterns for supporting common features:
  - Sorting, pagination, limit the number of returned resources
  - HTTP Caching
  - Compound documents (send related resources alongside the requested primary resources)
  - Sparse fieldsets (only request data from specific fields (GraphQL-like))

JSON API: example

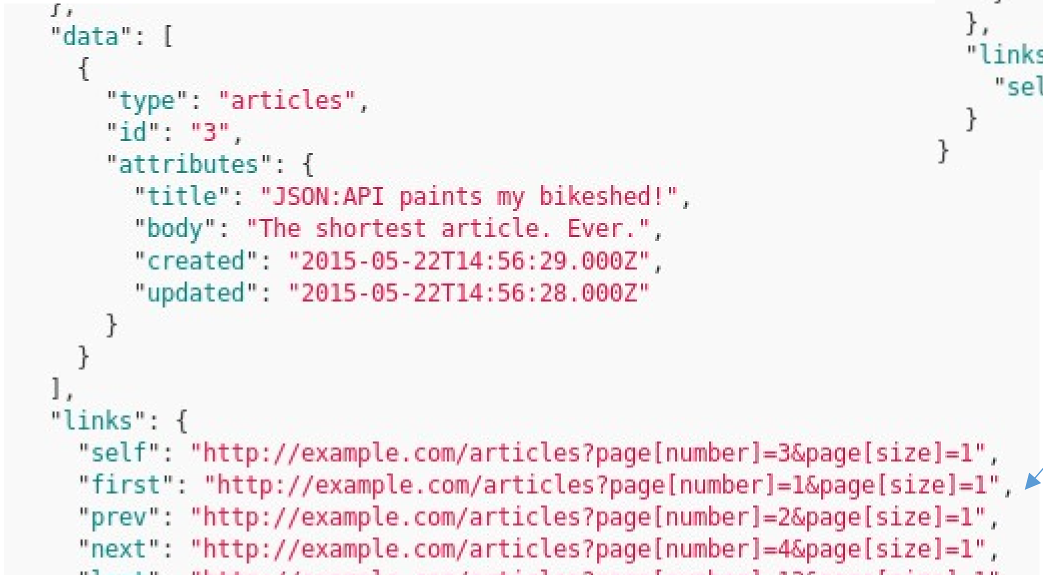
mandatory, that pair uniquely identifies the resource

attributes (state of the resource,
independent from other resources)

relationships (links to other resources)



```
{
  "type": "articles",
  "id": "1",
  "attributes": {
    "title": "Rails is Omakase"
  },
  "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"
  }
}
```

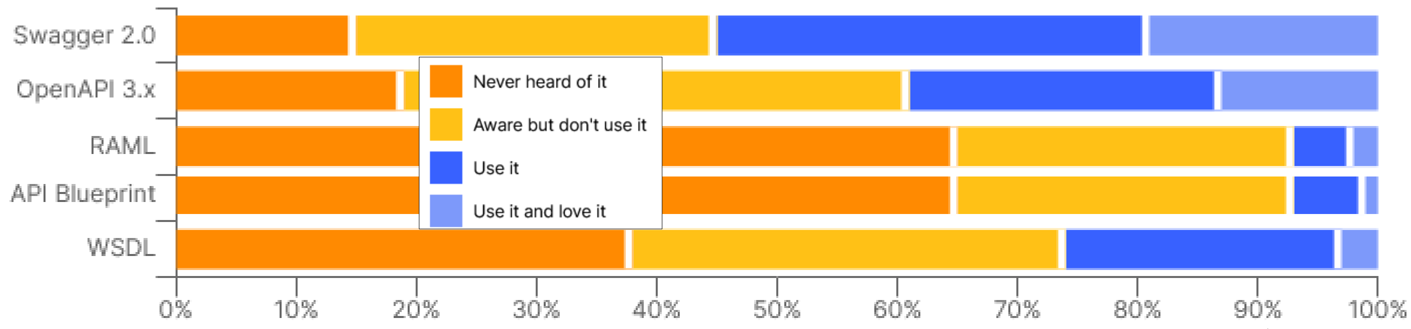


```
},
"data": [
  {
    "type": "articles",
    "id": "3",
    "attributes": {
      "title": "JSON:API paints my bikeshed!",
      "body": "The shortest article. Ever.",
      "created": "2015-05-22T14:56:29.000Z",
      "updated": "2015-05-22T14:56:28.000Z"
    }
  }
],
"links": {
  "self": "http://example.com/articles?page[number]=3&page[size]=1",
  "first": "http://example.com/articles?page[number]=1&page[size]=1",
  "prev": "http://example.com/articles?page[number]=2&page[size]=1",
  "next": "http://example.com/articles?page[number]=4&page[size]=1",
```

HATEOS links

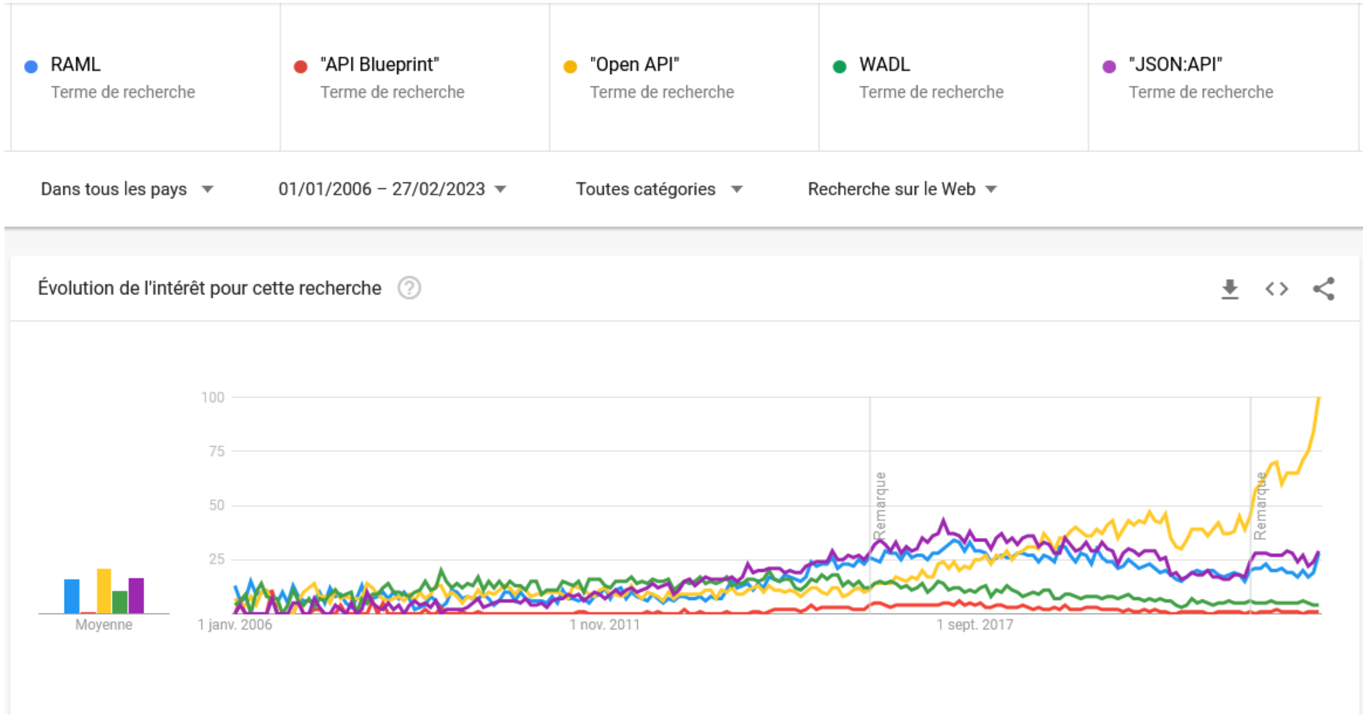
And the winner is…

	WADL	RAML	API blueprint	JSON:API	Open API
Questions tagged on StackOverflow	218	365	252	679	3869
Étoiles GitHub		3800	8500	7100	25900



Postman State of the API Report (summer 2022)

Open API Specification



● RAML Terme de recherche

● "API Blueprint" Terme de recherche

● "Open API" Terme de recherche

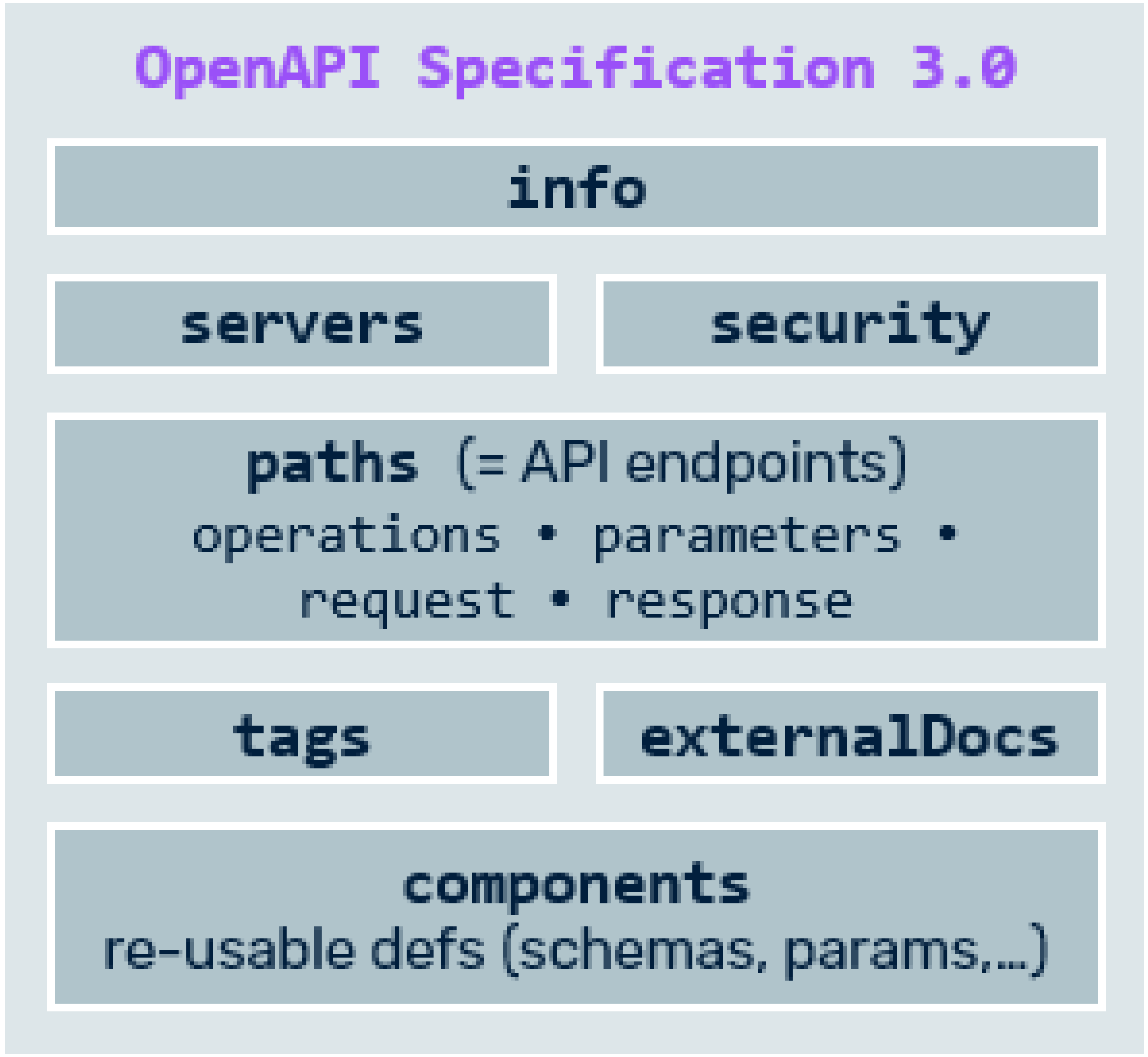
● WADL Terme de recherche

● "JSON:API" Terme de recherche

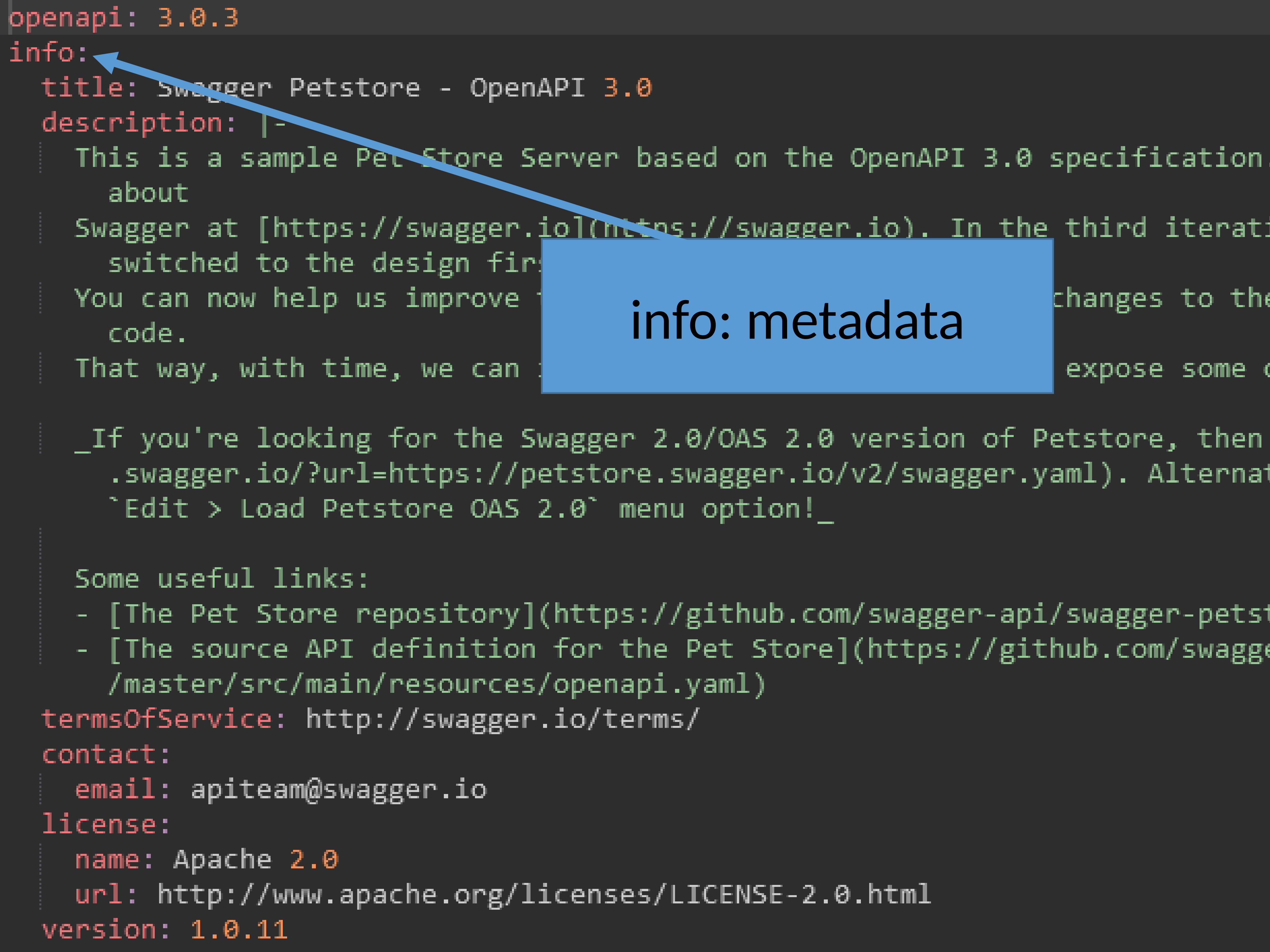
Dans tous les pays ▾ 01/01/2006 – 27/02/2023 ▾ Toutes catégories ▾ Recherche sur le Web ▾

Évolution de l'intérêt pour cette recherche

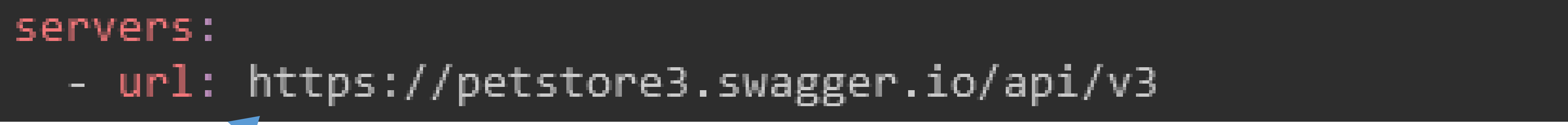
OpenAPI Structure


OpenAPI Specification 3.0
info
servers
security
paths (= API endpoints)
operations • parameters •
request • response
tags
externalDocs
components
re-usable defs (schemas, params,...)

description in JSON or YAML



```yaml
openapi: 3.0.3
info:
  title: Swagger Petstore - OpenAPI 3.0
  description: |-
    This is a sample Pet Store Server based on the OpenAPI 3.0 specification.
      about
    Swagger at [https://swagger.io](https://swagger.io). In the third iterati
      switched to the design fir
    You can now help us improve t                              changes to the
      code.
    That way, with time, we can i                              expose some o

    _If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then
      .swagger.io/?url=https://petstore.swagger.io/v2/swagger.yaml). Alternat
      `Edit > Load Petstore OAS 2.0` menu option!_


    Some useful links:
    - [The Pet Store repository](https://github.com/swagger-api/swagger-petst
    - [The source API definition for the Pet Store](https://github.com/swagge
      /master/src/main/resources/openapi.yaml)
  termsOfService: http://swagger.io/terms/
  contact:
    email: apiteam@swagger.io
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.11
```

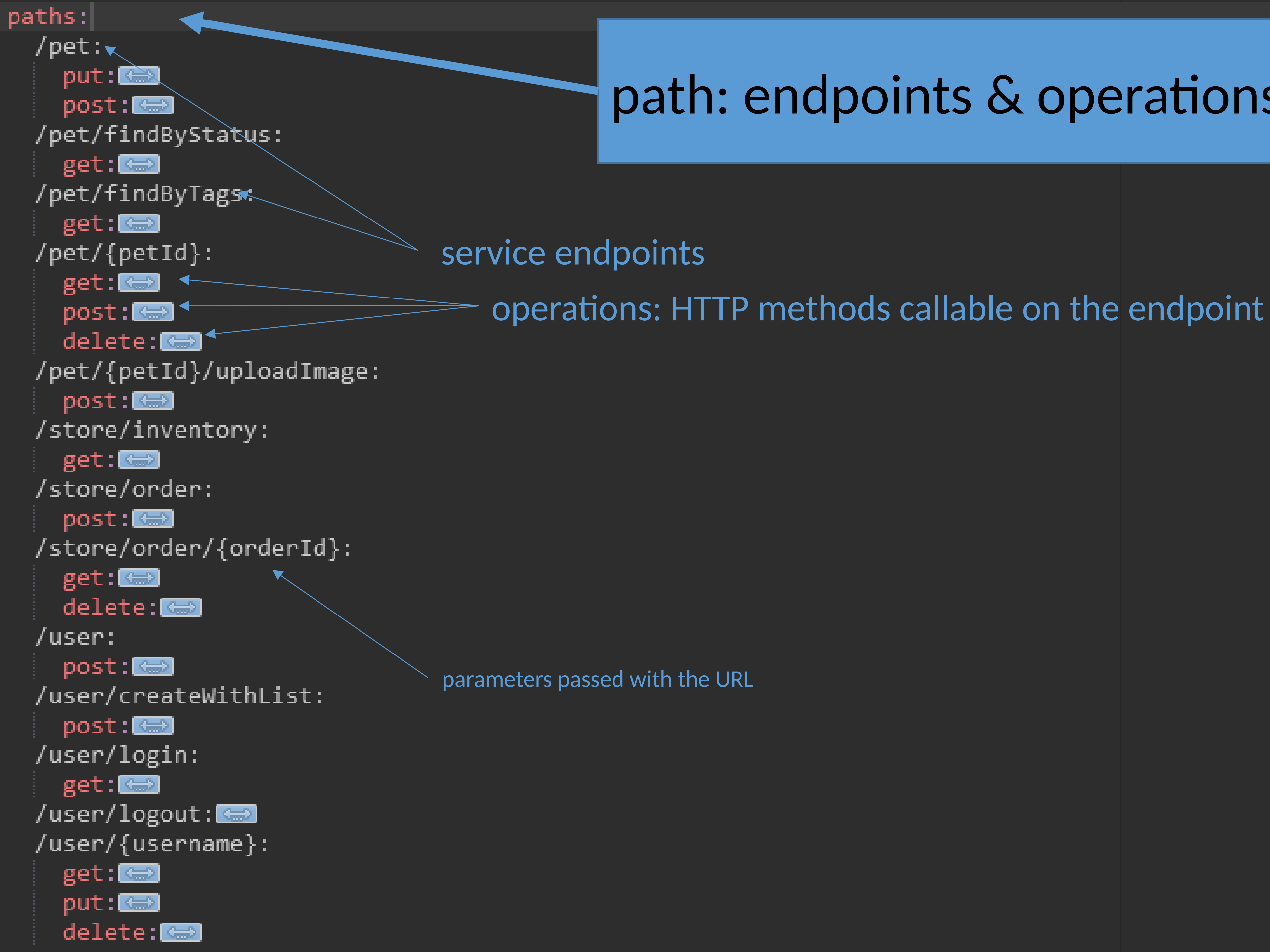


```yaml
servers:
  - url: https://petstore3.swagger.io/api/v3
```

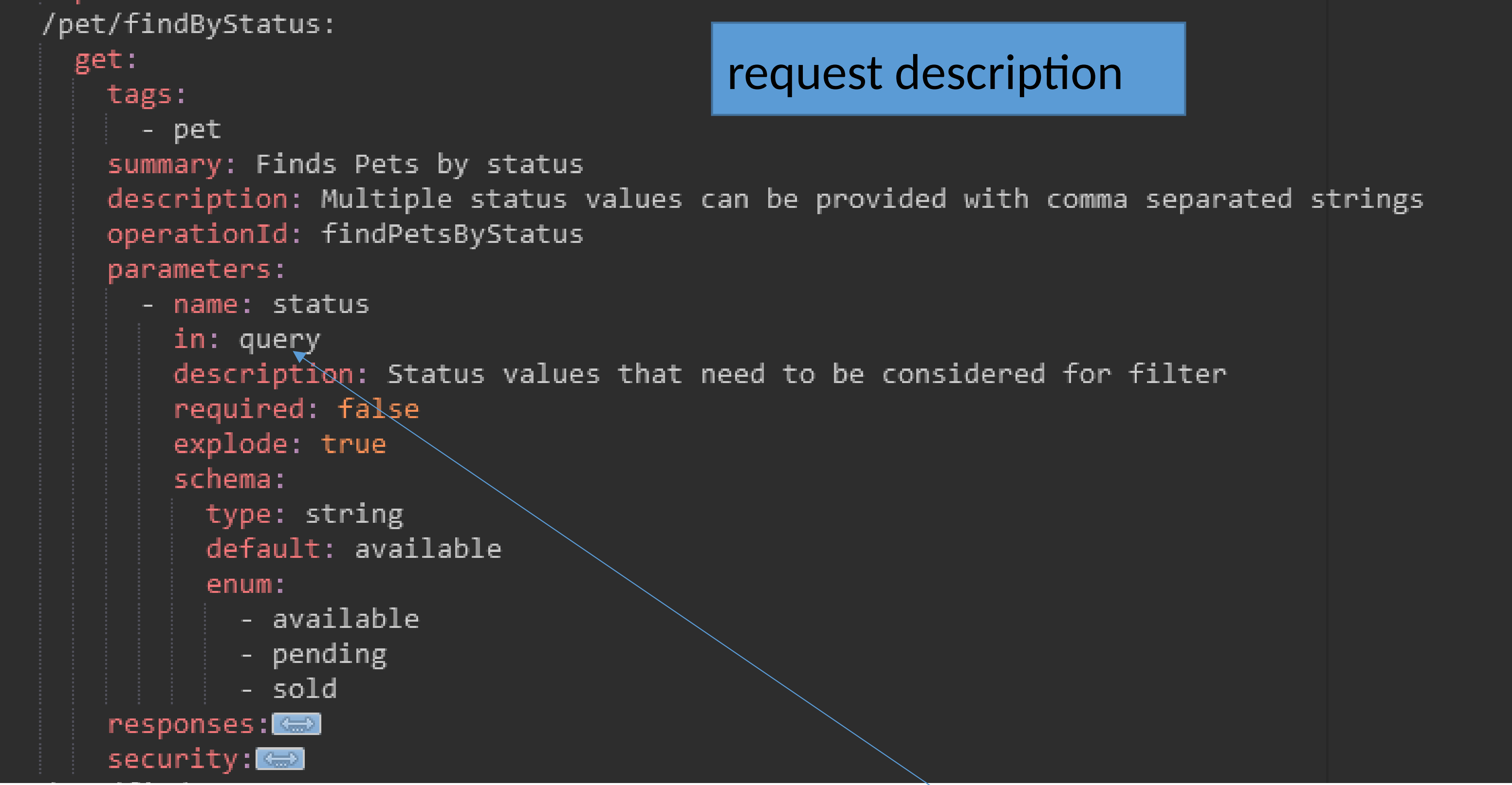
server: base URL

can define multiple servers:

```yaml
servers:
    - url: http://api.example.com/v1
      description: main (production) server
    - url: http://staging-api.example.com
      description: internal staging server for testing
    - url: https://staging-api.example.com
      description: internal staging HTTPS server for testing
```


path: endpoints & operation
service endpoints
operations: HTTP methods callable on the endpoint
parameters passed with the URL

```yaml
paths:
  /pet:
    put:
    post:
  /pet/findByStatus:
    get:
  /pet/findByTags:
    get:
  /pet/{petId}:
    get:
    post:
    delete:
  /pet/{petId}/uploadImage:
    post:
  /store/inventory:
    get:
  /store/order:
    post:
  /store/order/{orderId}:
    get:
    delete:
  /user:
    post:
  /user/createWithList:
    post:
  /user/login:
    get:
  /user/logout:
  /user/{username}:
    get:
    put:
    delete:
```



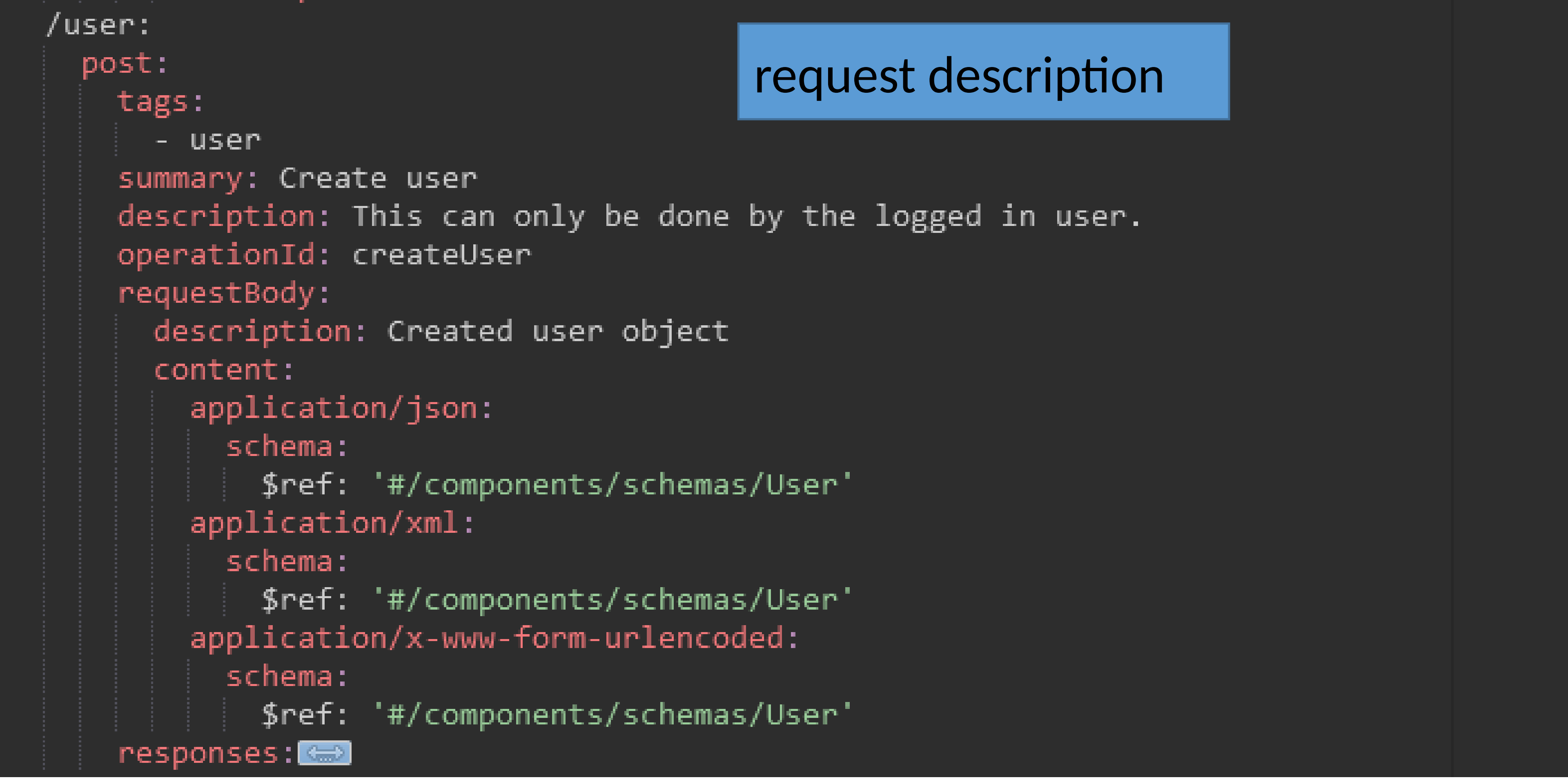
```yaml
/pet/findByStatus:
  get:
    tags:
      - pet
    summary: Finds Pets by status
    description: Multiple status values can be provided with comma separated strings
    operationId: findPetsByStatus
    parameters:
      - name: status
        in: query
        description: Status values that need to be considered for filter
        required: false
        explode: true
        schema:
          type: string
          default: available
          enum:
            - available
            - pending
            - sold
    responses:
    security:
```

request description

How is the parameter passed:

- “path”: /users/{userId}
- “query”: /users?role=admin
- “header”: X-CustomHeader: Value
- “cookie”: Cookie: debug=0

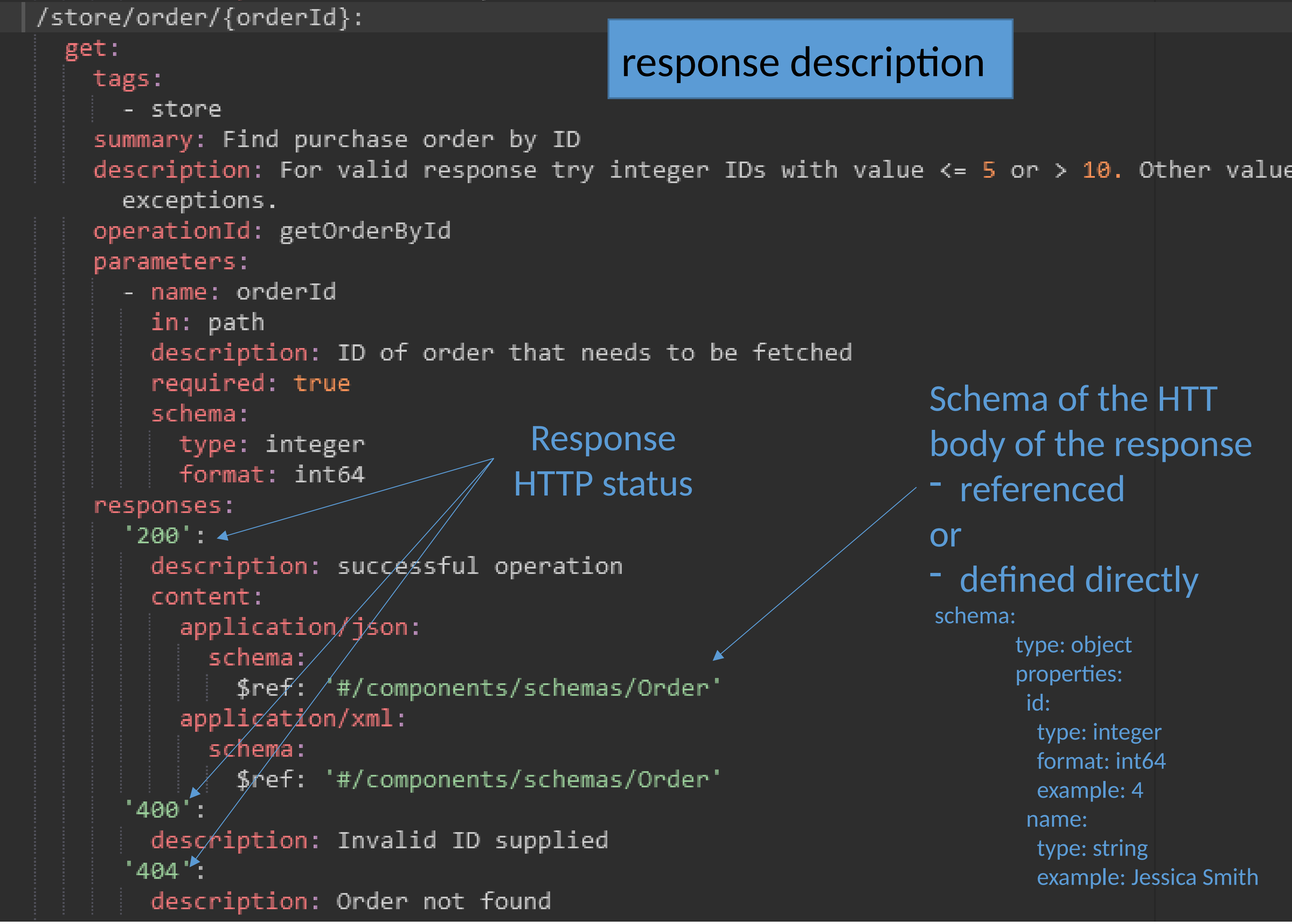
GET https://petstore3.swagger.io/api/v3/pet/findByStatus?status=sold



```
/user:
  post:
    tags:
      - user
    summary: Create user
    description: This can only be done by the logged in user.
    operationId: createUser
    requestBody:
      description: Created user object
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/User'
        application/xml:
          schema:
            $ref: '#/components/schemas/User'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/User'
    responses:
```

request description

if the operation includes an HTTP body, description of its contents



```yaml
/store/order/{orderId}:
  get:
    tags:
      - store
    summary: Find purchase order by ID
    description: For valid response try integer IDs with value <= 5 or > 10. Other value
      exceptions.
    operationId: getOrderById
    parameters:
      - name: orderId
        in: path
        description: ID of order that needs to be fetched
        required: true
        schema:
          type: integer
          format: int64
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Order'
          application/xml:
            schema:
              $ref: '#/components/schemas/Order'
      '400':
        description: Invalid ID supplied
      '404':
        description: Order not found
```

response description

Response
HTTP status

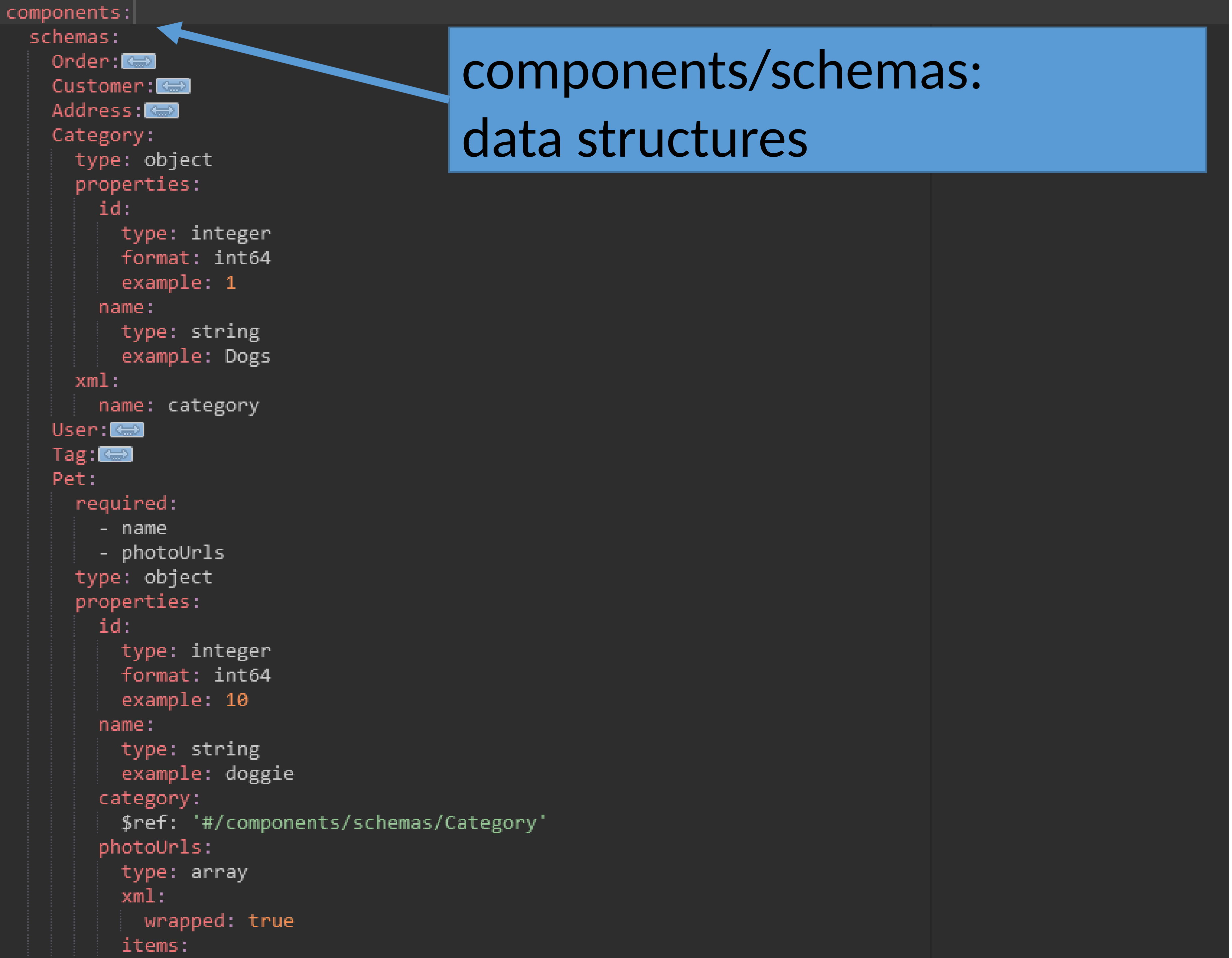
Schema of the HTT
body of the response
- referenced

or

- defined directly

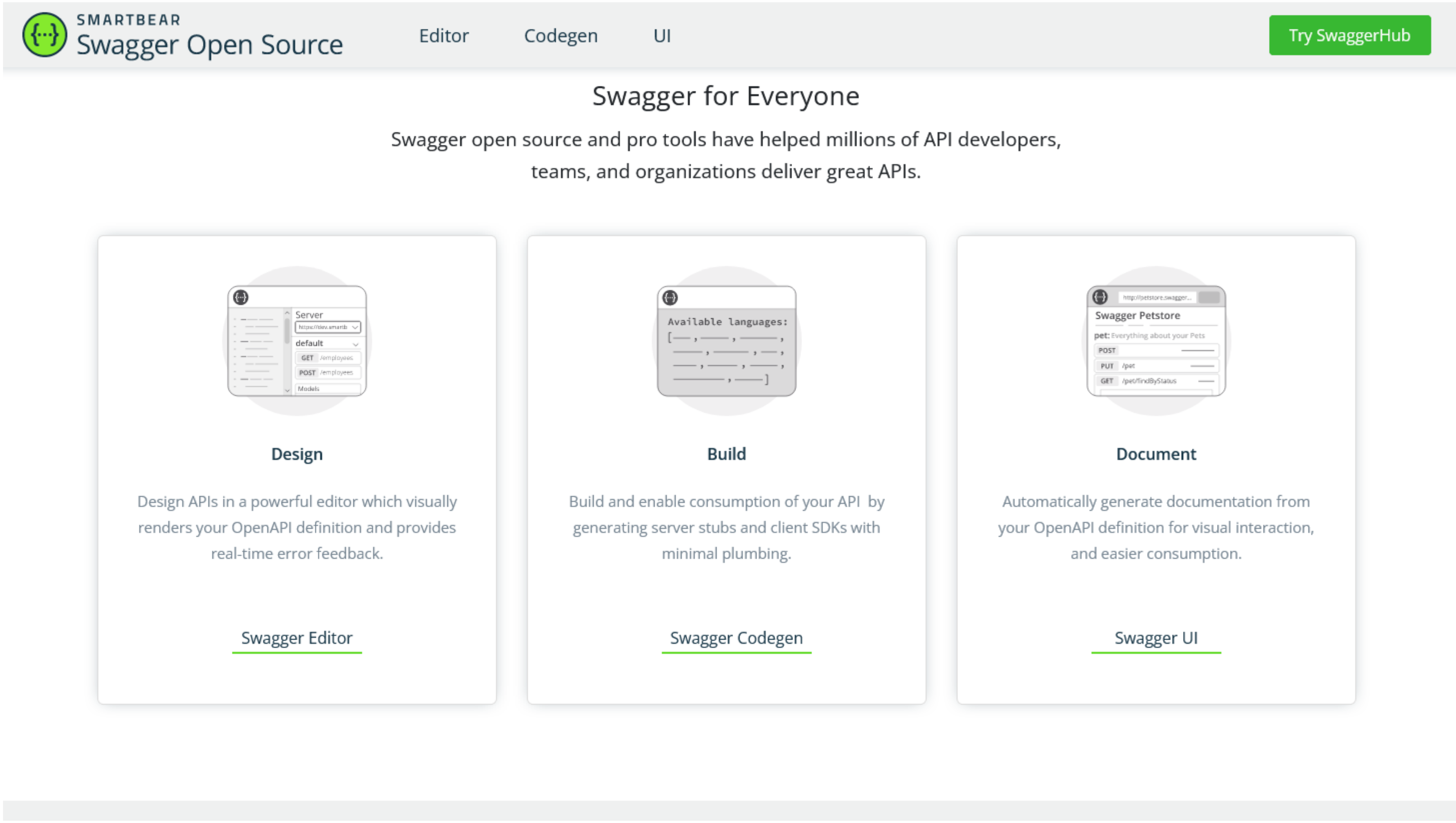
```
schema:
      type: object
      properties:
        id:
          type: integer
          format: int64
          example: 4
        name:
          type: string
          example: Jessica Smith
```

components/schemas:
data structures



```yaml
components:
  schemas:
    Order:⟺
    Customer:⟺
    Address:⟺
    Category:
      type: object
      properties:
        id:
          type: integer
          format: int64
          example: 1
        name:
          type: string
          example: Dogs
      xml:
        name: category
    User:⟺
    Tag:⟺
    Pet:
      required:
        - name
        - photoUrls
      type: object
      properties:
        id:
          type: integer
          format: int64
          example: 10
        name:
          type: string
          example: doggie
        category:
          $ref: '#/components/schemas/Category'
        photoUrls:
          type: array
          xml:
            wrapped: true
          items:
```

Swagger tools



Swagger for Everyone

Swagger open source and pro tools have helped millions of API developers, teams, and organizations deliver great APIs.

Design

Design APIs in a powerful editor which visually renders your OpenAPI definition and provides real-time error feedback.

Swagger Editor

Build

Build and enable consumption of your API by generating server stubs and client SDKs with minimal plumbing.

Swagger Codegen

Document

Automatically generate documentation from your OpenAPI definition for visual interaction, and easier consumption.

Swagger UI

Open API (specification) ≠ Open API (Public API)

- 2-3 kinds of APIs:
    - Public API (aka Open API)
        - can be used by client developers outside the organization of the service provider
        - Example: Google maps API
    - Private API
        - Only consumed within the organization that developed it
    - (Partner APIs: restricted to business partners)

- Both kinds of APIs can be described using the Open API specification… or something else!

Hands-On Activities

- YAML Basics
  - A human-readable data-serialization language
  - Short reading assignment
- Presentation of OpenAPI
  - A REST services description specification
  - Short reading assignment
- Understand an OpenAPI Document (GeoDataSource)
- Use Swagger Tools to Query an OpenAPI-Described Service (GeoDataSource)
- Use Swagger Editor to Write the OpenAPI Description of a Service (Chuck Norris)
- User Swagger Editor to generate a Java SDK or a Python package from the Chuck Norris service OpenAPI description