MAKE OUR

API Specs

GREAT AGAIN!

WHAT PROBLEMS ARE WE TRYING TO SOLVE?

- datatypes ambiguity
- Single Source of Truth violation
- inability to generate changelog
- absence of modeling tools

WHAT WOULD BE COOL TO HAVE?

- interactive testing tool (like postman)
- clients auto generation
- code ↔ spec bidirectional transformation
- mocks

WHAT ARE THE OPTIONS?

Standards:

- WADL
- RSDL
- API Blueprint
- RAML
- OpenAPI

Tools:

- Slate
- Spring REST Docs

WADL

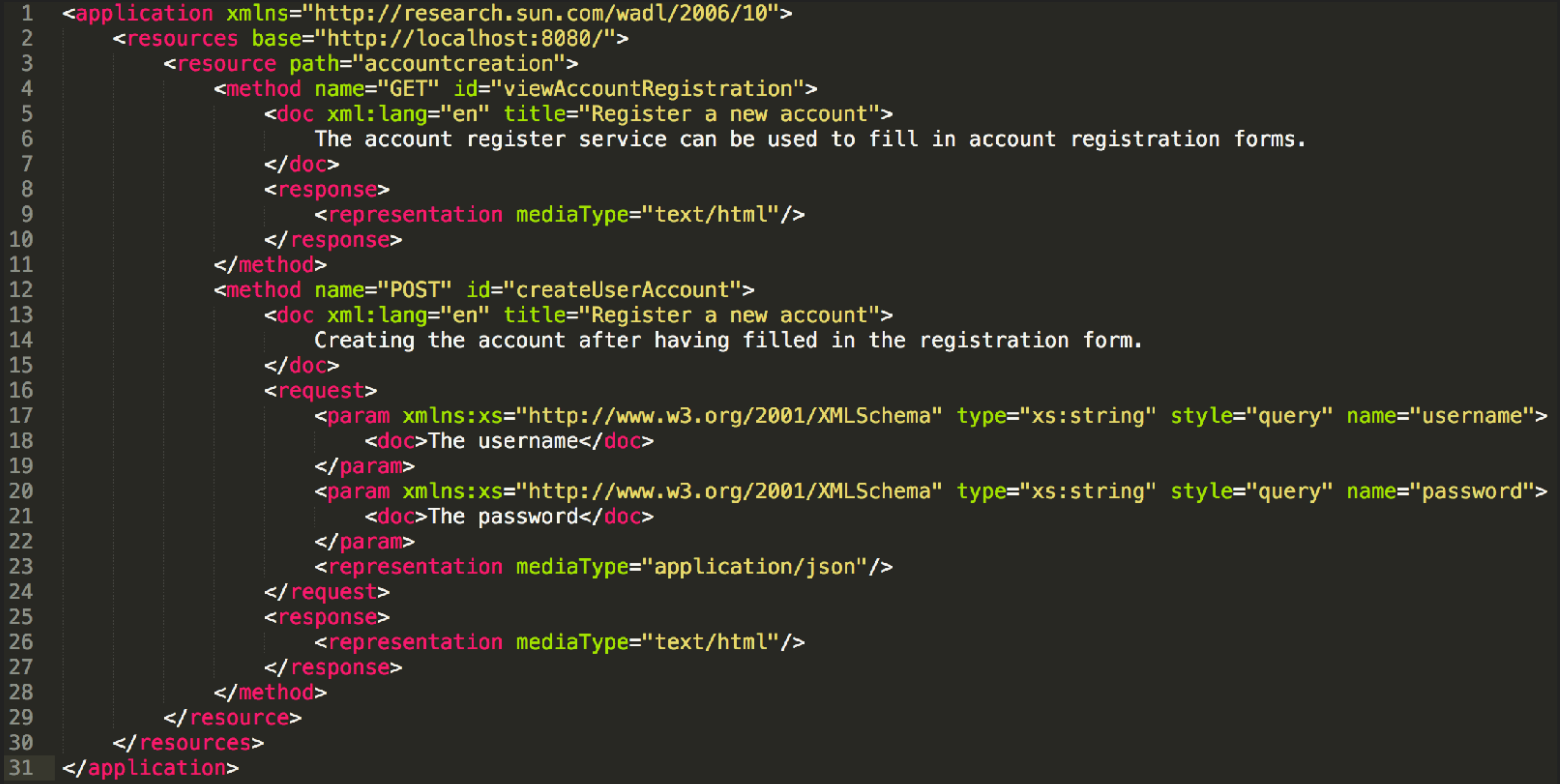
WHAT IS IT?

Web Application Description Language

REST equivalent of SOAP's Web Services Description Language (WSDL)

Submitted as proposal to W3C by Sun on 2009.

HOW DOES IT LOOK LIKE?



```
<application xmlns="http://research.sun.com/wadl/2006/10">
    <resources base="http://localhost:8080/">
        <resource path="accountcreation">
            <method name="GET" id="viewAccountRegistration">
                <doc xml:lang="en" title="Register a new account">
                    The account register service can be used to fill in account registration forms.
                </doc>
                <response>
                    <representation mediaType="text/html"/>
                </response>
            </method>
            <method name="POST" id="createUserAccount">
                <doc xml:lang="en" title="Register a new account">
                    Creating the account after having filled in the registration form.
                </doc>
                <request>
                    <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string" style="query" name="username">
                        <doc>The username</doc>
                    </param>
                    <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string" style="query" name="password">
                        <doc>The password</doc>
                    </param>
                    <representation mediaType="application/json"/>
                </request>
                <response>
                    <representation mediaType="text/html"/>
                </response>
            </method>
        </resource>
    </resources>
</application>
```

HOW TO WRITE / GENERATE?

- generate using JAX-RS implementation (Jersey, RESTEasy), enunciate
- write by hand in any editor :(

HOW CAN IT BE USED?

- SoapUI
- generate server-side stubs from the spec
- generate client (Apache CXF)

WHY IS THIS NOT WHAT WE NEED?

- mostly about a code-first approach - no one loves (to write) XML :)
- not very popular nowadays

RSDL

WHAT IS IT?

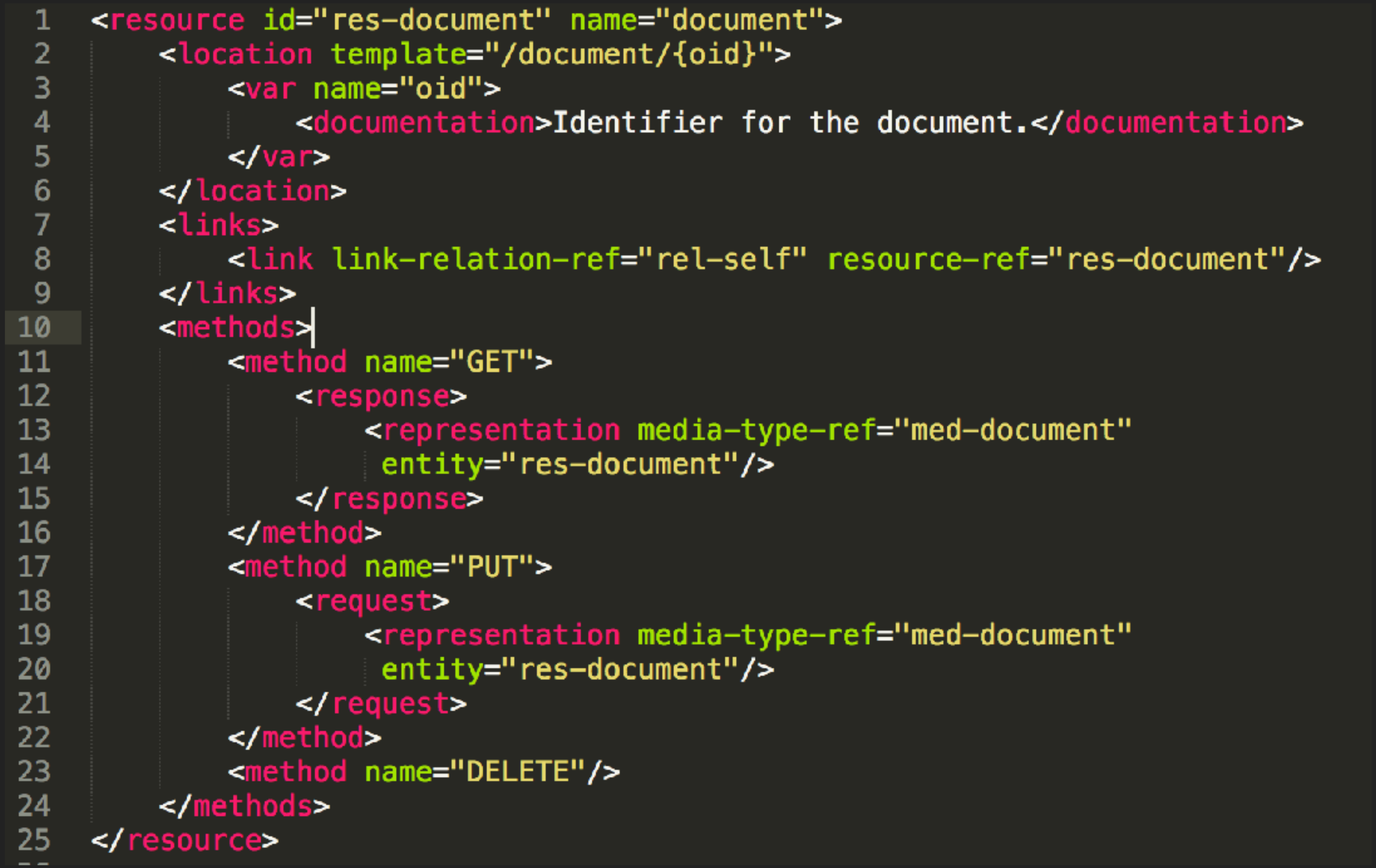
RESTful service description language.

Machine and human-readable XML description of HTTP-based web applications.

Mostly analog to WADL, with small differences.

Created by Michael Pasternak during his work on oVirt RESTful API (RedHat)

HOW DOES IT LOOK LIKE?



```xml
<resource id="res-document" name="document">
    <location template="/document/{oid}">
        <var name="oid">
            <documentation>Identifier for the document.</documentation>
        </var>
    </location>
    <links>
        <link link-relation-ref="rel-self" resource-ref="res-document"/>
    </links>
    <methods>
        <method name="GET">
            <response>
                <representation media-type-ref="med-document"
                 entity="res-document"/>
            </response>
        </method>
        <method name="PUT">
            <request>
                <representation media-type-ref="med-document"
                 entity="res-document"/>
            </request>
        </method>
        <method name="DELETE"/>
    </methods>
</resource>
```

HOW TO WRITE / GENERATE?

- write by hand in any editor :(

WHY IS THIS NOT WHAT WE NEED?

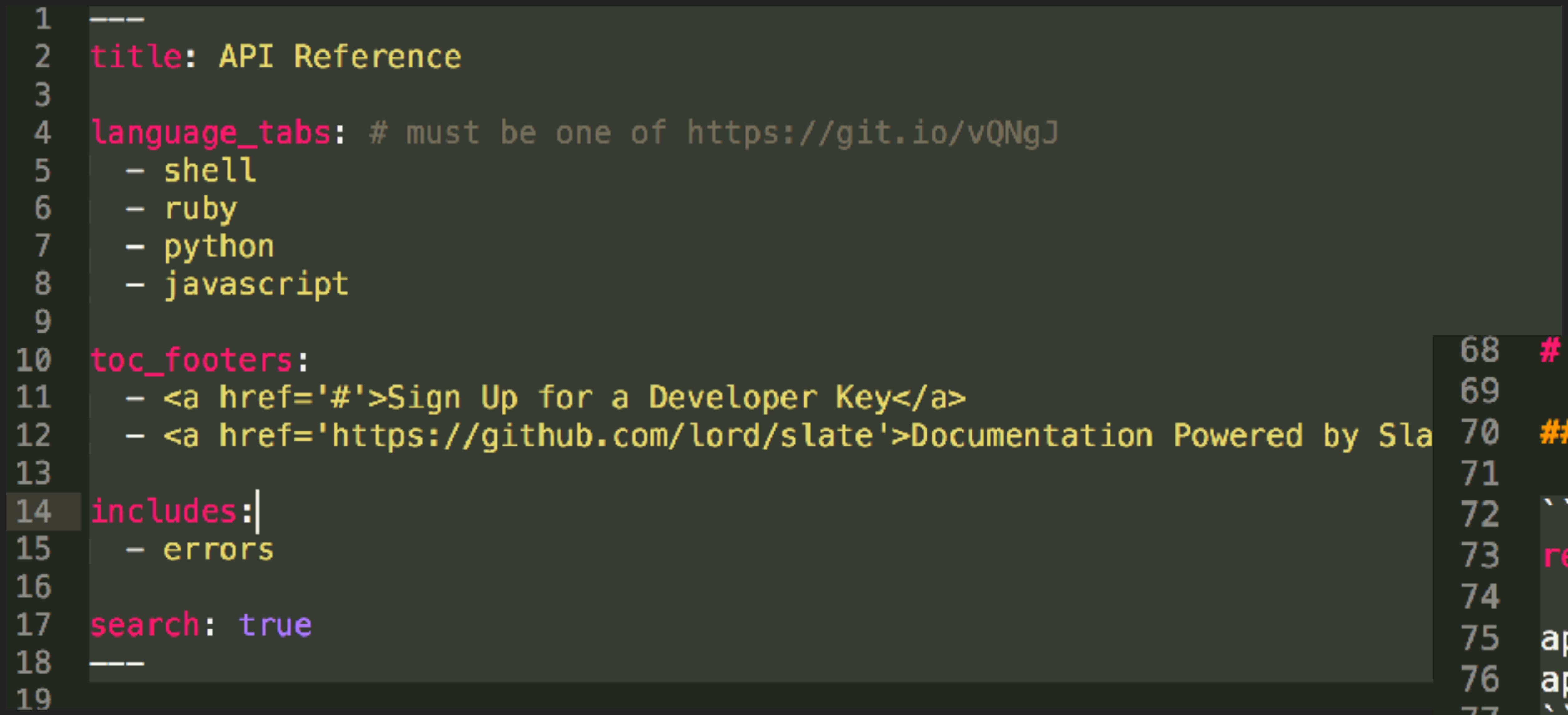
- hard to write / read - no one loves (to write) xml :)
- seems like it was abandoned even by authors

Slate

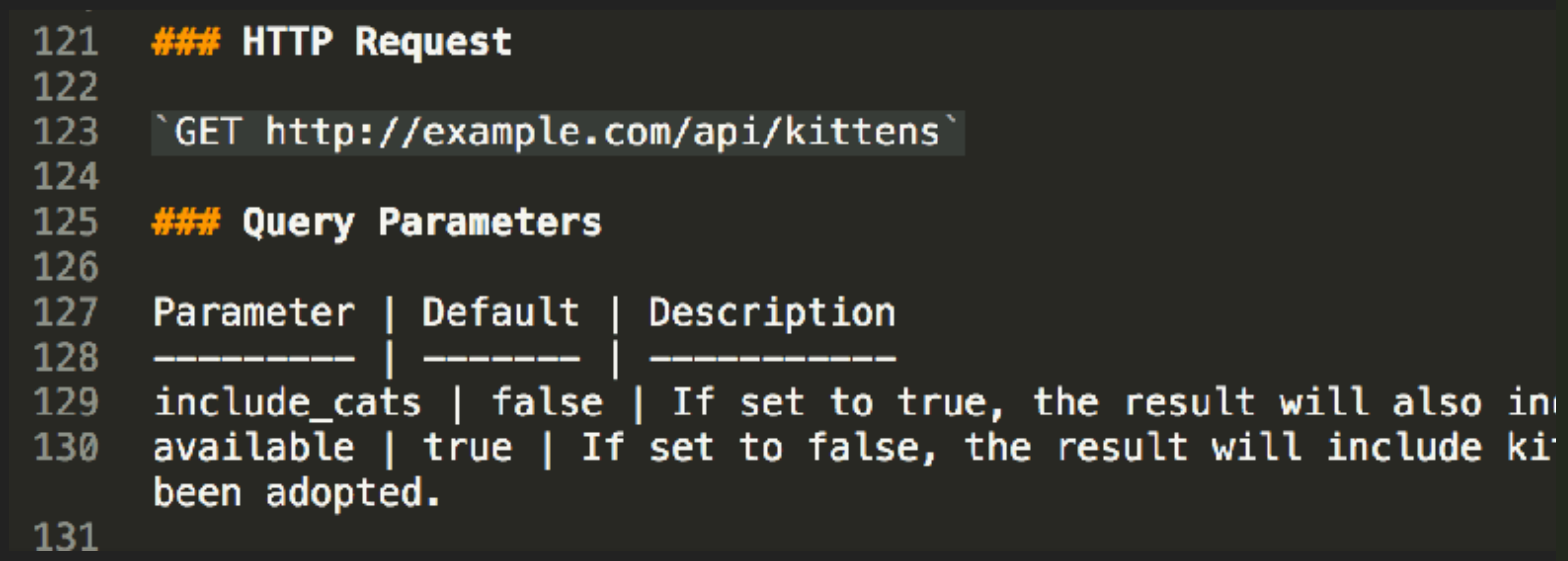
WHAT IS IT?

Markdown-based language for describing APIs + ruby tools for rendering doc.

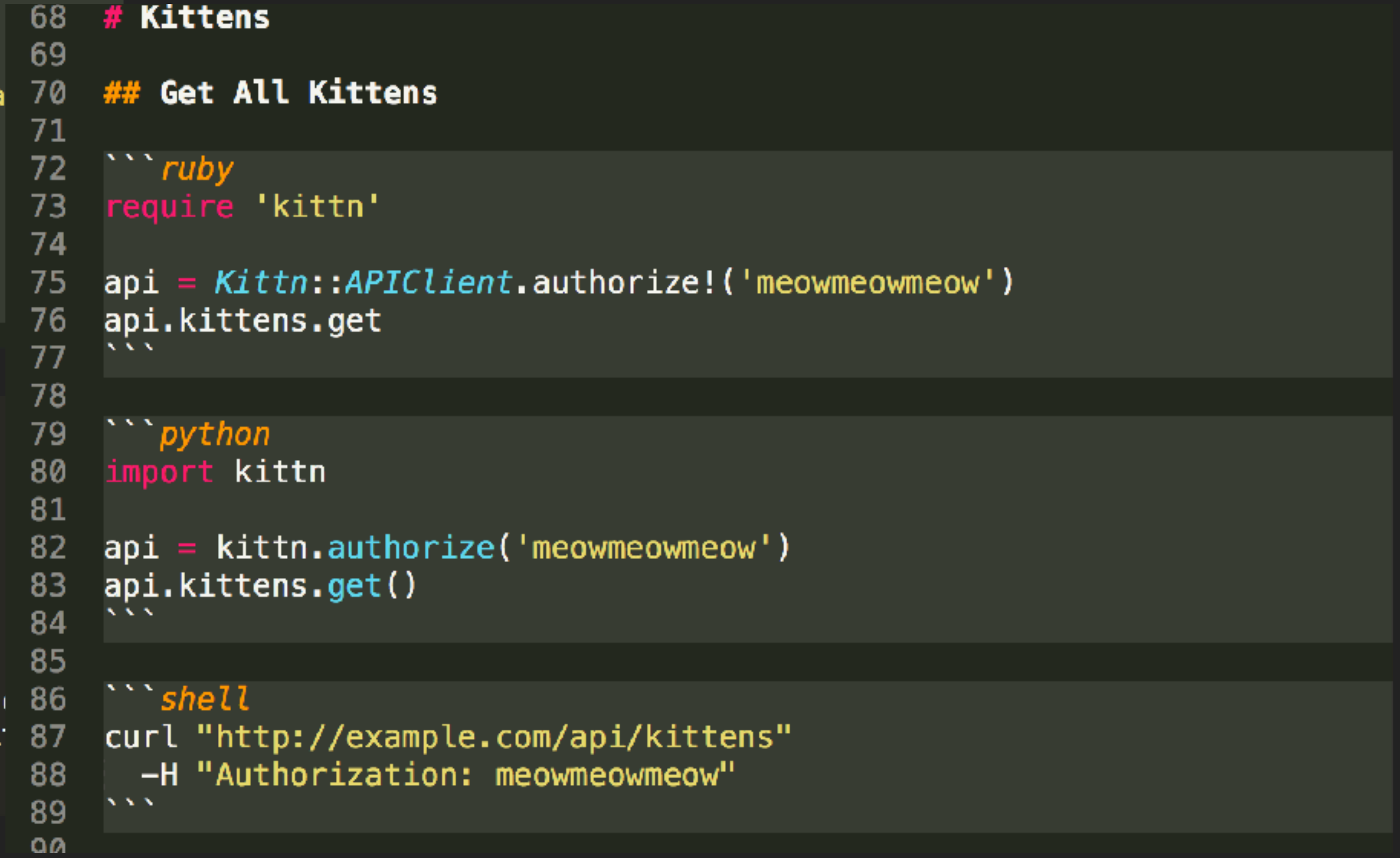
HOW DOES IT LOOK LIKE?



```
---
title: API Reference

language_tabs: # must be one of https://git.io/vQNgJ
  - shell
  - ruby
  - python
  - javascript

toc_footers:
  - <a href='#'>Sign Up for a Developer Key</a>
  - <a href='https://github.com/lord/slate'>Documentation Powered by Sla

includes:
  - errors

search: true
---
```



```
### HTTP Request

`GET http://example.com/api/kittens`

### Query Parameters

Parameter | Default | Description
--------- | ------- | -----------
include_cats | false | If set to true, the result will also in
available | true | If set to false, the result will include ki
been adopted.

```



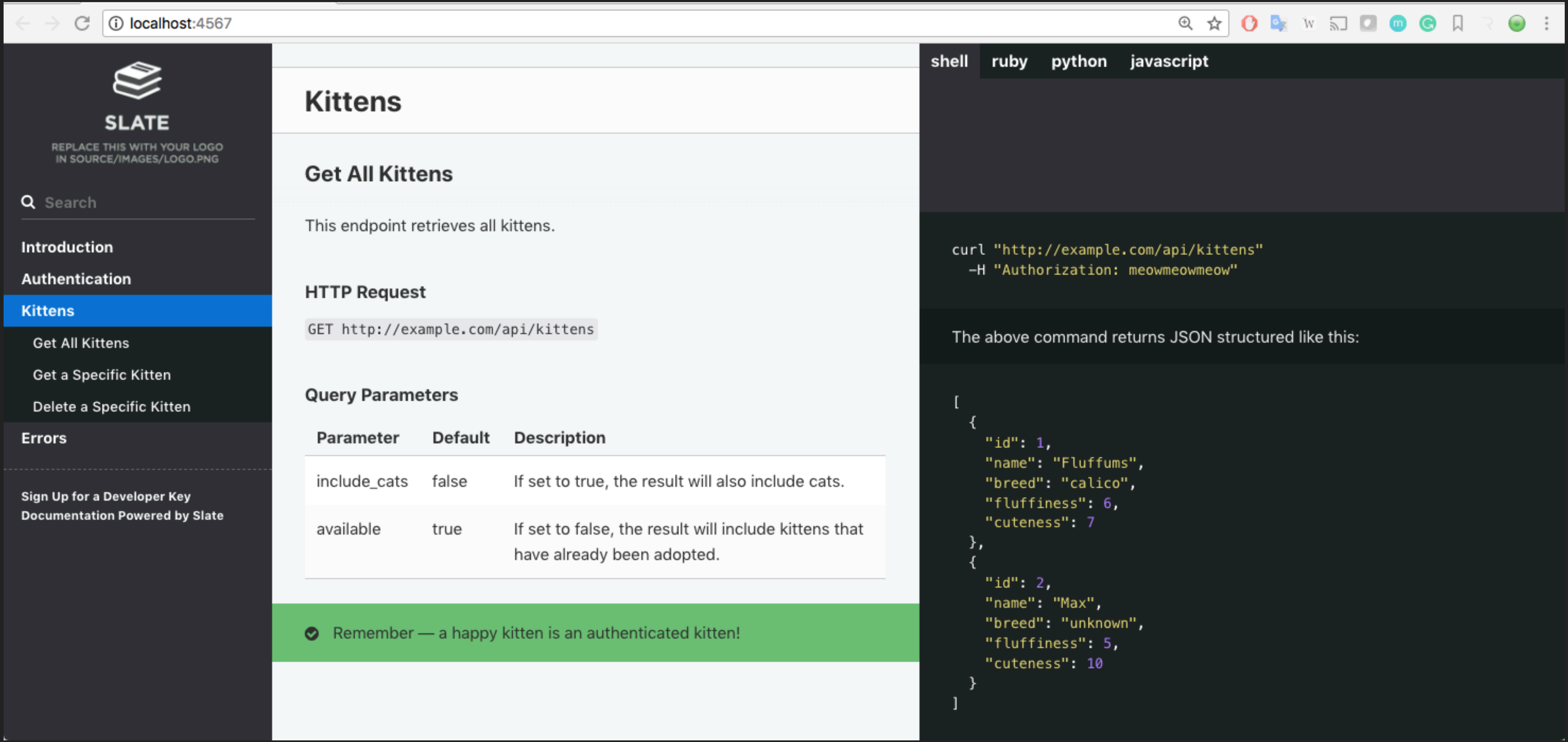
~~~
# Kittens

## Get All Kittens

```ruby
require 'kittn'

api = Kittn::APIClient.authorize!('meowmeowmeow')
api.kittens.get
```

```python
import kittn

api = kittn.authorize('meowmeowmeow')
api.kittens.get()
```

```shell
curl "http://example.com/api/kittens"
  -H "Authorization: meowmeowmeow"
```
~~~

HOW TO WRITE / GENERATE?

- write with any editor with MD highlighting
- no options for generating

HOW CAN IT BE USED?

- render nice doc :)



Kittens

Get All Kittens

This endpoint retrieves all kittens.

HTTP Request

`GET http://example.com/api/kittens`

Query Parameters

Parameter	Default	Description
include_cats	false	If set to true, the result will also include cats.
available	true	If set to false, the result will include kittens that have already been adopted.

Remember — a happy kitten is an authenticated kitten!

```
curl "http://example.com/api/kittens"
  -H "Authorization: meowmeowmeow"
```

The above command returns JSON structured like this:

```
[
  {
    "id": 1,
    "name": "Fluffums",
    "breed": "calico",
    "fluffiness": 6,
    "cuteness": 7
  },
  {
    "id": 2,
    "name": "Max",
    "breed": "unknown",
    "fluffiness": 5,
    "cuteness": 10
  }
]
```

WHY IS THIS NOT WHAT WE NEED?

- hard to maintain (no reusability, special editors)
- could not be generated/used for code generation

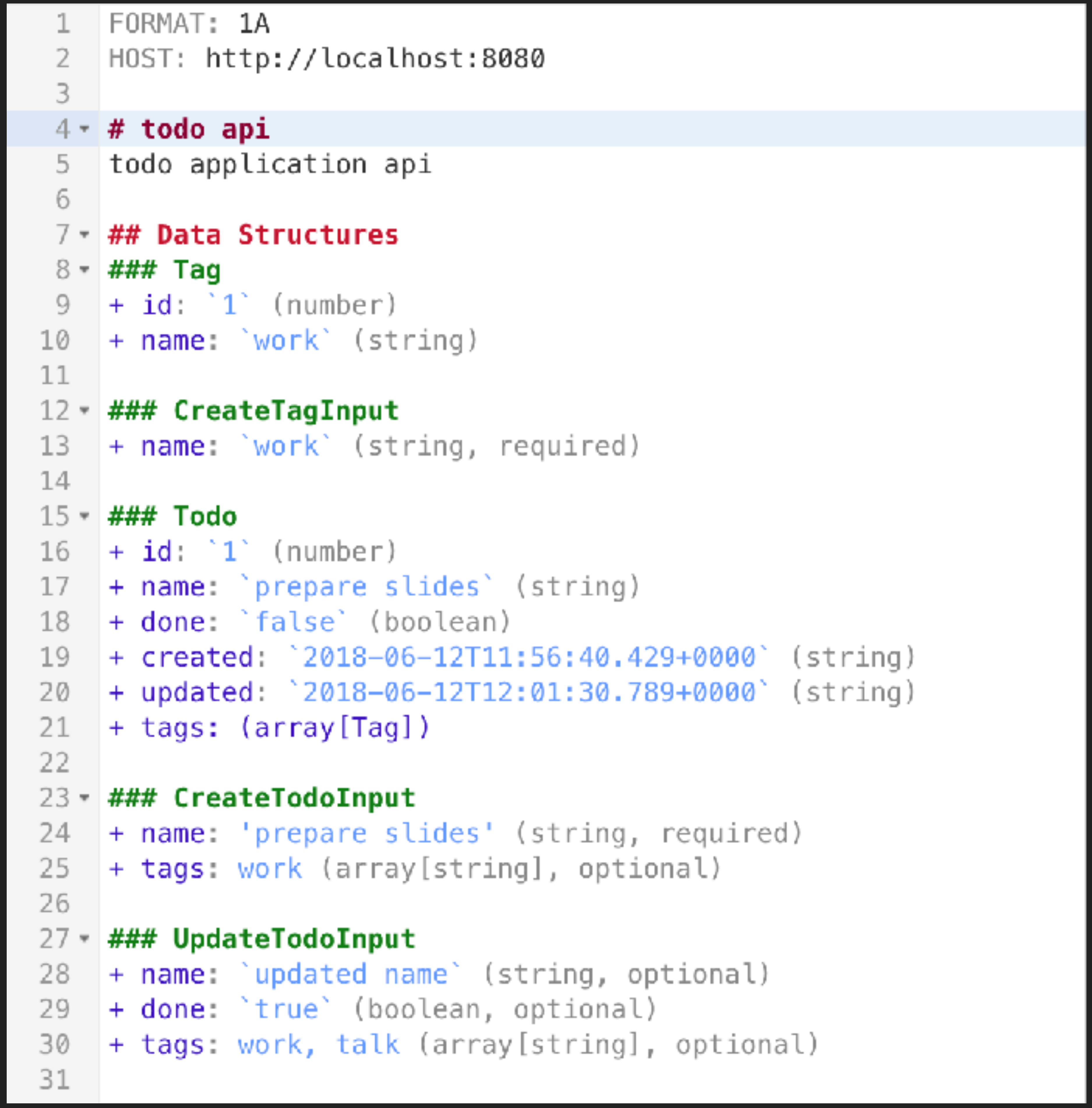
API Blueprint

WHAT IS IT?

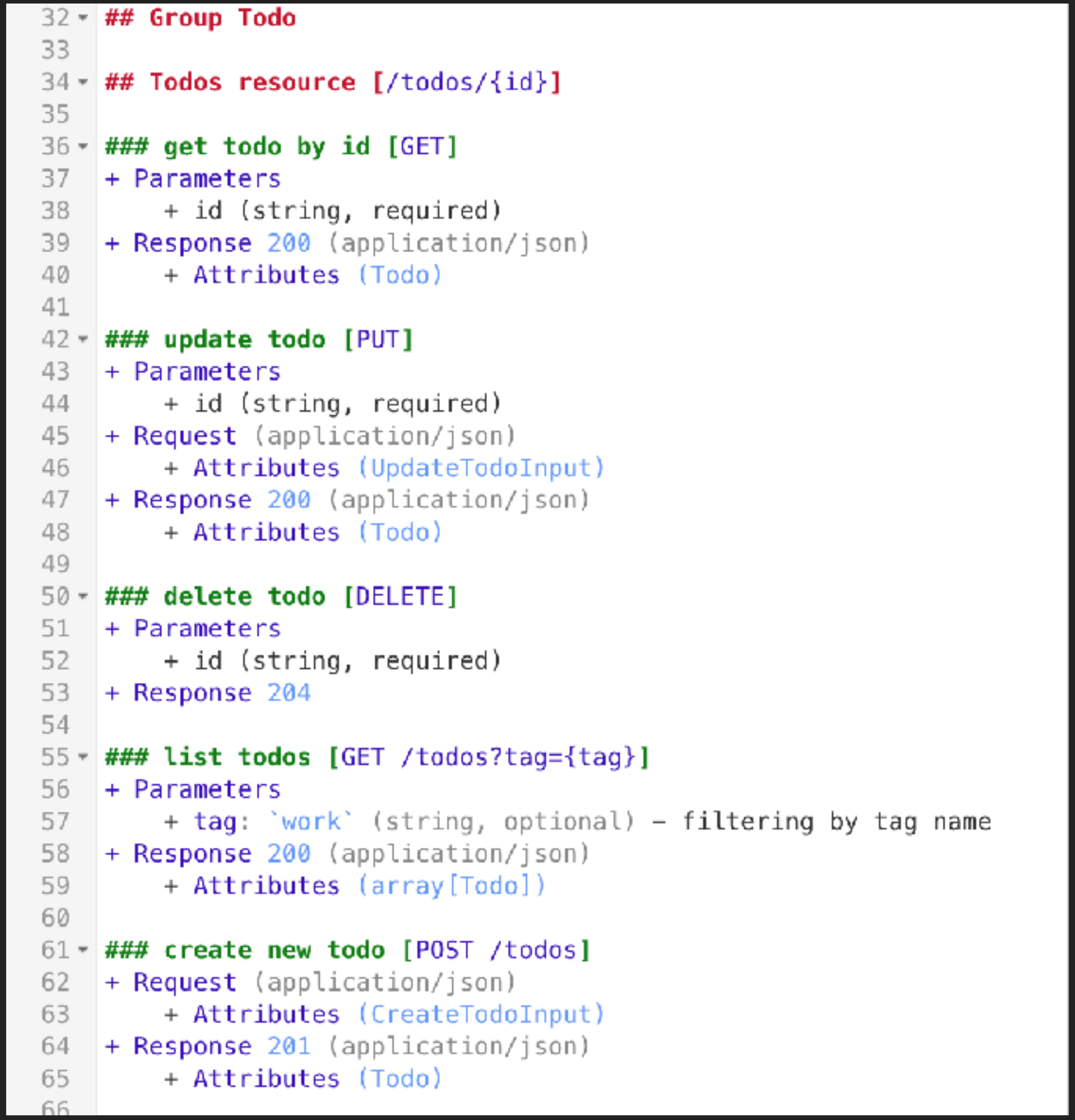
API description language for web services.

Actually, nothing more than just a markdown language.

HOW DOES IT LOOK LIKE?



```
FORMAT: 1A
HOST: http://localhost:8080

# todo api
todo application api

## Data Structures
### Tag
+ id: `1` (number)
+ name: `work` (string)

### CreateTagInput
+ name: `work` (string, required)

### Todo
+ id: `1` (number)
+ name: `prepare slides` (string)
+ done: `false` (boolean)
+ created: `2018-06-12T11:56:40.429+0000` (string)
+ updated: `2018-06-12T12:01:30.789+0000` (string)
+ tags: (array[Tag])

### CreateTodoInput
+ name: 'prepare slides' (string, required)
+ tags: work (array[string], optional)

### UpdateTodoInput
+ name: `updated name` (string, optional)
+ done: `true` (boolean, optional)
+ tags: work, talk (array[string], optional)

```



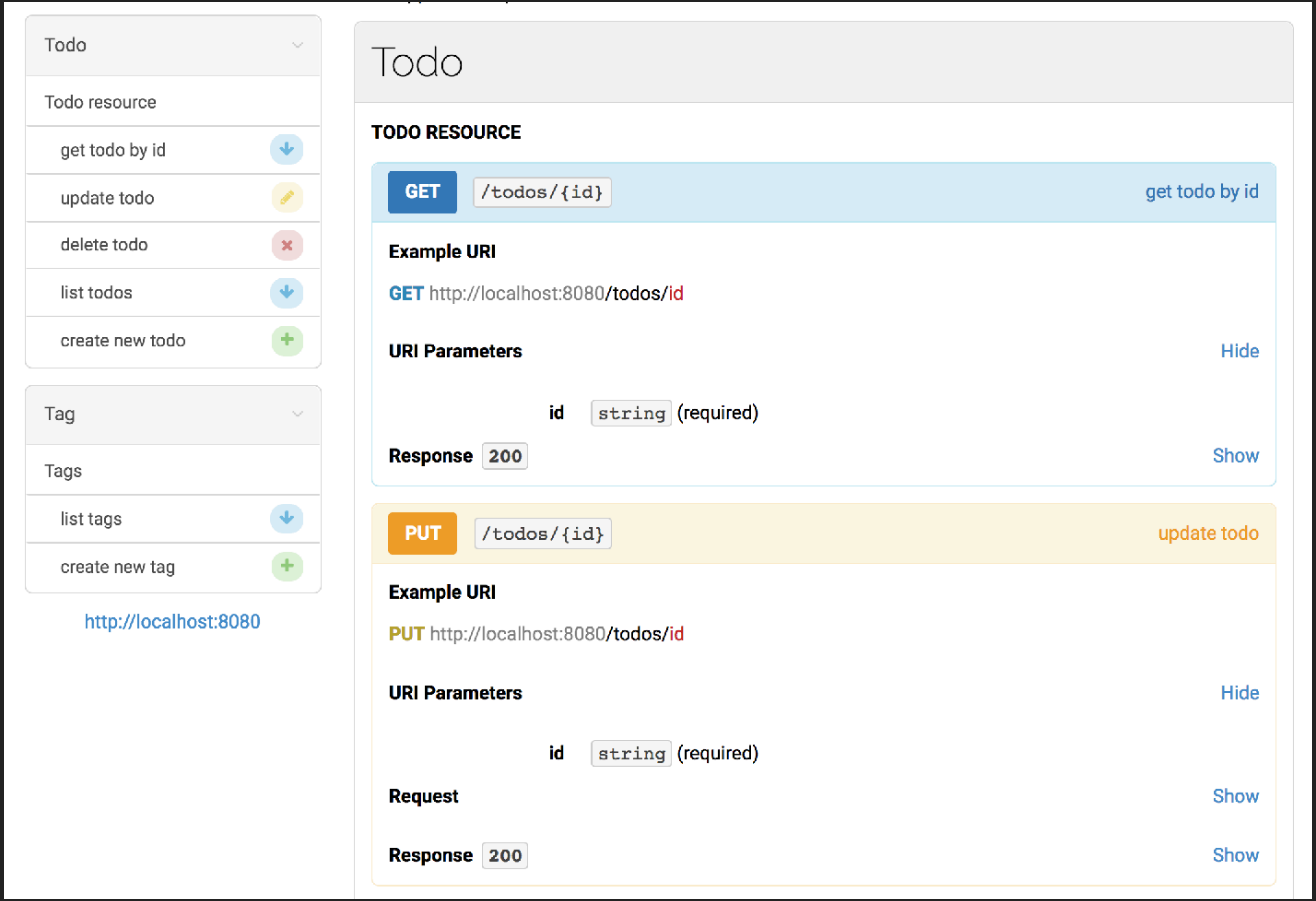
```
## Group Todo

## Todos resource [/todos/{id}]

### get todo by id [GET]
+ Parameters
    + id (string, required)
+ Response 200 (application/json)
    + Attributes (Todo)

### update todo [PUT]
+ Parameters
    + id (string, required)
+ Request (application/json)
    + Attributes (UpdateTodoInput)
+ Response 200 (application/json)
    + Attributes (Todo)

### delete todo [DELETE]
+ Parameters
    + id (string, required)
+ Response 204

### list todos [GET /todos?tag={tag}]
+ Parameters
    + tag: `work` (string, optional) - filtering by tag name
+ Response 200 (application/json)
    + Attributes (array[Todo])

### create new todo [POST /todos]
+ Request (application/json)
    + Attributes (CreateTodoInput)
+ Response 201 (application/json)
    + Attributes (Todo)
```

HOW TO WRITE / GENERATE?

- write manually:
  - apiary
  - atom + plugins
  - sublime + plugins
- no options for generating

HOW CAN IT BE USED?

- render nice doc
- test against implementation (dredd)
- mock server (drake)



Todo

Todo resource

get todo by id

update todo

delete todo

list todos

create new todo

Tag

Tags

list tags

create new tag

http://localhost:8080

Todo

TODO RESOURCE

GET `/todos/{id}` get todo by id

Example URI

GET http://localhost:8080/todos/id

URI Parameters Hide

id `string` (required)

Response `200` Show

PUT `/todos/{id}` update todo

Example URI

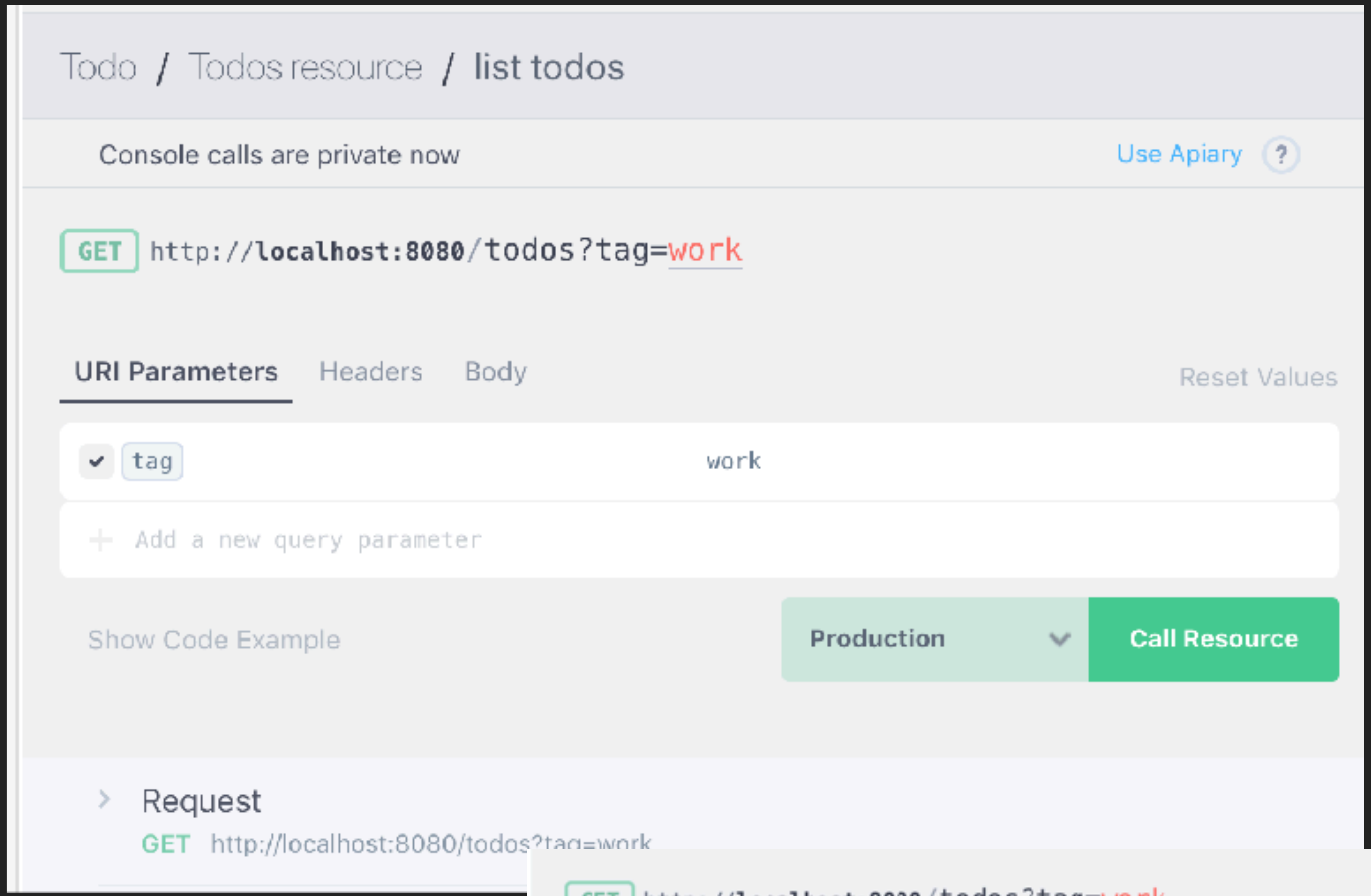
PUT http://localhost:8080/todos/id

URI Parameters Hide

id `string` (required)

Request Show

Response `200` Show


Todo / Todos resource / list todos

Console calls are private now — Use Apiary ?

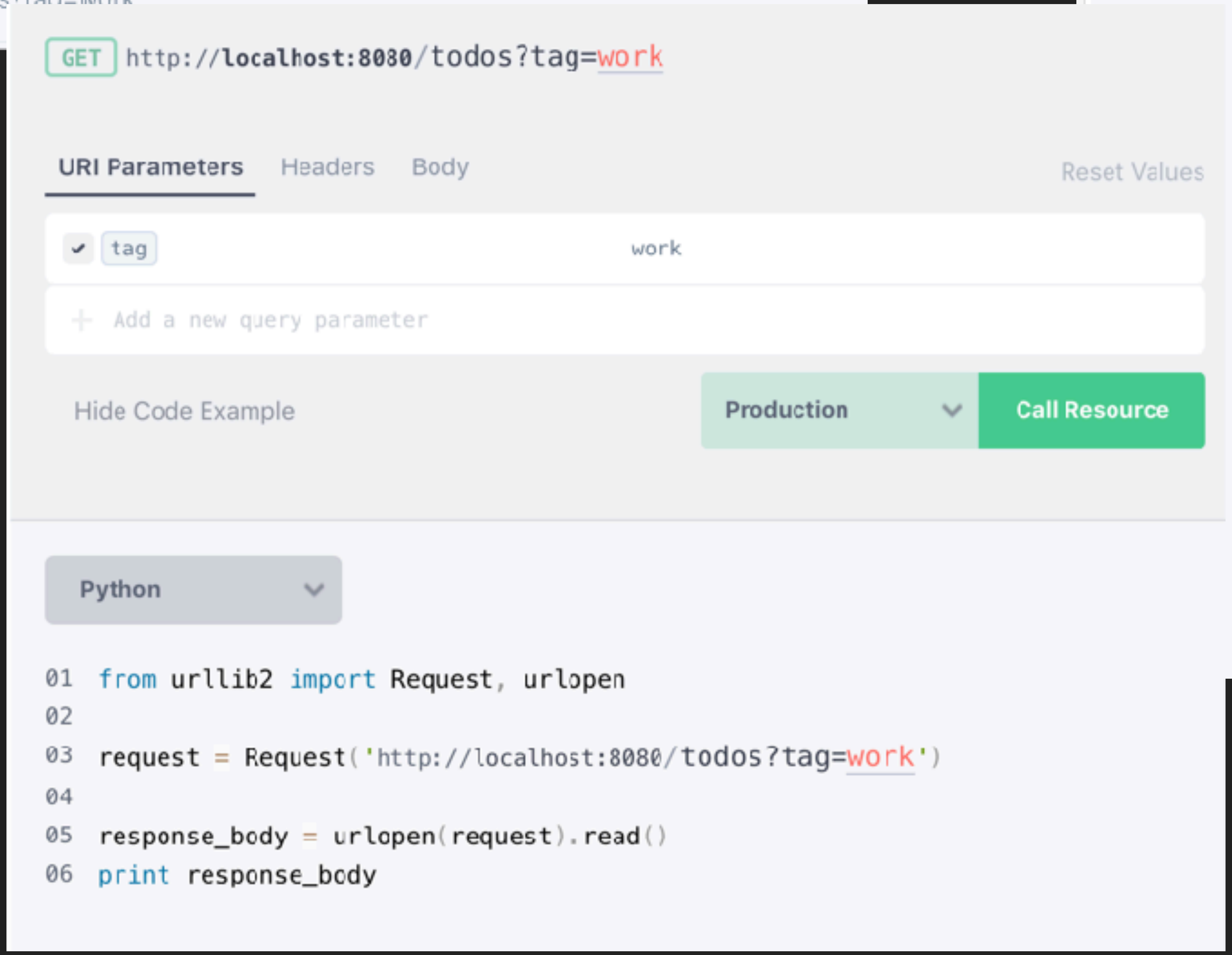
```
GET http://localhost:8080/todos?tag=work
```

URI Parameters | Headers | Body — Reset Values

✓ tag	work
+ Add a new query parameter	

Show Code Example — Production — Call Resource

Request
GET http://localhost:8080/todos?tag=work


```
GET http://localhost:8080/todos?tag=work
```

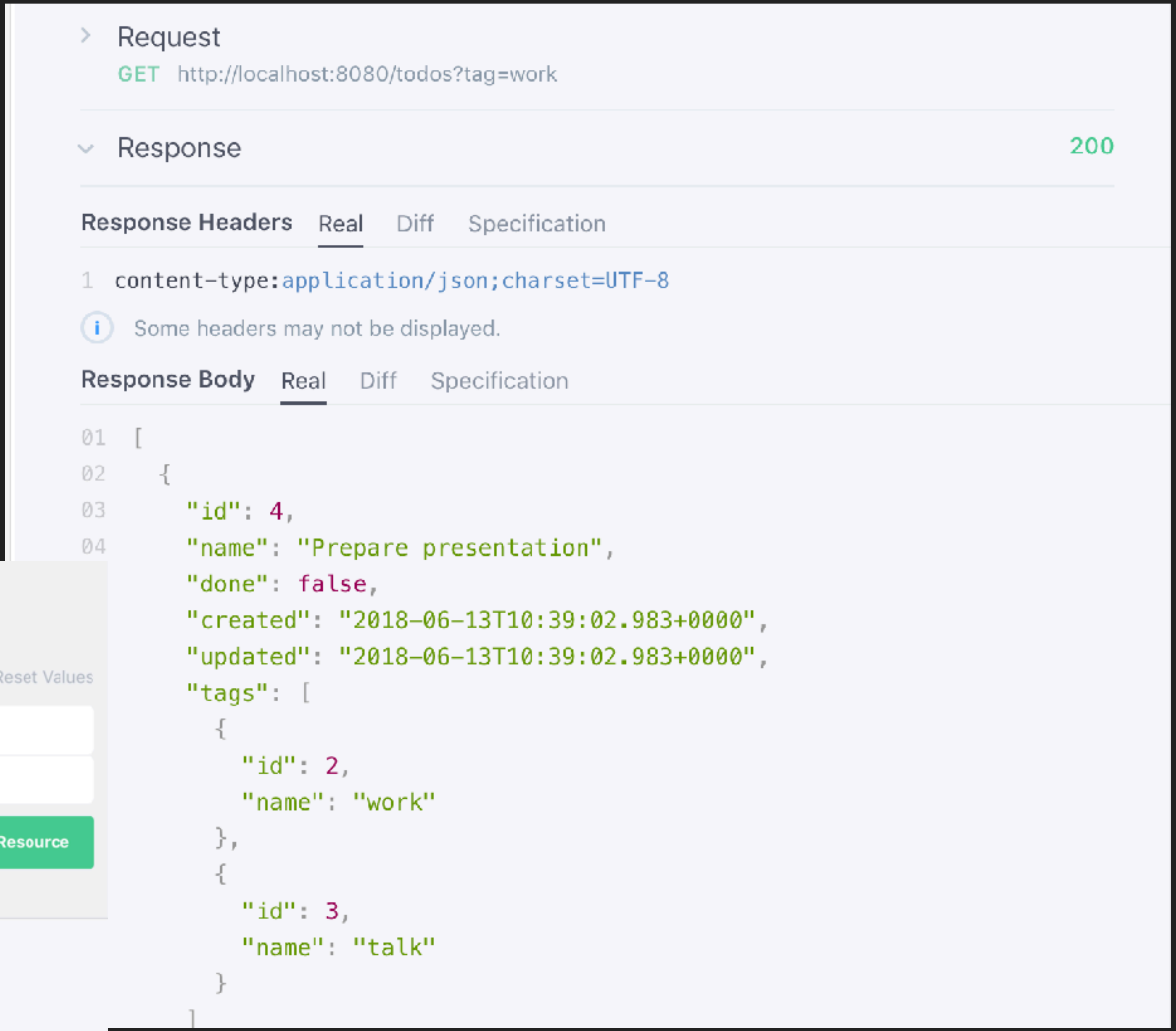
URI Parameters | Headers | Body — Reset Values

✓ tag	work
+ Add a new query parameter	

Hide Code Example — Production — Call Resource

Python

```python
from urllib2 import Request, urlopen

request = Request('http://localhost:8080/todos?tag=work')

response_body = urlopen(request).read()
print response_body
```


Request
GET http://localhost:8080/todos?tag=work

Response 200

Response Headers — Real | Diff | Specification

```
content-type:application/json;charset=UTF-8
```

Some headers may not be displayed.

Response Body — Real | Diff | Specification

```
[
  {
    "id": 4,
    "name": "Prepare presentation",
    "done": false,
    "created": "2018-06-13T10:39:02.983+0000",
    "updated": "2018-06-13T10:39:02.983+0000",
    "tags": [
      {
        "id": 2,
        "name": "work"
      },
      {
        "id": 3,
        "name": "talk"
      }
    ]
```

WHY IS THIS NOT WHAT WE NEED?

- cannot be used for code generation
- cannot be generated from code (except Ruby)

Spring REST Docs

WHAT IS IT?

Library, which turns your integration tests into documentation.

Part of spring ecosystem.

TALK IS CHEAP
SPEC
SHOW ME THE ~~CODE~~

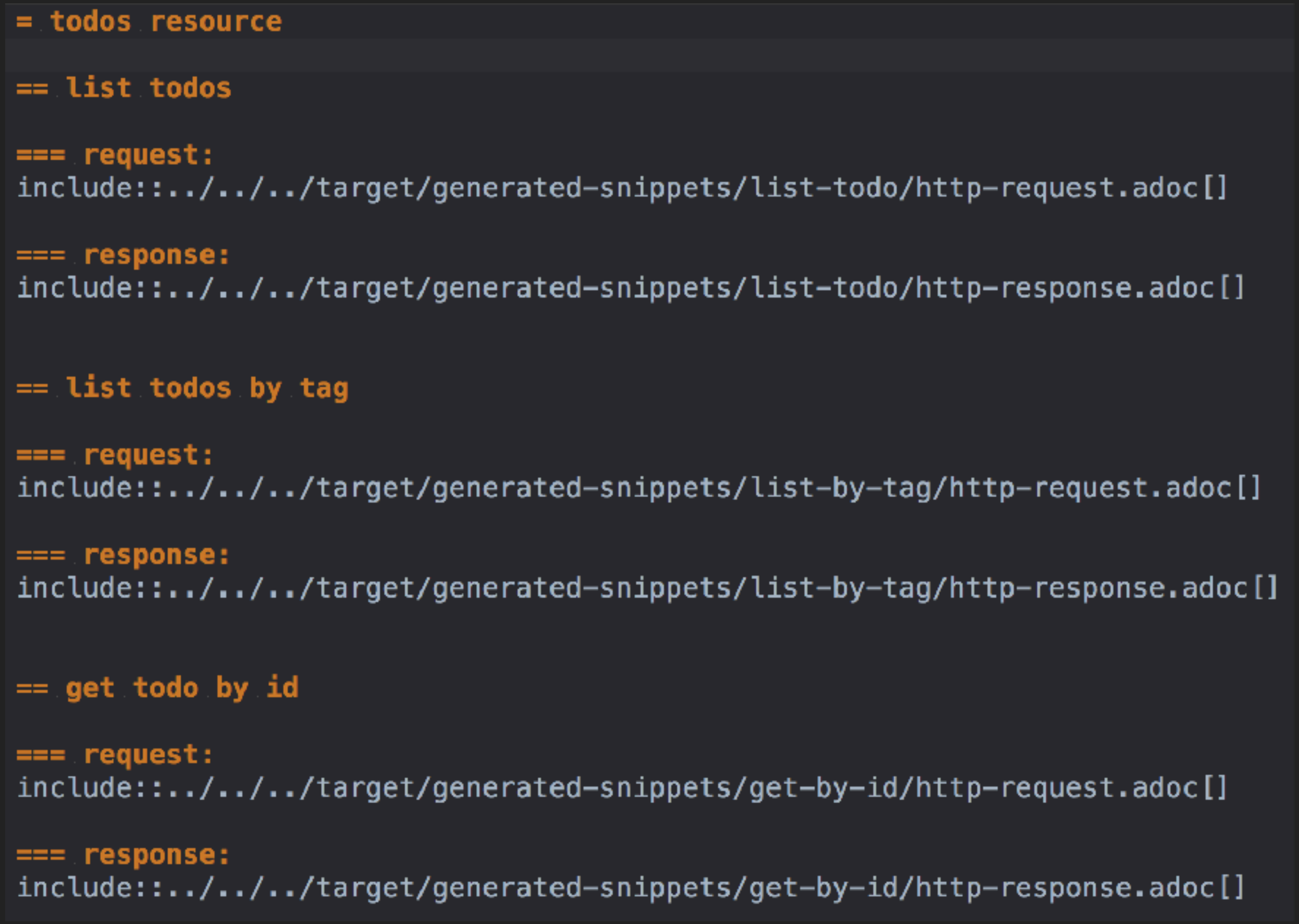
REFERENCE APPLICATION

GET	/todos - retrieve the list of all todo items
POST	/todos - create new todo
GET	/todo/{id} - retrieve todo info by id
PUT	/todo/{id} - update todo info
DELETE	/todo/{id} - delete todo

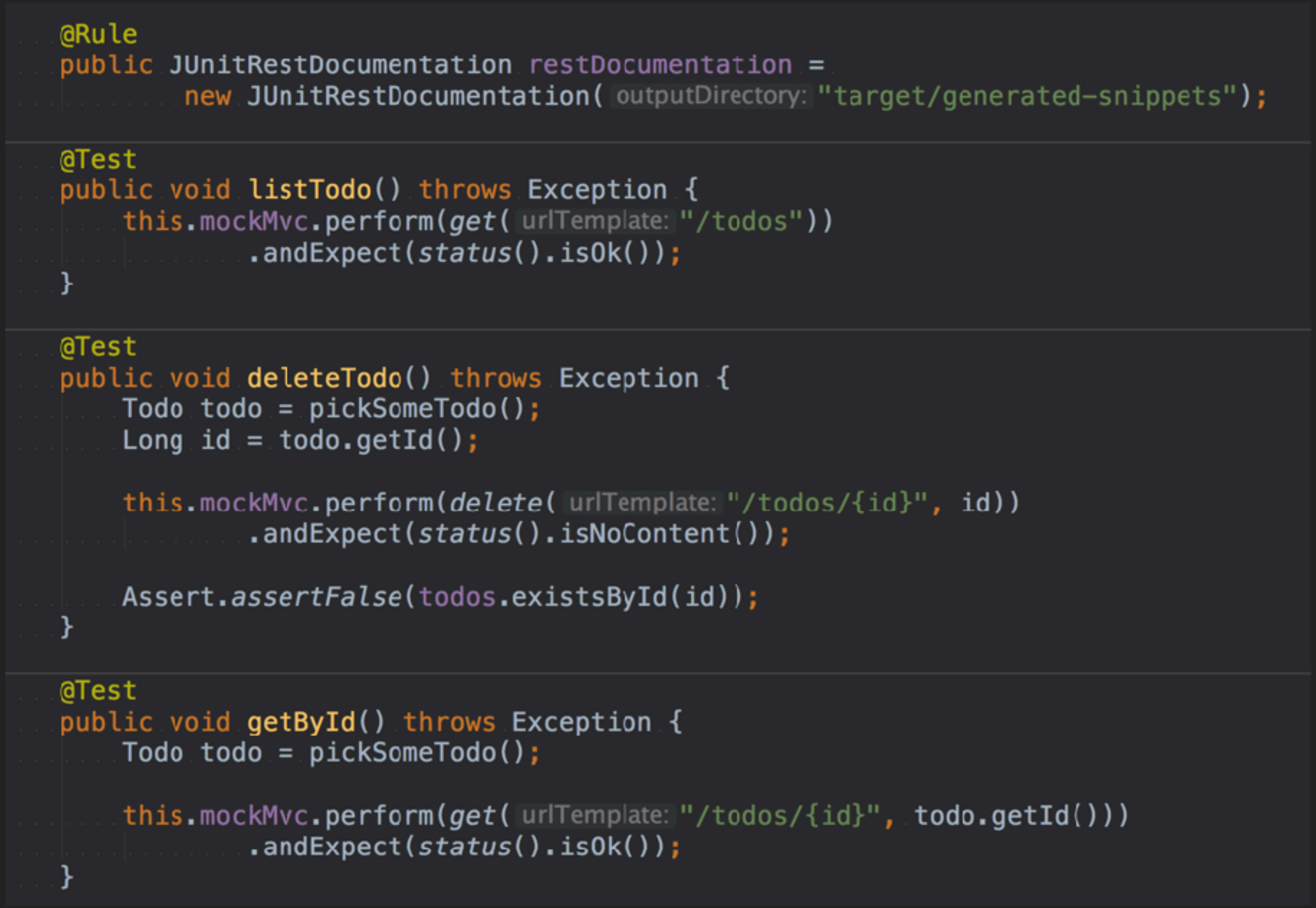
HOW TO WRITE / GENERATE?

- create hand-written ascii doc templates
- write and run integration tests with JUnitRestDocumentation rule

HOW DOES IT LOOK LIKE?



```
= todos resource

== list todos

=== request:
include::../../../target/generated-snippets/list-todo/http-request.adoc[]

=== response:
include::../../../target/generated-snippets/list-todo/http-response.adoc[]


== list todos by tag

=== request:
include::../../../target/generated-snippets/list-by-tag/http-request.adoc[]

=== response:
include::../../../target/generated-snippets/list-by-tag/http-response.adoc[]


== get todo by id

=== request:
include::../../../target/generated-snippets/get-by-id/http-request.adoc[]

=== response:
include::../../../target/generated-snippets/get-by-id/http-response.adoc[]
```



```java
@Rule
public JUnitRestDocumentation restDocumentation =
        new JUnitRestDocumentation( outputDirectory: "target/generated-snippets");

@Test
public void listTodo() throws Exception {
    this.mockMvc.perform(get( urlTemplate: "/todos"))
            .andExpect(status().isOk());
}

@Test
public void deleteTodo() throws Exception {
    Todo todo = pickSomeTodo();
    Long id = todo.getId();

    this.mockMvc.perform(delete( urlTemplate: "/todos/{id}", id))
            .andExpect(status().isNoContent());

    Assert.assertFalse(todos.existsById(id));
}

@Test
public void getById() throws Exception {
    Todo todo = pickSomeTodo();

    this.mockMvc.perform(get( urlTemplate: "/todos/{id}", todo.getId()))
            .andExpect(status().isOk());
}
```

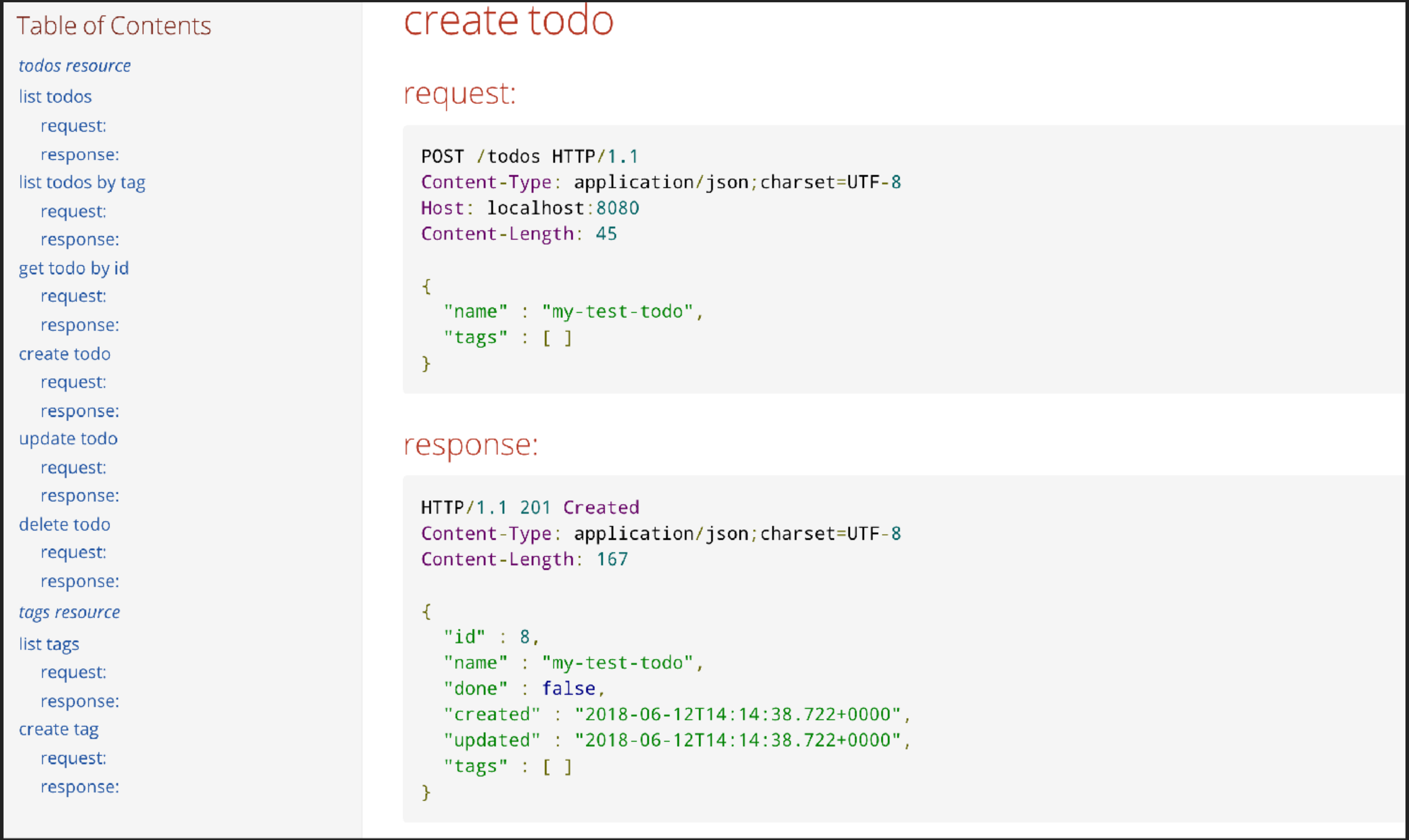


Table of Contents

create todo

request:

```
POST /todos HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: localhost:8080
Content-Length: 45

{
  "name" : "my-test-todo",
  "tags" : [ ]
}
```

response:

```
HTTP/1.1 201 Created
Content-Type: application/json;charset=UTF-8
Content-Length: 167

{
  "id" : 8,
  "name" : "my-test-todo",
  "done" : false,
  "created" : "2018-06-12T14:14:38.722+0000",
  "updated" : "2018-06-12T14:14:38.722+0000",
  "tags" : [ ]
}
```

WHY IS THIS NOT WHAT WE NEED?

- unidirectional
- more about human-readable documentation, not a specification
- requires hand-written templates

RAML

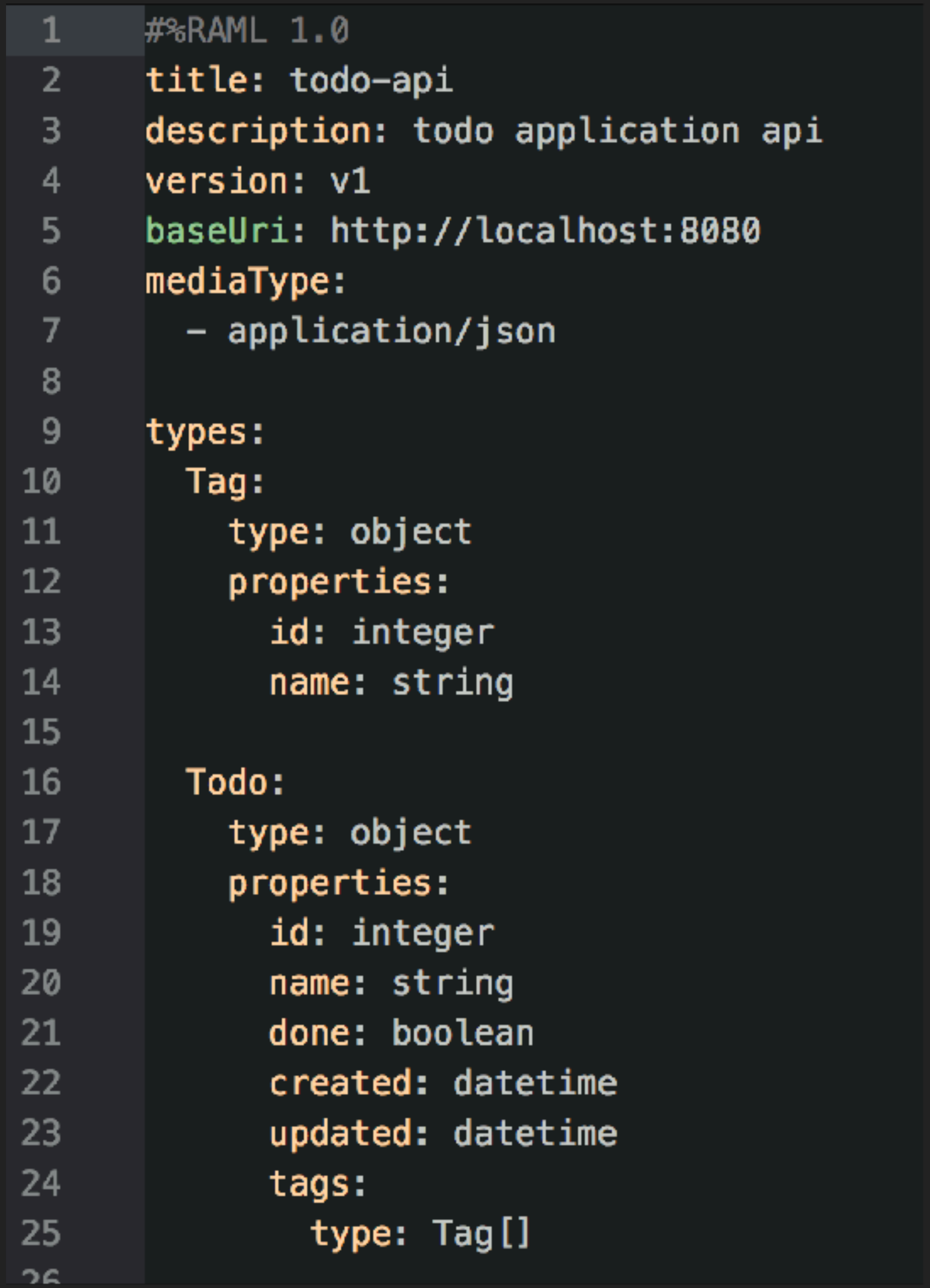
WHAT IS IT?

RESTful API Modeling Language

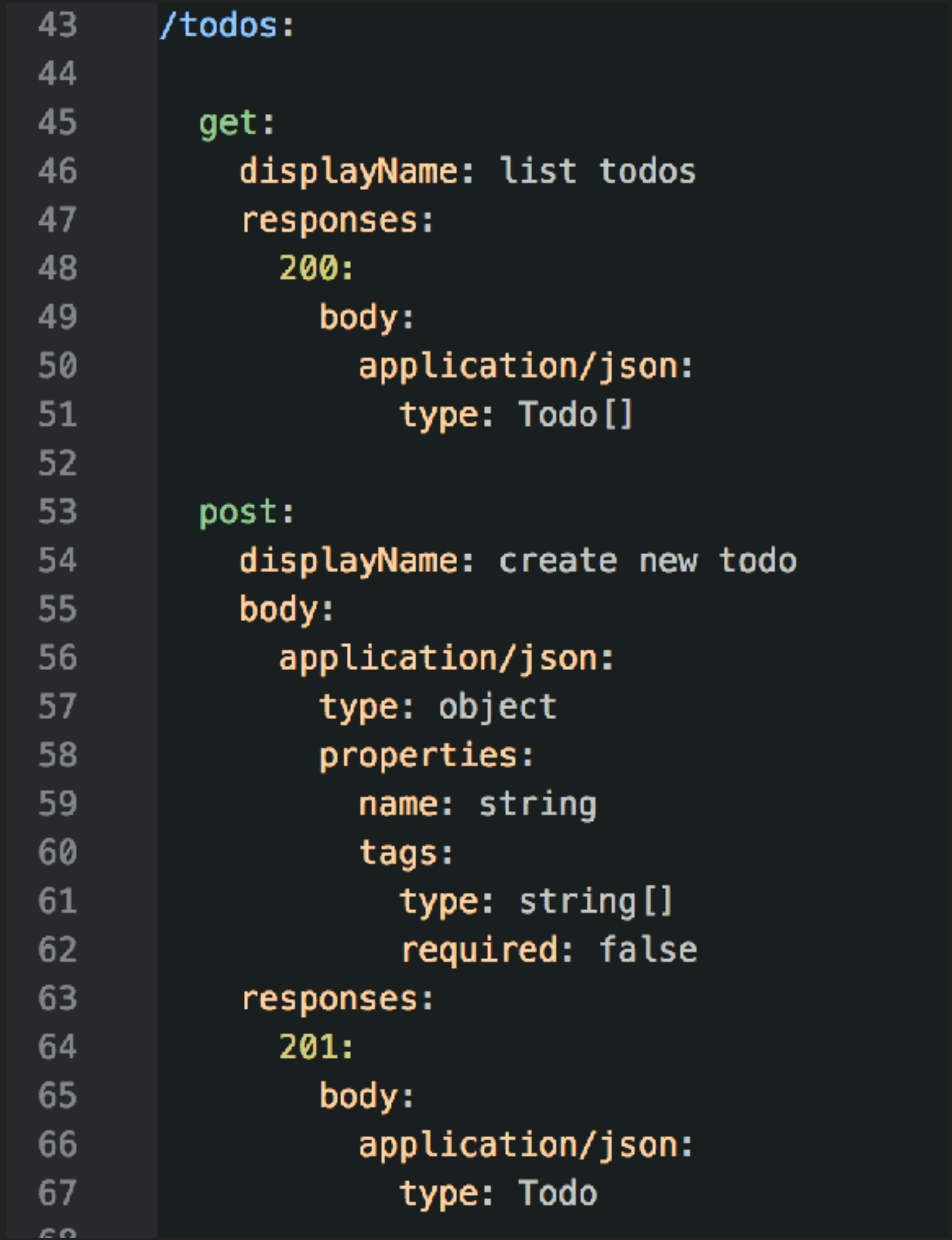
YAML-based language for describing APIs

Developed and supported by Mulesoft

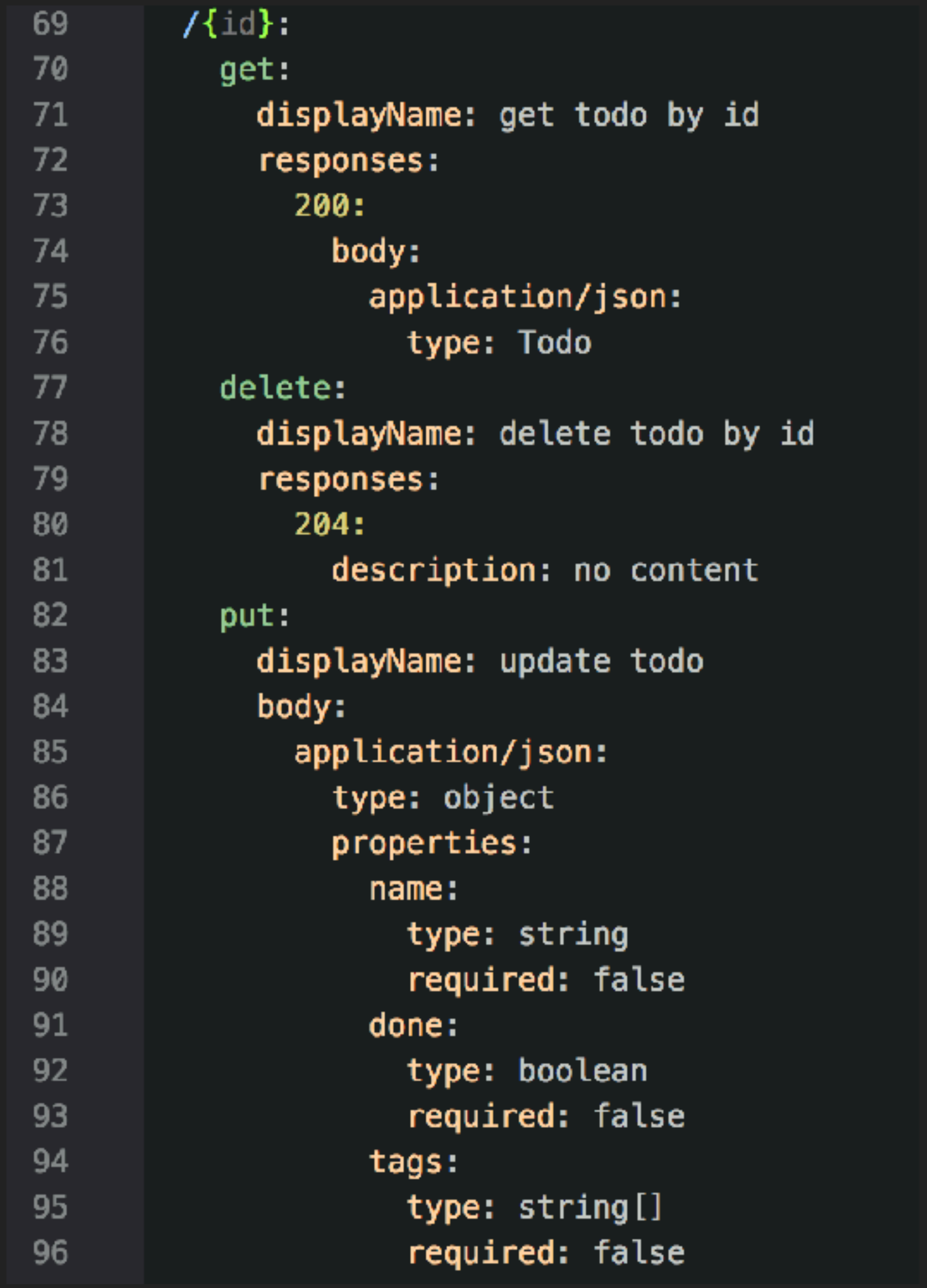
HOW DOES IT LOOK LIKE?



```yaml
#%RAML 1.0
title: todo-api
description: todo application api
version: v1
baseUri: http://localhost:8080
mediaType:
  - application/json

types:
  Tag:
    type: object
    properties:
      id: integer
      name: string

  Todo:
    type: object
    properties:
      id: integer
      name: string
      done: boolean
      created: datetime
      updated: datetime
      tags:
        type: Tag[]
```



```yaml
/todos:

  get:
    displayName: list todos
    responses:
      200:
        body:
          application/json:
            type: Todo[]

  post:
    displayName: create new todo
    body:
      application/json:
        type: object
        properties:
          name: string
          tags:
            type: string[]
            required: false
    responses:
      201:
        body:
          application/json:
            type: Todo
```



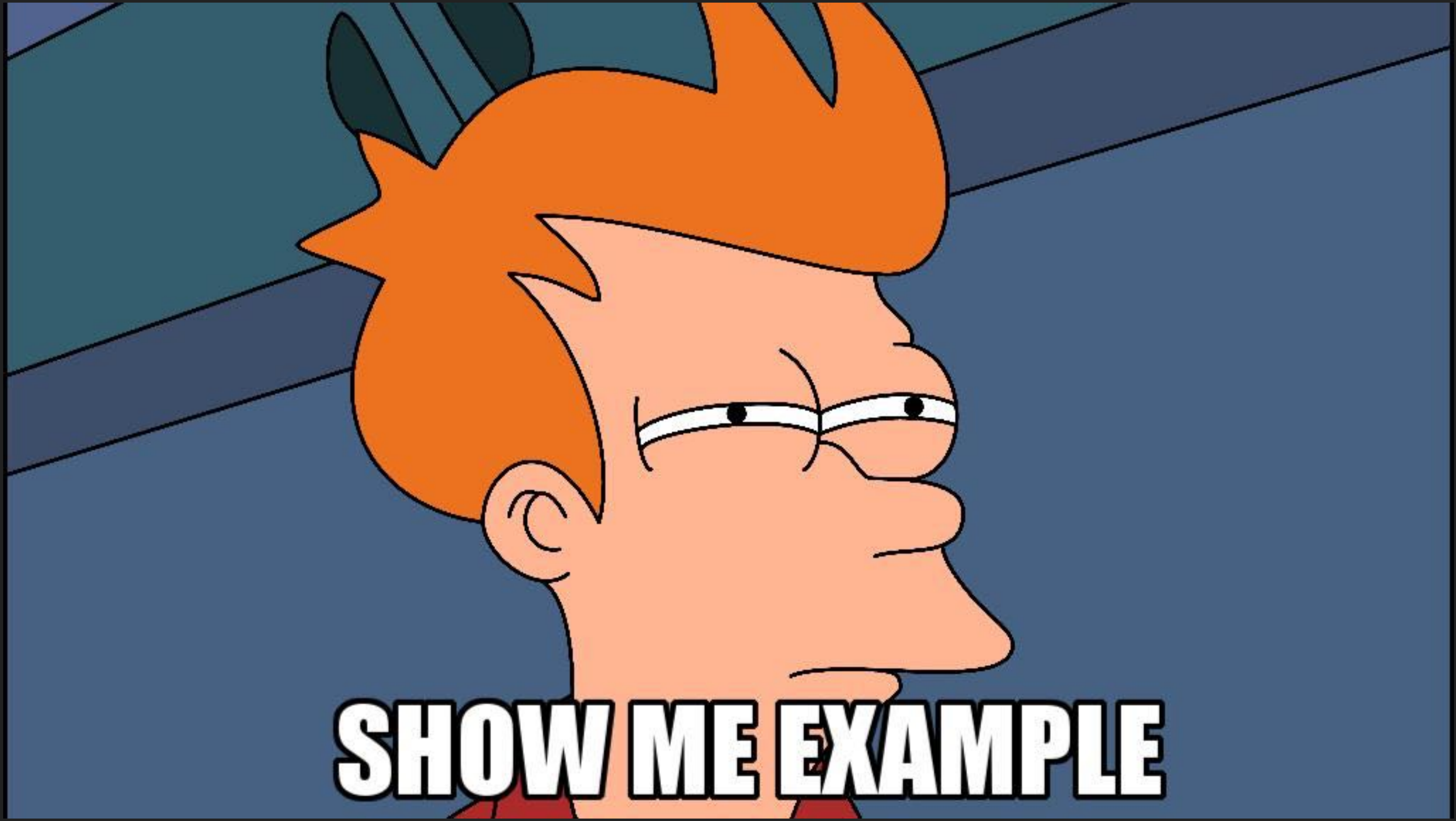
```yaml
/{id}:
  get:
    displayName: get todo by id
    responses:
      200:
        body:
          application/json:
            type: Todo
  delete:
    displayName: delete todo by id
    responses:
      204:
        description: no content
  put:
    displayName: update todo
    body:
      application/json:
        type: object
        properties:
          name:
            type: string
            required: false
          done:
            type: boolean
            required: false
          tags:
            type: string[]
            required: false
```

HOW TO WRITE / GENERATE?

- API Designer
- Atom + API Workbench plugin
- Sublime / IntelliJ via plugins

HOW CAN IT BE USED?

- render static HTML doc
- interactive API console (example)
- setup mock server
- code generation (JAX-RS)


SHOW ME EXAMPLE

WHY MAY IT BE THE OPTION FOR US?

- easy-readable/writable format
- really convenient editors (namely Atom + API Workbench)
- widely adopted by the community
- rich tools ecosystem

WHY IS IT NOT THE BEST CHOICE?

- no options to generate spec for existing codebase
- even authors joined to OpenAPI initiative

OpenAPI

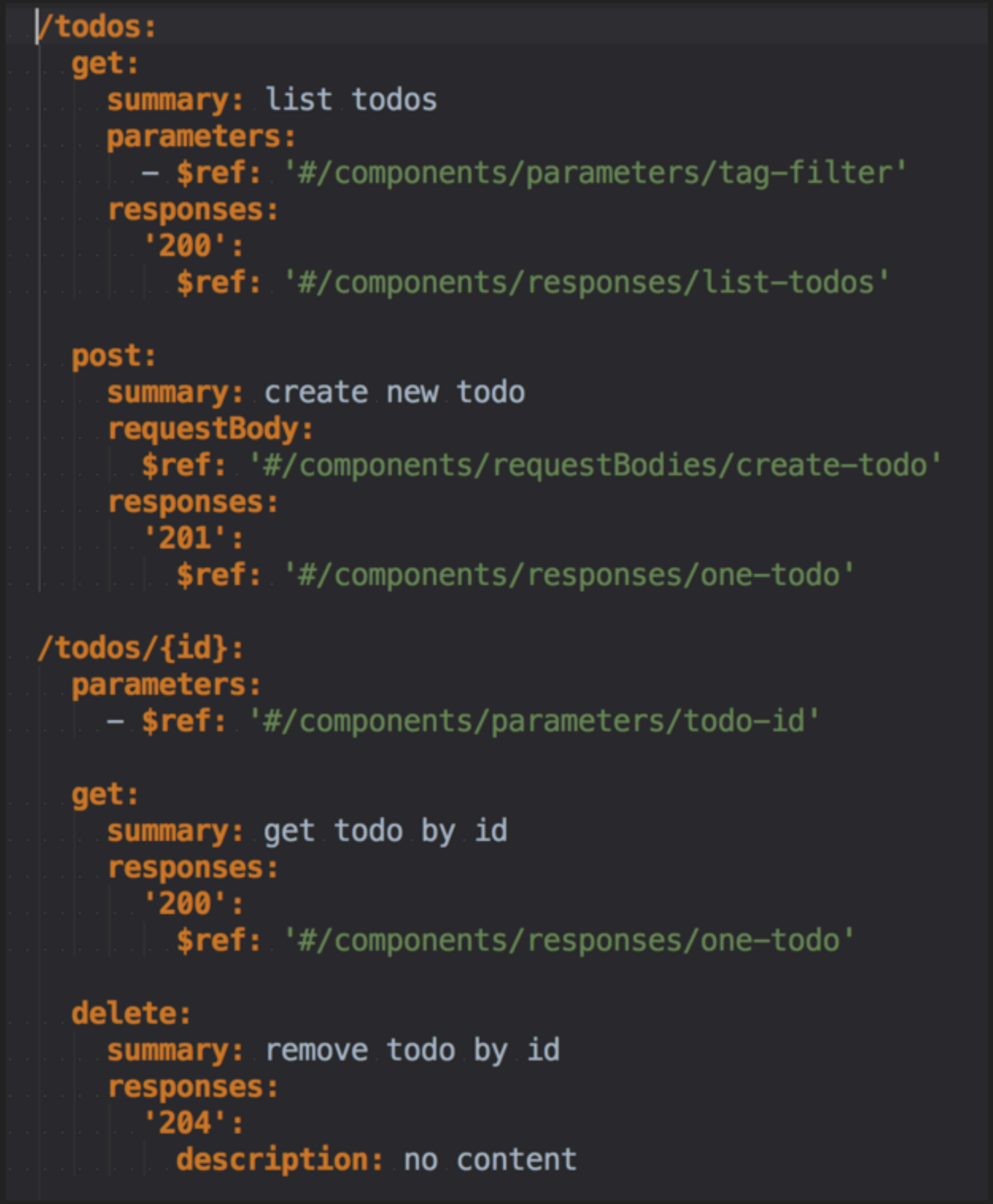
WHAT IS IT?

Specification for describing API specifications.

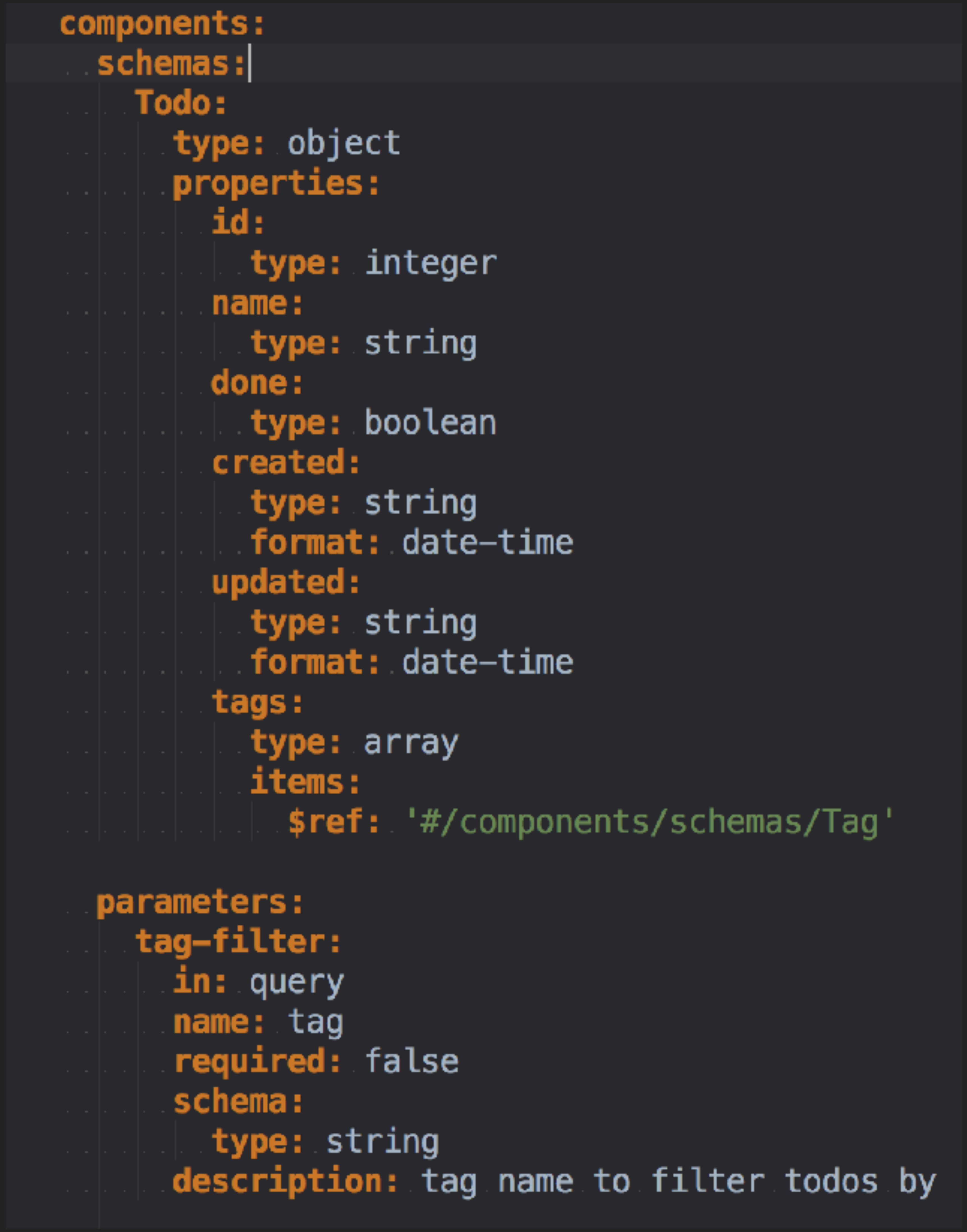
Formerly known as Swagger.

Development is overseen by the Open API Initiative, an open source collaborative project of the Linux Foundation.

HOW DOES IT LOOK LIKE?



```yaml
/todos:
  get:
    summary: list todos
    parameters:
      - $ref: '#/components/parameters/tag-filter'
    responses:
      '200':
        $ref: '#/components/responses/list-todos'

  post:
    summary: create new todo
    requestBody:
      $ref: '#/components/requestBodies/create-todo'
    responses:
      '201':
        $ref: '#/components/responses/one-todo'

/todos/{id}:
  parameters:
    - $ref: '#/components/parameters/todo-id'

  get:
    summary: get todo by id
    responses:
      '200':
        $ref: '#/components/responses/one-todo'

  delete:
    summary: remove todo by id
    responses:
      '204':
        description: no content
```



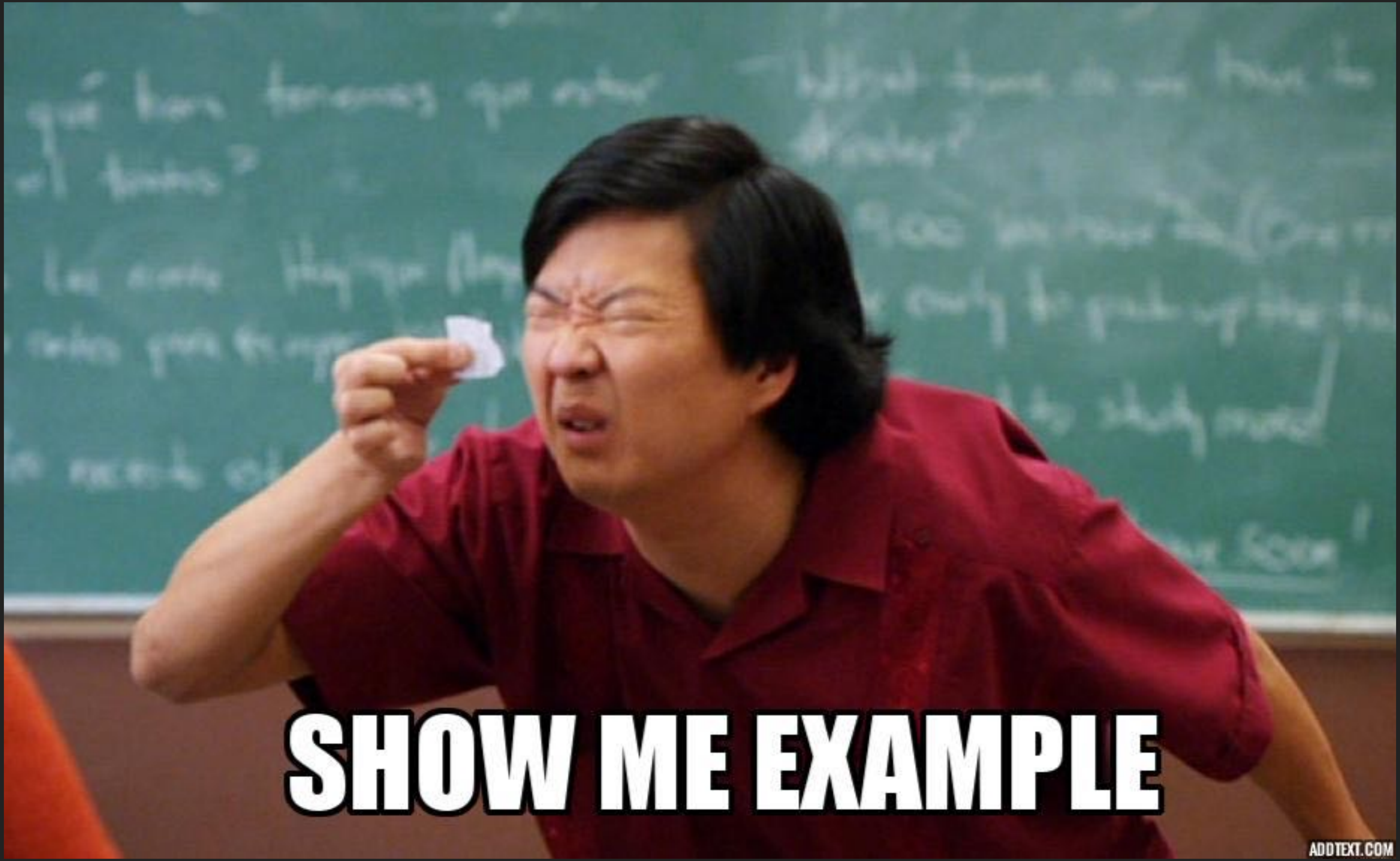
```yaml
components:
  schemas:
    Todo:
      type: object
      properties:
        id:
          type: integer
        name:
          type: string
        done:
          type: boolean
        created:
          type: string
          format: date-time
        updated:
          type: string
          format: date-time
        tags:
          type: array
          items:
            $ref: '#/components/schemas/Tag'

  parameters:
    tag-filter:
      in: query
      name: tag
      required: false
      schema:
        type: string
      description: tag name to filter todos by
```

HOW TO WRITE / GENERATE?

- swagger-hub / swagger-editor
- IntelliJ / Atom / Sublime + plugins
- generate from code (springfox, enunciate)

HOW CAN IT BE USED?

- client / server-side code generation for different languages
- mocks via swagger-hub / third-party tools
- render docs / interactive console (swagger-ui)


SHOW ME EXAMPLE
ADDTEXT.COM

WHY IS IT SO COOL?

- recognized by the community as a standard
- rich toolset
- meets all our requirements

DRAWBACKS

- versions (v2 vs v3)
- subjectively not so convenient editor, comparing to the RAML (Atom + API Workbench)

Summary

CONVERSION

- oas-raml-converter-cli
- web-version oas-raml-converter
- apimatic.io

MAKE OUR API SPECS GREAT AGAIN

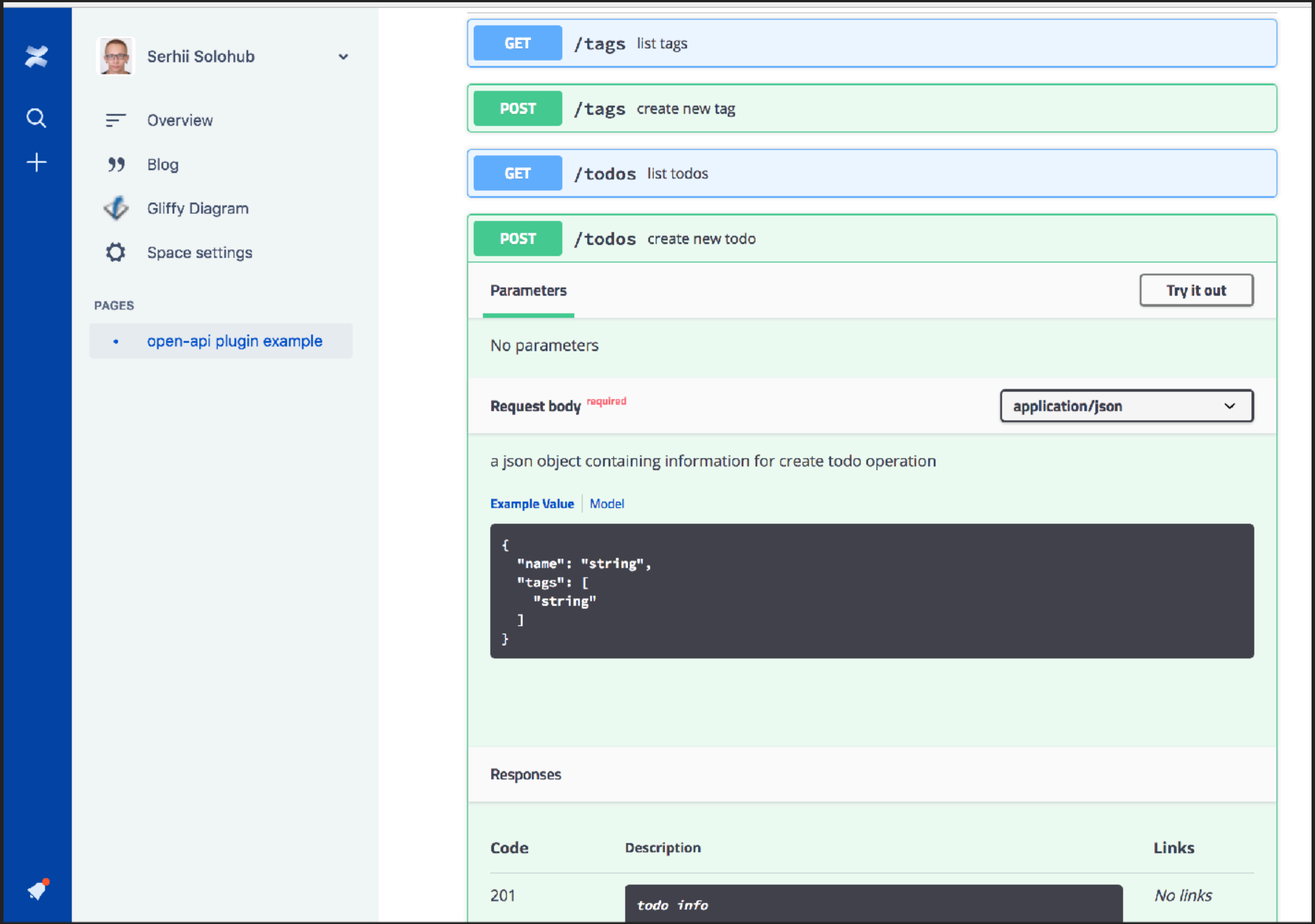
- ~~datatypes ambiguity~~
- ~~Single Source of Truth violation~~
- ~~absence of modeling tools~~
- ~~inability to generate changelog~~

- strict specification format
- components reusability
- atom / intellij / swagger-editor
- swagger-diff

BONUS

Confluence integration:

- rendering
- publishing



GET /tags list tags

POST /tags create new tag

GET /todos list todos

POST /todos create new todo

Parameters

Try it out

No parameters

Request body required

application/json

a json object containing information for create todo operation

Example Value | Model

```
{
  "name": "string",
  "tags": [
    "string"
  ]
}
```

Responses

Code	Description	Links
201	*todo info*	*No links*

QUESTIONS?

Sources and slides:

https://github.com/q1nt/api-specs-talk