

# REST API tests in less than a minute

Marcin Maj

June 2023

# Agenda

**01** Problems to solve

---

**02** Open API / Swagger

---

**03** Generate clients and servers

---

**04** Demo – client generate and test

---

**05** Integration scenarios

---

**06** GitHub

---

You are here



## 01 Problems to solve

---

02 Open API / Swagger

---

03 Generate clients and servers

---

04 Demo – client generate and test

---

05 Integration scenarios

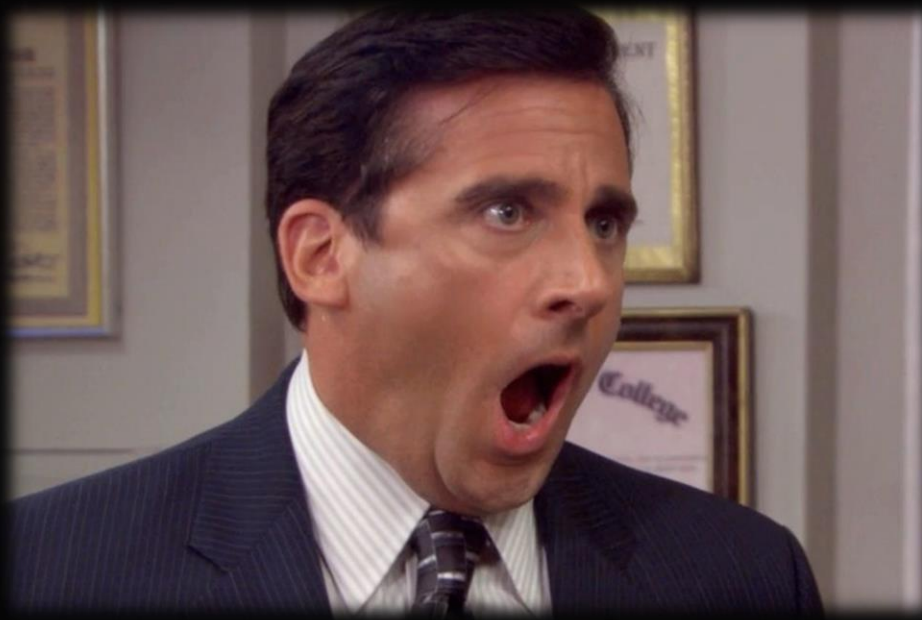
---

06 GitHub

---

# Problem: Codeless maintenance

Effort needed to maintain codeless, almost codeless or tools solutions



Name

Get Conversations

Description Authorization Pre-request Scripts Tests ●

These tests will execute after every request in this folder. [Learn more about Postman's execution order.](#)

```
1 const responseBody = pm.response.json().content;
2
3 if (responseCode.code === 200 && !pm.info.requestName.includes('Count')) {
4
5   pm.test(`${pm.info.requestName}: ` + 'Each object and all nested objects have correct keys.', () => {
6     //each object
7     responseBody.forEach(body => {
8       pm.expect(body, `${pm.info.requestName}: ` + 'Response body keys are not correct.')
9         .to.have.keys('id', 'phone', 'type', 'status', 'userId', 'hasAttachments',
10          'hasSMS', 'interactionsCount', 'unseenCount', 'preview', 'info',
11          'customer', 'createdAt', 'updatedAt', 'lastIncomingAt', 'lastRepliedAt',
12          'lastAssignedAt');});
13
14     //customer objects
15     responseBody.forEach(body => {
16       pm.expect(body.customer, `${pm.info.requestName}: ` + 'customer object keys are not correct.')
17         .to.have.keys('id', 'firstName', 'lastName');});
18
19     //info objects
20     var i;
21     for (i = 0; i < responseBody.length; i++){
22       if (responseBody[i].type === "call"){
23         pm.expect(responseBody[i].info, `${pm.info.requestName}: ` + 'info object keys are not correct.')
24           .to.have.keys('direction', 'flags', 'callID', 'result');
25       }
26       else {
27         pm.expect(responseBody[i].info, `${pm.info.requestName}: ` + 'info object keys are not correct.')
28           .to.have.keys('direction');
29       }
30     }
31   })
32 }
```

## Problem: Boiler plate code

To stop generate boiler plate code. Focus on technical excellence and productivity of testing solutions

NO, JUST NO.

Generate Plain Old Java Objects from JSON or JSON-Schema.

```
1  {
2    "element count": 101,
3    "near earth objects": {
4      "2020-04-12": {
5        "links": {
6          "self": "http://www.neowsapp.com/rest/v1/neo/38425967aj"
7        },
8        "id": "3842596",
9        "neo reference id": "3842596",
10       "name": "(2019 KM2)",
11       "nasa_jpl url": "http://ssd.jpl.nasa.gov/sbdb.cgi?sstr=3842596",
12       "absolute magnitude h": 25.5,
13       "estimated diameter": {
14         "kilometers": {
15           "estimated_diameter_min": 0.0211132445,
16           "estimated_diameter_max": 0.0472106499
17         },
18         "meters": {
19           "estimated_diameter_min": 21.113244479,
20           "estimated_diameter_max": 47.2106498806
21         },
22         "miles": {
23           "estimated_diameter_min": 0.0131191578,
24           "estimated_diameter_max": 0.0293353287
25         },
26         "feet": {
27           "estimated_diameter_min": 69.2691778164,
28           "estimated_diameter_max": 154.8905885541
29         }
30       },
31       "is potentially hazardous asteroid": false,
32       "close approach data": {
33         "close approach date": "2020-04-12",
34         "close approach date full": "2020-Apr-12 04:06",
35         "epoch date close approach": 1586664360000,
36         "relative velocity": {
37           "kilometers per second": "10.3161466279",
38           "kilometers per hour": "37130.1278605498",
39           "miles per hour": "23076.1883920177"
40         },
41         "miss distance": {
42           "astronomical": "0.2878000908",
43           "lunar": "111.9545465212",
44           "kilometers": "43054400.247782596",
45           "miles": "26752763.7718319848"
46         },
47         "orbiting body": "Earth"
48       },
49       "is sentry object": false
50     }
51   }
52 }
```

Preview

Zip

NeoWsData-sources.zip

Package com.example

Class name NeoWsData

Target language:

☒ Java ☐ Scala

Source type:

☐ JSON Schema ☒ JSON

☐ YAML Schema ☐ YAML

Annotation style:

☒ Jackson 2.x ☐ Jackson 1.x

☐ Gson ☐ Moshi ☐ None

☐ Generate builder methods

☒ Use primitive types

☒ Use long integers

☒ Use double numbers

☒ Use Joda dates

☐ Use Commons-Lang3

☐ Include getters and setters

☐ Include constructors

☐ Include hashCode and equals

☐ Include toString

☐ Include JSR-303 annotations

☐ Allow additional properties

☐ Make classes serializable

☐ Make classes parcelable

☐ Initialize collections

Property word delimiters: - \_

## Problem: Weak code quality

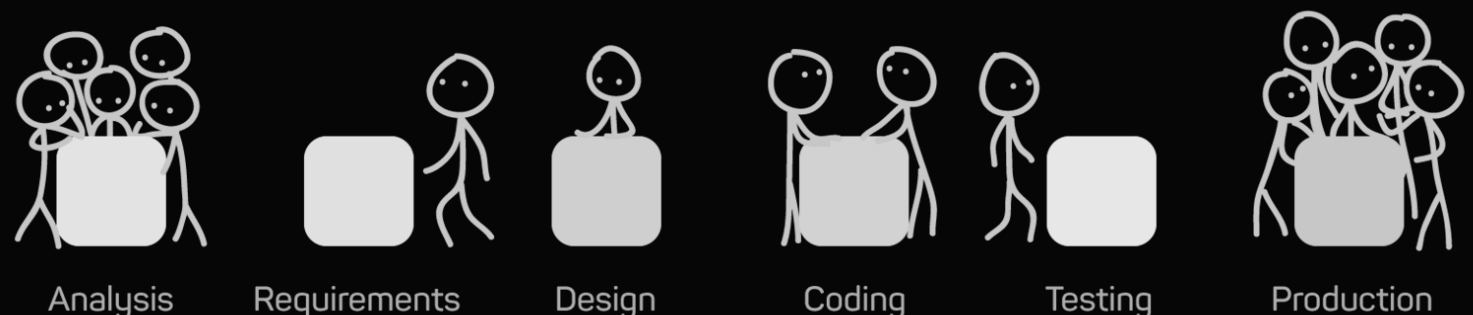
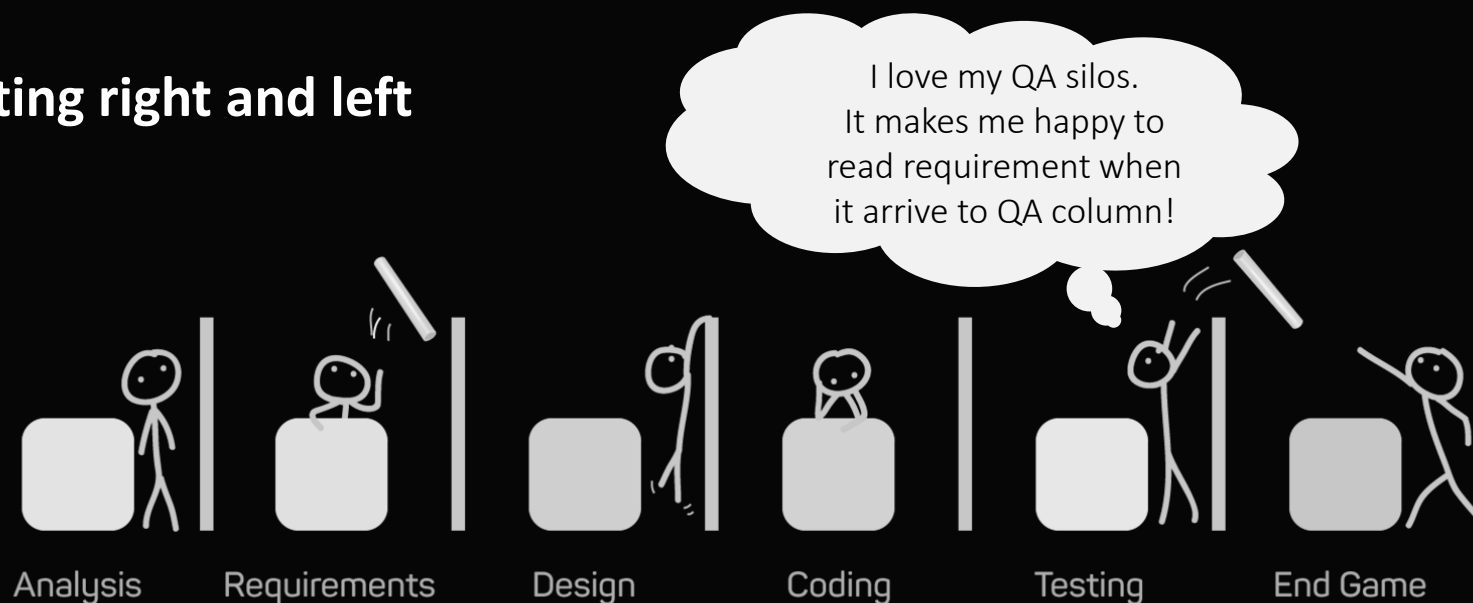
To start using proper tools and create high quality code



```
1 package JsonPathForJsonObject;
2
3 import io.restassured.path.json.JsonPath;
4
5 public class NestedJsonObject {
6
7     public static void main(String[] args) {
8
9         String jsonString = "{\r\n" +
10             "  \"firstName\": \"Amod\", \r\n" +
11             "  \"lastName\": \"Mahajan\", \r\n" +
12             "  \"address\": {\r\n" +
13             "    \"houseNo\": 404, \r\n" +
14             "    \"buildingName\": \"Not Found\", \r\n" +
15             "    \"streetName\": \"Gumnam gali\", \r\n" +
16             "    \"city\": \"Bengaluru\", \r\n" +
17             "    \"state\": \"Karnataka\", \r\n" +
18             "    \"country\": \"India\" \r\n" +
19             "  }, \r\n" +
20             "  \"skills\": {\r\n" +
21             "    \"language\": {\r\n" +
22             "      \"name\": \"Java\", \r\n" +
23             "      \"proficiency\": \"Medium\" \r\n" +
24             "    } \r\n" +
25             "  } \r\n" +
26             "}";
27
28         //Get JsonPath instance of above JSON string
29         JsonPath jsonPath = JsonPath.from(jsonString);
30
31         // Since houseNo holds an int value use getInt() method and provide json path
32         int houseNo = jsonPath.getInt("address.houseNo");
33         System.out.println("House no is : "+houseNo);
34
35         String name = jsonPath.getString("skills.language.name");
36         System.out.println("Name is : "+name);
37
38     }
39
40 }
```



## Problem: Shifting right and left



Common understanding, prototyping and shifting LEFT

QA Column

I LOVE IT SO MUCH



GREAT JOB!

YOU ROCK!

You are here



- 01 Problems to solve

---

- 02 Open API / Swagger**

---

- 03 Generate clients and servers

---

- 04 Demo – client generate and test

---

- 05 Integration scenarios

---

- 06 GitHub

---



## What is Open API Specification

*The **OpenAPI Specification**, previously known as the **Swagger Specification**,*

*is a specification for a machine-readable **interface definition language** for*

- ***describing,***
- ***producing,***
- ***consuming***
- ***visualizing web services.***

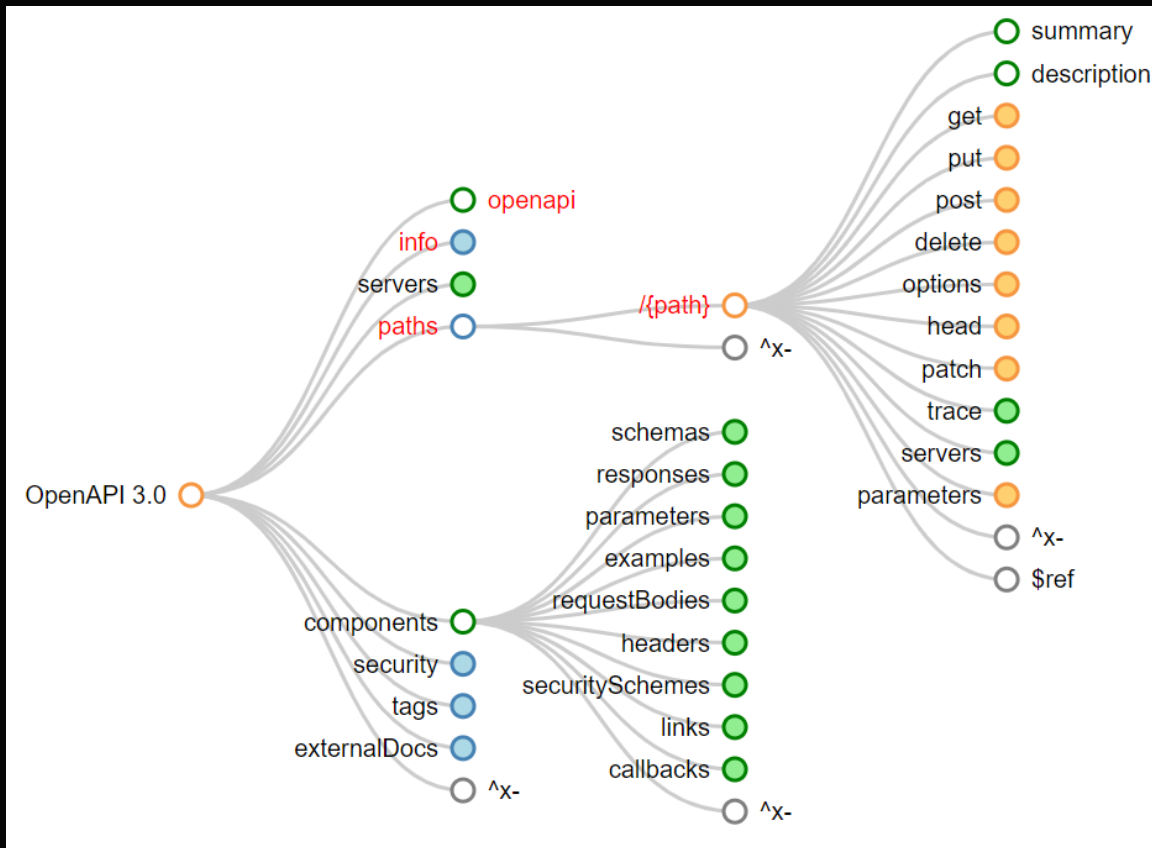
*Previously part of the Swagger framework, it became a separate project in 2016, overseen by the OpenAPI Initiative, an open-source collaboration project of the Linux Foundation.*

*Swagger and some other tools can*

- ***generate code,***
- ***documentation***
- ***and test cases***

*from interface files.*

# What's inside



```

openapi: 3.0.1
info:
  title: Spring PetClinic
  description: Spring PetClinic Sample Application.
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0
  version: '1.0'
servers:
  - url: http://localhost:9966/petclinic/api
  
```

```

openapi.yml C:\epam\spring-petclinic-rest\target\
  < > YAML document
    > openapi 3.0.1
    < > info
      > title Spring PetClinic
      > description Spring PetClinic Sample App
      < > license
      > version 1.0
    > < > servers
    > < > tags
    > < > paths
    > < > components
  
```

```

paths:
  /oops: <1 key>
  /owners: <2 keys>
    /owners/{ownerId}: <3 keys>
      /owners/{ownerId}/pets: <1 key>
      /owners/{ownerId}/pets/{petId}: <2 keys>
      /owners/{ownerId}/visits: <1 key>
    /pettypes: <2 keys>
      /pettypes/{petTypeId}: <3 keys>
  /pets: <2 keys>
    /pets/{petId}: <3 keys>
    /visits: <2 keys>
  
```

<https://swagger.io/specification/>  
<https://spec.openapis.org/oas/v3.1.0>  
<https://openapi-map.apihandyman.io/?version=3.1>

## Open API Spec and „Swagger view“

The image displays a comparison between an OpenAPI specification (left) and its visualization in Swagger UI (right).

**OpenAPI Specification (Left):**

```
paths:
  /pets/{petId}:
    get:
      tags:
        - pet
      operationId: getPet
      summary: Get a pet by ID
      description: Returns the pet or a 404 error.
      parameters:
        - name: petId
          in: path
          description: The ID of the pet.
          required: true
          schema:
            type: integer
            format: int32
            minimum: 0
            example: 1
      responses:
        200:
          description: Pet details found and returned.
          headers:
            ETag:
              description: An ID for this version of the response.
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Pet'
        304: {}
        400: {}
        404: {}
        500: {}
    put:
      tags:
        - pet
      operationId: updatePet
      summary: Update a pet by ID
      description: Returns the pet or a 404 error.
```

**Swagger UI (Right):**

The Swagger UI provides a user-friendly interface for the API. It includes:

- GET /pets/{petId}**: Operation title and summary.
- Returns the pet or a 404 error.**: Description of the endpoint's behavior.
- Parameters**: A section listing the required `petId` parameter (integer, path). A text input field shows the value `1`.
- Responses**: A table showing the expected status codes and their descriptions.

Code	Description	Links
200	Pet details found and returned.	No links
- Media type**: A dropdown menu set to `application/json`.
- Example Value**: A JSON object representing a pet:

```
{
  "name": "Leo",
  "birthDate": "2010-09-07",
  "type": {
    "name": "cat",
    "id": 1
  },
  "id": 1,
  "ownerId": 1,
  "visits": [
    {
      "date": "2013-01-01",
      "description": "first visit"
    }
  ]
}
```

# Open API Spec and „Swagger view“

The screenshot displays an IDE with three panes illustrating the OpenAPI specification and its Swagger view.

**Structure Pane (Left):** Shows the hierarchy of the `openapi.yml` file. The `components` section is expanded, and the `Pet` schema is selected under `schemas`.

**YAML Document Pane (Middle):** Displays the raw OpenAPI specification for the `Pet` schema. A red box highlights the `id` property definition:

```
id:  
  title: ID  
  description: The ID of the pet.  
  type: integer  
  format: int32  
  minimum: 0  
  example: 1  
  readOnly: true
```

**Pet fields Pane (Right):** Displays the Swagger view of the `Pet` schema. A red box highlights the `id` field definition:

```
id* integer(int32)  
  title: ID  
  minimum: 0  
  example: 1  
  readOnly: true  
  The ID of the pet.
```

Red arrows indicate the flow from the selected schema in the Structure pane to the corresponding YAML and Swagger views.

Question

Do we have all needed data?  
Can we use it to make testing easier?

You are here



**01** Problems to solve

---

**02** Open API / Swagger

---

**03** Generate clients and servers

---

**04** Demo – client generate and test

---

**05** Integration scenarios

---

**06** GitHub & QA

---

# Open API Generator



## Client Generators (66 technologies!)

- CLI version
- Maven Plugin
- and others

<https://openapi-generator.tech/docs/generators>

```
> $$("#server-generators + ul a").map(g => g.innerText).length  
< 57  
> $$("#client-generators + ul a").map(g => g.innerText).length  
< 66
```

## Server Generators (57 technologies!)

We are not going to cover it today

You can generate not only client but also API itself

### Use Case:

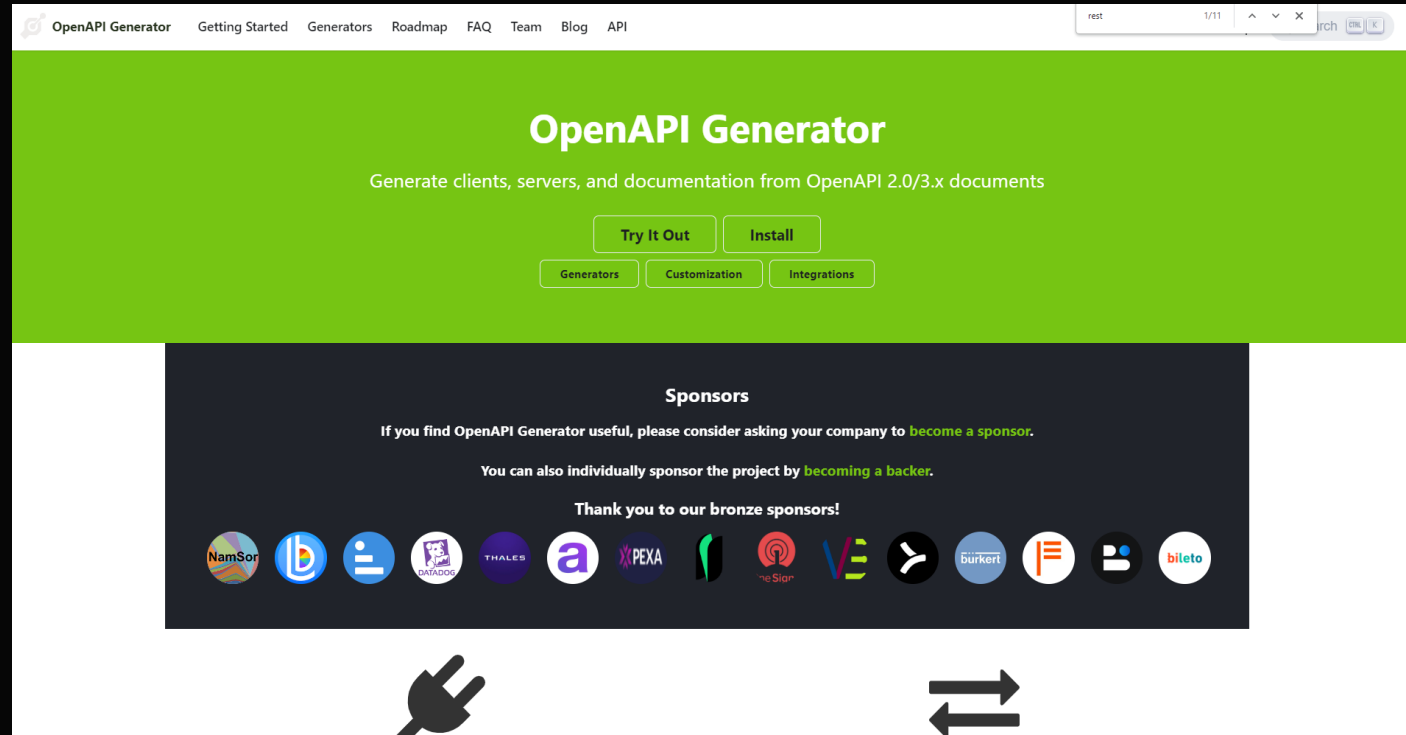
- You need a mocked version of one of your APIs in QA env

### Solution

- Generate server from Open API spec
- Put dummy / mocked data as response
- Build & Deploy it 😊



## Open API Generator – no screens, just explore web page!



<https://openapi-generator.tech/>

You are here

>>>

**04** Demo – client generate and test

**01** Problems to solve

---

**02** Open API / Swagger

---

**03** Generate clients and servers

---

**05** Integration scenarios

---

**06** GitHub & QA

---

Code time 😊



01 Problems to solve

---

02 Open API / Swagger

---

03 Generate clients and servers

---

04 Demo – client generate and test

---

You are here >>> 05 Integration scenarios

---

06 GitHub & QA

---

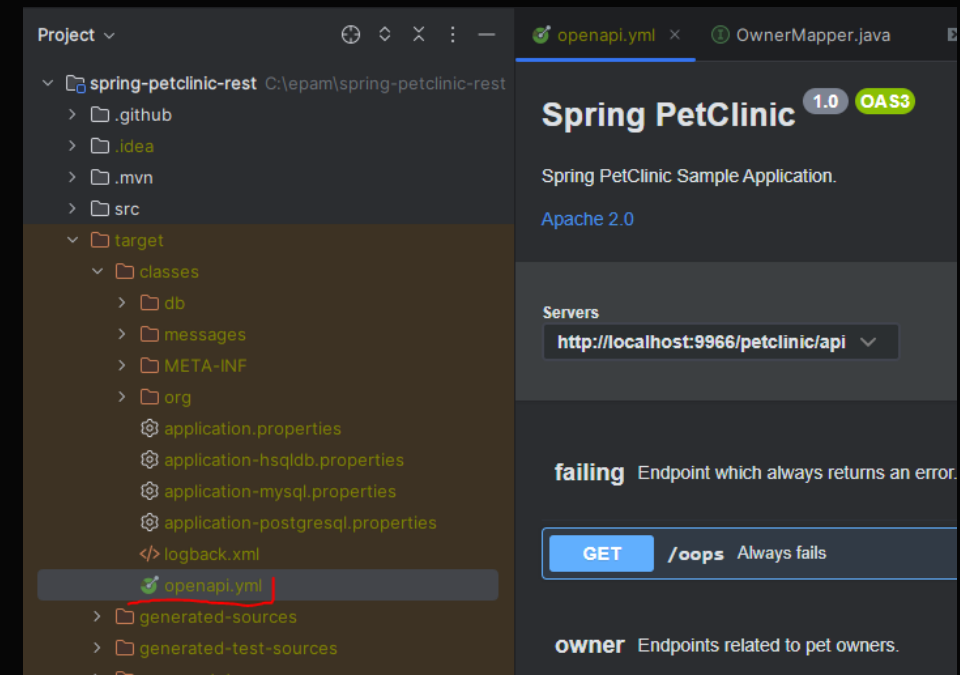
# No Open API Specification

## Dev

- No action

## QA

- Take code of API
- Add documentation or generate
- Build client
- Implement tests
- Run/Report



```
const swaggerAutogen = require('swagger-autogen')()

const outputFile = './swagger_output.json'
const endpointsFiles = ['./routers/personRouter.js']

swaggerAutogen(outputFile, endpointsFiles)
```

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-ui</artifactId>
  <version>1.7.0</version>
</dependency>
```

# One timer – (generated project)

## Dev

- Deploy Swagger
- Publish OAS on SwaggerHub



## Cons

- Boiler plate code in repository
- Have to be careful with configuration
- Hard to maintain on frequent changes

## QA

- Take Open Api Specification
  - Or take ready client code from Swagger Hub
- Generate client(s)
- Implement Tests
- Run/Report

## Pros

- Super quick to generate
- Test stubs are ready to use and run
- Ready to run on CI



# Popular – dependency to client on TAF

## Dev

- Deploy Swagger
- Publish OAS on SwaggerHub



## Cons

- After change in API commit is needed (spec) – if used as file
- Changes in spec are not tracked in repository – if used as URL
- Tests are run against currently deployed version

## QA

- Take Open API Specification (or URL)
- Generate client(s) – **as separate project or module**  
You can use Maven Open API Generator plugin
- **Tests are implemented as part of TAF**
- **TAF just holds dependency to client**
- Run/Report

## Pros

- Super quick to generate
- Test stubs are ready to use and run
- Ready to run on CI





# Automated and Integrated

## Dev

- Deploy OAS (yaml or json) to artifact repository as part of CI
- Deploy Swagger
- Publish OAS on SwaggerHub



## QA

- Take any version of Open Api Specification
- Generate client(s) – during each build
  - You can use Maven Open API Generator plugin (put URL in pom.xml)
- Tests are implemented as part of TAF
- TAF just holds dependency to client
- Run/Report



## Cons

- ?

## Pros

- No boilerplate code in repository
- No commit on spec change (spec is downloaded from Swagger during build)
- Super quick to generate
- Test stubs are ready to use and run
- Ready to run on CI

**01** Problems to solve

---

**02** Open API / Swagger

---

**03** Generate clients and servers

---

**04** Demo – client generate and test

---

**05** Integration scenarios

---

**You are here**



**06** GitHub & QA

---

## FAQ (and problems)

### We do not have Swagger.

Or our swagger is old, deprecated or not reflect current state.

- With PO/PM/SM put testability requirement into backlog to make it usable and deployable
- Do not forget that it is also project documentation, so you improve documentation and testability
- As mentioned, – take ownership and generate spec on your own

### We already have a test suites for API

- Like always - small steps
- Like every software also TAF's and test suites getting deprecated or obsolete
- They produce or are tech debt
- Or with growing complexity of project are not valid / hard to maintain.
- Nothing should be set in stone
- Continuous refactoring

### What in case of changes in API

- Nonbreaking changes
  - Regression still works
  - You must cover new code with test cases
- Breaking changes, like:
  - Removed endpoint
  - Change in model (i.e. removed field)
  - Your client will be generated and compiled
  - BUT tests fail on compilation = you must adjust to new state of API!

## FAQ (and problems)

### What about versioning?

- When you have OAS in your TAF repository it's just versioned with your code
- When you use Nexus or Artifactory you can run against any version with combination of
  - Git tag or commit
  - Specified version of artifact

### What about negative tests?

- Seems to be tricky?
  - When generate client enums are fixed, you cannot use not existing entry!

#### **BUT**

- You can still serialize/deserialize and change payload
- You can use RestAssured filters to modify payload
- You can use API client and connectivity from Open API and work on raw payloads using i.e. JsonPaths
- It can be mixed!

# Me and my 50 microservices

	C1	C2	C3	C4
1	group	name	specUrl	schemas
2	pet	pet_service	http://{{env}}.docs.example.com/swagger.json	http://{{env}}.docs.example.com/schema.json
3	pet	pet_owners	http://{{env}}.docs.example.com/swagger.json	http://{{env}}.docs.example.com/schema.json
4	pet	pet_types	http://{{env}}.docs.example.com/swagger.json	http://{{env}}.docs.example.com/schema.json
5	pet	pet_visits	http://{{env}}.docs.example.com/swagger.json	http://{{env}}.docs.example.com/schema.json
6	pet	pet_service	http://{{env}}.docs.example.com/swagger.json	http://{{env}}.docs.example.com/schema.json
7	pet	pet_service	http://{{env}}.docs.example.com/swagger.json	http://{{env}}.docs.example.com/schema.json

## Problem

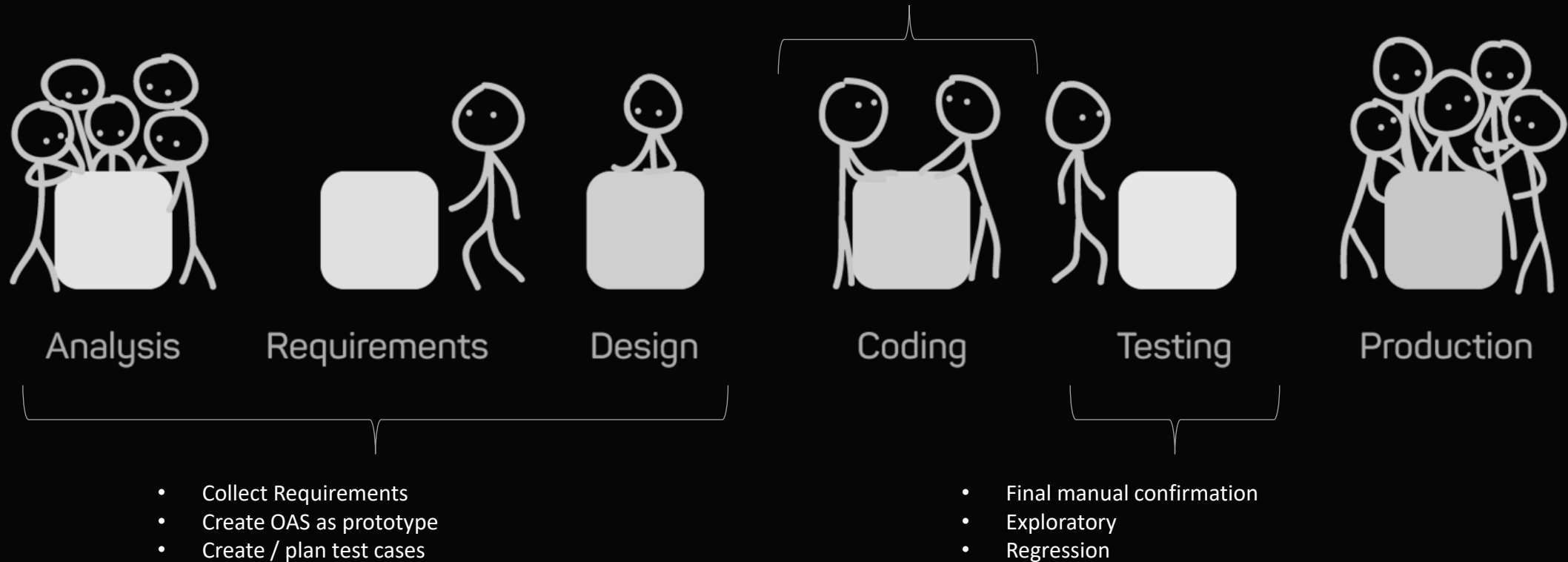
- I have 50 microservices in my project to test
- Each service has own Swagger instance and specification
- I have to handle 50 specs and

## Solution

- Build tools have specified structure – **use that fact!**
- Collect all of them to CSV file (as example)
- Before building TAF – generate structure of you solution
- Using the Move you need
  - For each service directory with
    - pom.xml (with open-api-generator-maven-plugin)
    - Open Api specs (json or yaml)
    - .gitignore (to ignore all generated code)
- Create small python script which
  - Create project structure
  - Add modules to <modules> section in parent pom.xml
- Then build all the clients – Maven Reactor will handle it!

## BDD/ATDD/TDD & shifted left friends with OAS

- OAS prototype is created
- Automation goes first as TDD
- Or automation in parallel with development
- Devs can run new test cases with new code



THANK YOU!

## Finally – Q and A



[https://github.com/majran/tech\\_talks\\_sample\\_petstore](https://github.com/majran/tech_talks_sample_petstore)



Now we are asking 😊

**Name two „core” elements of Open API Specifiction structure?**

