

Генерация сетевого слоя на основе OpenAPI (Swagger) спецификации

Теория + практика

Антон Давыдов @dydus0x14
Swoo & Кошелёк

Дисклеймер:

При использовании на практике рассмотренных подходов из вашей Core/инфраструктурной команды могут быть уволены разработчики. Используйте на свой страх и риск. Podlodka и спикер ответственности не несут.

Как разговаривают клиент и сервер?

API

- REST (JSON/XML)
- RPC (gRPC/Thrift)
- SOAP
- GraphQL
- ...

**В высокой долей вероятности
у вас REST API с JSON**

Типичные проблемы и сложности на практике при разработке/поддержке API

- Нет документации
- Версионирование
- Много бойлерплейт кода (в сериализации)
- Нужно ковыряться в нюансах работы API

Способы декларации API и документирования

- Из уст в уста
- В базе знаний список эндпойнтов
- Примеры curl, postman запросов
- IDL (RPC)
- OpenAPI/Swagger (REST)

Хорошая документация – это

**Когда не надо изучать
исходники или снифать
траффик, чтобы понять как
работает API**

Что такое OpenAPI и Swagger?



<https://www.openapis.org>

<https://swagger.io>

OpenAPI Specification (OAS)

The OpenAPI Specification allows the description of a remote API accessible through HTTP or HTTP-like protocols.

<https://oai.github.io/Documentation/specification.html>

Пример. Twitter OAS #1

```
{
  "openapi" : "3.0.0",
  "info" : {
    "description" : "API Reference – Labs v2",
    "version" : "2.13",
    "title" : "Tweets and Users",
    "termsOfService" : "https://developer.twitter.com/en/developer-terms/agreement-and-policy.html",
    "contact" : {
      "name" : "Twitter Developers",
      "url" : "https://developer.twitter.com/"
    },
    "license" : {
      "name" : "Twitter Developer Agreement and Policy",
      "url" : "https://developer.twitter.com/en/developer-terms/agreement-and-policy.html"
    }
  },
  "servers" : [ {
    "description" : "Twitter API",
    "url" : "https://api.twitter.com"
  } ],
  "tags" : [ {
    "name" : "General",
    "description" : "Miscellaneous endpoints for general API functionality",
    "externalDocs" : {
      "description" : "Find out more",
      "url" : "https://developer.twitter.com/en/docs/labs"
    }
  }, {
    "name" : "Tweets",
    "description" : "APIs related to Tweets",
    "externalDocs" : {
      "description" : "Find out more",
      "url" : "https://developer.twitter.com/en/docs/labs/tweets-and-users/overview"
    }
  }, {
    "name" : "Users",
    "description" : "APIs related to Users",
    "externalDocs" : {
      "description" : "Find out more",
      "url" : "https://developer.twitter.com/en/docs/labs/tweets-and-users/overview"
    }
  } ],
  "paths" : {
```

<https://api.twitter.com/labs/2/openapi.json>

Пример. Twitter OAS #2

```
"/labs/2/users/{id}" : {
  "get" : {
    "tags" : [ "Users" ],
    "summary" : "Return details for the specified users",
    "description" : "This endpoint returns information about a user. Specify user by ID.",
    "operationId" : "findUserById",
    "parameters" : [ {
      "name" : "id",
      "in" : "path",
      "description" : "Required. A User ID.",
      "required" : true,
      "schema" : {
        "$ref" : "#/components/schemas/UserID"
      }
    }, {
      "$ref" : "#/components/parameters/UserExpansionsParameter"
    }, {
      "$ref" : "#/components/parameters/TweetFieldsParameter"
    }, {
      "$ref" : "#/components/parameters/UserFieldsParameter"
    }, {
      "$ref" : "#/components/parameters/MediaFieldsParameter"
    }, {
      "$ref" : "#/components/parameters/PlaceFieldsParameter"
    }, {
      "$ref" : "#/components/parameters/PollFieldsParameter"
    } ],
    "responses" : {
      "200" : {
        "description" : "The request was successful",
        "content" : {
          "application/json" : {
            "schema" : {
              "$ref" : "#/components/schemas/SingleUserLookupResponse"
            }
          }
        }
      },
      "default" : {
        "$ref" : "#/components/responses/HttpErrorResponse"
      }
    }
  }
},
```

<https://api.twitter.com/labs/2/openapi.json>

OAS

- Описывает REST API с помощью JSON, YAML
- Machine-readable
- Human-readable

OpenAPI и Swagger

OpenAPI refers to the industry-standard specification for RESTful API design.
Swagger refers to a set of SmartBear tools.

<https://nordicapis.com/whats-the-difference-between-swagger-and-openapi/>

Поддержка Swagger со стороны бекенда

- Спецификация на основании кода (Java Annotations)
- Спецификация отдельно от кода (Swagger Editor или руками)

Пример Swagger UI

Swagger

Supported by SMARTBEAR

Select a definition

Fitness users

Application API

0.0.1-SNAPSHOT

OAS3

V3/api-docs/Fitness users

The project to support your fit

Contact Roman

Apache 2.0

Servers

http://localhost:8080 - Dev service

User

The User API

DELETE

/users/{id}

Delete user by current Id in the gym

GET

/users/{id}

Get a user by Id in the gym

GET

/users

Gets all users in the gym

POST

/users

Add new user in the gym

PUT

/users/{id}

Update user by current Id in the gym

GET

/users

Gets all users in the gym

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Found the users	No links

Media type

application/json

Controls Accept header.

Example Value

Schema

```
[
  {
    "userId": 0,
    "fullName": "Roman",
    "email": "test@gmail.com",
    "birthDay": "0030-04-17",
    "passType": "ANNUAL"
  }
]
```

<https://habr.com/ru/post/541592/>

Пример Swagger Editor

Swagger Editor

File Edit Generate Server Generate Client

```
1  swagger: "2.0"
2  info:
3    description: "This is a sample server Petstore server. You can find out more
      about Swagger at [http://swagger.io](http://swagger.io) or on [irc
      .freenode.net, #swagger](http://swagger.io/irc/). For this sample, you
      can use the api key `special-key` to test the authorization filters."
4    version: "1.0.0"
5    title: "Swagger Petstore"
6    termsOfService: "http://swagger.io/terms/"
7    contact:
8      email: "apiteam@swagger.io"
9    license:
10     name: "Apache 2.0"
11     url: "http://www.apache.org/licenses/LICENSE-2.0.html"
12 host: "petstore.swagger.io"
13 basePath: "/v2"
14 tags:
15 - name: "pet"
16   description: "Everything about your Pets"
17   externalDocs:
18     description: "Find out more"
19     url: "http://swagger.io"
20 - name: "store"
21   description: "Access to Petstore orders"
22 - name: "user"
23   description: "Operations about user"
24   externalDocs:
25     description: "Find out more about our store"
26     url: "http://swagger.io"
27 schemes:
28 - "https"
29 - "http"
30 paths:
31   /pet:
32     post:
33       tags:
34       - "pet"
35       summary: "Add a new pet to the store"
36       description: ""
37       operationId: "addPet"
38       consumes:
39       - "application/json"
40       - "application/xml"
41       produces:
```

Swagger Petstore1.0.0

[Base URL: petstore.swagger.io/v2]

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

[Terms of service](#)
[Contact the developer](#)
[Apache 2.0](#)
[Find out more about Swagger](#)

Schemes

HTTPS

Authorize

petEverything about your Pets

Find out more: <http://swagger.io>

POST/petAdd a new pet to the store

PUT/petUpdate an existing pet


GET/pet/findByStatusFinds Pets by status

GET/pet/findByTagsFinds Pets by tags

GET/pet/{petId}Find pet by ID

17

Типичные проблемы и сложности на практике при разработке/поддержке API

-  Нет документации
- Версионирование
- Много бойлерплейт кода (в сериализации)
- Нужно ковыряться в нюансах работы API

Хотелка при создании сетевого слоя

Скрыть факт выполнения кода на удаленной
машине

Кодогенераторы по OAS

Правильный кодогенератор

- Прячет семантическую сложность
- Сгенерированный код не нужно трогать и саппортить
- В целом не обязательно пушить в репозиторий
- Убирает сложность работы с рантайме

Неправильный кодогенератор

- Не прячет "кишки" реализации
- Необходимо вникать в детали работы
- Может генерировать странный/невалидный код

Принцип работы генератора

- Парсинг входного файла во внутреннее представление
- Генерация конечного кода на базе шаблонов

Кодогенерация REST клиента для iOS в 2014

```
{
  "struct": "GoogleLocation",
  "typedef" : {
    "lat" : "double",
    "lng" : "double"
  }
},

{
  "struct": "GoogleGeometry",
  "typedef" : {
    "location" : "GoogleLocation"
  }
}
```

ifacegen

<https://habr.com/ru/company/ifree/blog/213697/>

openapi-generator

	Languages/Frameworks
API clients	ActionScript , Ada , Apex , Bash , C , C# (.net 2.0, 3.5 or later, .NET Standard 1.3 - 2.0, .NET Core 2.0, .NET 5.0. Libraries: RestSharp, HttpClient), C++ (Arduino, cpp-restsdk, Qt5, Tizen, Unreal Engine 4), Clojure , Crystal , Dart , Elixir , Elm , Eiffel , Erlang , Go , Groovy , Haskell (http-client, Servant), Java (Jersey1.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign, RestTemplate, RESTEasy, Vertx, Google API Client Library for Java, Rest-assured, Spring 5 Web Client, MicroProfile Rest Client), k6 , Kotlin , Lua , Nim , Node.js/JavaScript (ES5, ES6, AngularJS with Google Closure Compiler annotations, Flow types, Apollo GraphQL DataStore), Objective-C , OCaml , Perl , PHP , PowerShell , Python , R , Ruby , Rust (hyper, reqwest, rust-server), Scala (akka, http4s, scalaz, sttp, swagger-async-httpclient), Swift (2.x, 3.x, 4.x, 5.x), Typescript (AngularJS, Angular (2.x - 11.x), Aurelia, Axios, Fetch, Inversify, jQuery, Nestjs, Node, redux-query, Rxjs)
Server stubs	Ada , C# (ASP.NET Core, NancyFx), C++ (Pistache, Restbed, Qt5 QHTTPEngine), Erlang , F# (Giraffe), Go (net/http, Gin, Echo), Haskell (Servant), Java (MSF4J, Spring, Undertow, JAX-RS: CDI, CXF, Inflector, Jersey, RestEasy, Play Framework, PKMST , Vert.x), Kotlin (Spring Boot, Ktor, Vertx), PHP (Laravel, Lumen, Mezzio (fka Zend Expressive), Slim, Silex, Symfony), Python (FastAPI, Flask), NodeJS , Ruby (Sinatra, Rails5), Rust (rust-server), Scala (Akka, Finch , Lagom , Play , Scalatra)
API documentation generators	HTML, Confluence Wiki, AsciiDoc, Markdown, PlantUML
Configuration files	Apache2
Others	GraphQL, JMeter, Ktorm, MySQL Schema, Protocol Buffer, WSDL

<https://github.com/OpenAPITools/openapi-generator>

SwagGen

SwagGen

PLATFORMS

LINUX | MACOS

SPM

COMPATIBLE

RELEASE

V4.5.0

LICENSE

MIT

SwagGen is a library and command line tool for parsing and generating code for [OpenAPI/Swagger 3.0](#) specs, completely written in Swift.

Swagger 2 support has been removed. For Swagger 2 use version `3.0.2` or the `swagger_2` branch

Swagger parser

It contains a `Swagger` library that can be used in Swift to load and parse Swagger specs.

Swagger code generator

`SwagGen` is command line tool that generates code from a [OpenAPI/Swagger 3.0](#) spec. Templates for any language can be written that leverage this generator.

It is an alternative the official [swagger-codegen](#) java code generator, and adds some improvements such as speed, configurability, simplicity, extensibility, and an improved templating language.

Swift template






`SwagGen` includes a bundled template for generating a client side Swift library for interfacing with the Swagger spec. It includes support for model inheritance, shared enums, discrete and mutable request objects, inline schemas, Codable and Equatable models, configurable options, generic networking stack, and many other niceties.

<https://github.com/yonaskolb/SwagGen>

Постучим по клавиатуре

Swift генерация по OAS на практике

Решенные проблемы

-  Нет документации
-  Версионирование
-  Много бойлерплейт кода (в сериализации)
-  Нужно ковыряться в нюансах работы API
-  Единый источник правды

Доп возможности

**Генерация мок сервера на
случай, если настоящий
бекенд ещё не готов**

Текущие недостатки кодогенерации по OAS

- Странная архитектура генерируемого кода
- Генерация некомпилируемого кода
- Кодген заточен под один сервис (одну OAS)
- Ошибки в спецификациях


✗ Поддержка нового эндпойнта врукопашную

- Изучить доку (поснифать трафик)
- Задекларировать Codable модельки
- Создать новый метод сетевого слоя
- Решить нерешаемую задачу нейминга
- Написать тесты

✓ Поддержка нового эндпойнта с OAS + кодогенерацией

- Попросить бекендера обновить Swagger
- Запустить код ген
- PROFIT!

Выводы

- Человечество уже решило задачу (де)сериализации JSON
- В перспективе создание и поддержка сетевого слоя - задачка без кодинга
- Кодогенераторы по OAS будут в плачевном состоянии, пока их не станут использовать и исправлять
- Пробуйте! 

Спасибо! 🙏



tg/tw @dydus0x14

tg канал https://t.me/km_engineering