A Beginner's Guide to Code Generation for REST APIs

How to use OpenAPI Generator to streamline RESTful API development

William Cheng
@wing328

Important update

As you probably know, myself and more than 40 top contributors and template creators of Swagger Codegen decided to fork Swagger Codegen in May 2018 to maintain a community-driven version called “OpenAPI Generator” (http://openapi-generator.tech). For more information on why we forked, please refer to the Q&A in the project page:

https://github.com/OpenAPITools/openapi-generator/blob/master/docs/qna.md

To migrate from Swagger Codegen to OpenAPI Generator, please refer to the migration guide:

https://github.com/OpenAPITools/openapi-generator/blob/master/docs/migration-from-swagger-codegen.md

Since the fork, we’ve released more than 40 stable versions and we just released v5.0.0 - the second major version since the fork, in Dec 2020. Please continue to support OpenAPI Generator so that we can bring the project to the next level.

(Swagger is a registered trademark of SmartBear Software, Inc)

The background image in the book cover is designed by Harryarts / Freepik (http://www.freepik.com)

About the author

William Cheng is an experienced IT professional with 10+ years of experience in IT startups, academic research, a leading semiconductor equipment manufacturer and a top-tier global investment bank. He started contributing to OpenAPI Generator in 2014 and has been leading the project on a voluntary basis with more than 1800 merged pull requests and over 3.5 million line changes. Since the inception in 2011, the project has grown to one of the most active open-source projects with more than 1500 contributors and many companies, from start-ups to IT conglomerates, are using it in their technology stacks for production usage.

If you have any questions or comments on this book, please email me at wing328hk@gmail.com.

Thank you again for purchasing a copy of this book to support my work on OpenAPI Generator.

1. Introduction

OpenAPI Generator has been gaining significant popularity in recent years. Companies, big or small, are using it in production to streamline the software development process and to reduce maintenance cost. Here are some companies using OpenAPI Generator in production:

- Adaptant Solutions AG
- Angular.Schule
- ASKUL
- b<>com
- BIMData.io
- Bithost GmbH
- Boxever
- Camptocamp
- codecentric AG
- Cupix
- FormAPI
- GenFlow
- GMO Pepabo
- GoDaddy
- JustStar
- Klarna
- Metaswitch
- Myworkout
- Prometheus/Alertmanager
- Raiffeisen Schweiz Genossenschaft
- RepreZen API Studio
- REST United
- Stingray
- Suva
- Telstra
- TUI InfoTec GmbH
- unblu inc.
- Veamly
- Xero
- Zalando

The OpenAPI Generator community is also growing bigger and bigger every day with 15000+ commits and 4000+ merged pull requests, 1700+ contributors who have at least 1 merged pull request.

This book aims to show IT professionals how they can leverage code generation with OpenAPI Generator to generate RESTful API clients, server stubs and documentation within minutes, and customize the output with various options to meet their unique requirements.

1.1 What is “OpenAPI Generator”?

OpenAPI Generator is a completely free and open-source (Apache License v2) project to generate REST[1] API clients, server stubs and documentation based on an OpenAPI specification (formerly known as Swagger specification) . In case you’re not familiar with OpenAPI specification, it’s the most popular standard in terms of describing RESTful API and is backed by many well-known companies such as Adobe, Atlassian, CA Technologies, eBay, IBM, Google, Microsoft, PayPal, Salesforce, SAP, SmartBear and more[2]. For the full specification of OpenAPI 2.0 and 3.0, please refer to the project page in Github[3].

1.2 Why “OpenAPI Generator”?

OpenAPI Generator aims to benefit IT professionals with tremendous savings in terms of time and manpower. The conventional approach is to write and maintain the API clients and documentation manually, which is not a bad approach if the API only has a few endpoints and the requirement is to provide Software Development Kit (SDK) in one or two programming languages only. As the API grows bigger and bigger, the manual approach simply would not scale to meet the changing requirements and this is exactly how OpenAPI Generator comes into play. OpenAPI Generator is using the contract-first approach that API owners only need to update the specification with new or modified endpoints and then generate the SDKs in various programming languages and API documentation within minutes.

There is no doubt that OpenAPI Generator is not the only code generator available on the market so why choose OpenAPI Generator? Here are a couple of points to convince you OpenAPI Generator is the obvious choice:

1. OpenAPI Generator is completely free and open-source with Apache License version 2.0. The auto-generated files are not licensed so you can apply whatever software license to the auto-generated code as you deem appropriate.

2. OpenAPI Generator is used by many companies in production, from listed companies such as Cisco, Datadog, IBM and more to startups that have raised rounds of venture capital fundings including Boxever, Xero just to name a few, which means the project itself is mature and production ready.

3. OpenAPI Generator, with 7000+ stars on the Github project page and 15000+ commits, has a vibrant and growing community. More than 1700 software developers all over the world have filed a Pull Request, which is the standard way to propose code changes in git, to make OpenAPI Generator better. You will find many activities related to OpenAPI Generator in StackOverflow, Twitter, Github and other social

---

1 https://en.wikipedia.org/wiki/Representational_state_transfer

2 OpenAPI members: https://www.openapis.org/membership/members

3 https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

networks. For a list of presentations, videos, tutorials and eBooks on OpenAPI Generator, please refer to the README in the Github repository[4].

4. OpenAPI Generator supports more than 30 API clients, 30 server stubs and API documentation. We have also added schema converters for protocol buffers, GraphQL, JMeter and more.

1.3 Who needs “OpenAPI Generator”?

OpenAPI Generator is a huge time saver and many IT professionals can benefit from it. We will walk through a few examples to show how OpenAPI Generator changes the way people work.

1. Backend developers - OpenAPI Generator comes with 25+ server stub generators for different server-side frameworks such as PHP Symfony, C# Nancy, Java Spring, Python Flask and more. The auto-generated server-side code allows back-end developers to easily implement a RESTful backend given an OpenAPI/Swagger 2.0 or OpenAPI 3.0 specification file.

2. Frontend developers - OpenAPI Generator is a huge time saver for front-end developers who need access to the RESTful backend. Instead of manually writing a thin wrapper or SDK for the RESTful backend, frontend developers can easily generate full-featured SDKs in their favorite languages like TypeScript and JavaScript within a minute. Currently, the TypeScript client generators support AngularJS, Angular 2.x, 4.x, Fetch and more to meet different requirements.

3. Technical writer - Keeping API documentation up-to-date could be hard but with OpenAPI Generator, one can effortlessly generate up-to-date API documentation with the latest API specification. Technical writers can also customize the layout, look and feel of the API documentation, or add a new section by customizing the documentation templates.

4. QA engineer - The auto-generated API clients also come with test files that a QA engineer can simply fill out the details or easily create new test cases to cover more scenarios.

5. API evangelist - The success of an API evangelist hinges on many factors and one of them is how many developers actually use the API in production. Imagine your startup just launches an API for weather forecast and the API evangelist will need to engage as many developers as possible to consume the API. Using OpenAPI Generator, one can easily generate full-featured SDKs in more than 10 programming languages to reduce the friction for onboarding new developers.

[4] https://github.com/OpenAPITools/openapi-generator#5---presentationsvideostutorialsbooks

1.4 Who should read this book?

IT professionals who have never used OpenAPI Generator will definitely find this book very helpful in understanding OpenAPI Generator and how to leverage OpenAPI Generator to streamline API development in different platforms: Mac, Linux, Windows. Those who have tried OpenAPI Generator before can also find tips and hacks to fine tune the output of OpenAPI Generator to accommodate different use cases.

In this book, we will walk through examples of using OpenAPI Generator in different scenarios so those who want to explore other use cases of OpenAPI Generator will find this book very informative as well.

2. OpenAPI 2.0 Specification

In this chapter, we will walk you through the basics of OpenAPI 2.0 specification as it would be difficult to explain OpenAPI Generator output without understanding the specification. There are many tools on the market that fully support OpenAPI 2.0 specification. ReadyAPI, Postman, Runscope just to name a few. As the specification becomes the common language for these API tools, understanding it will help you navigate the API ecosystem with ease.

The best way to learn OpenAPI 2.0 specification is by examples. In this book, we will mainly use the Pet Store API specification (http://pubi.org/petstore.json), which describes a RESTful API for an online pet store with some operations and several models. Both JSON and YAML can be used to represent the specification[5]. It is important to note that all field names are case-sensitive.

2.1 Basic

The full Pet Store API specification in YAML can be found in the Github page of this eBook[6]. Let's start with the basic information of the REST API:

```
swagger: '2.0'
info:
  description: 'This is a sample server Petstore server ...'
  version: 1.0.0
  title: Swagger Petstore
  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'
host: api.pubi.org
basePath: /v2
schemes:
  - http
```

"**swagger**" states the version of the specification, which is 2.0 for our examples. OpenAPI specification 3.0 is the successor to OpenAPI/Swagger specification 2.0[7].

"**info**" provides various information about the API such as version, contact, license and more. Only "**title**' and "**version**" are required fields and you can find out more from the link below:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#info-object

5 https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#format
6 https://github.com/openapi-generator-ebook/introduction/blob/master/petstore.yaml
7 https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md

As the name implies, “**host**” is the hostname of the API. One can use an IP address as well, for example, 192.168.7.5:8080. “basePath” is where the API is served. It’s the common URL shared by all the endpoints and it can be omitted if the endpoints do not share a common URL. Consider the following endpoint URL:

Example: https://api.codegen.org:8080/v3/api/generator

The “**host**” is *api.codegen.org:8080* and the “**basePath**” is */v3/api* assuming */v3/api* is common across all the endpoints.

“**schemes**” is the transfer protocol of the API and must be one of the following values: *http, https, ws, wss*

For more information on these top-level fields, please refer to the “Schema” section in the official OpenAPI specification 2.0:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#schema

2.2 Tags

The next section in the specification is “**tags**”[8], which can be used for classifying the endpoints. Here is an example:

```
tags:
  - name: pet
    description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: 'http://swagger.io'
```

Then we can tag all the operations related to Pet with the “pet” (case-sensitive) tag.

2.3 Security definitions

“**securityDefinitions**” describes what authentication mechanism is needed for the endpoint(s). Currently, three mechanisms are supported: API key in header or query, HTTP basic and OAuth2 token. One endpoint can have more than one authentication mechanism. Here is an example:

```
securityDefinitions:
  petstore_auth:
    type: oauth2
    authorizationUrl: 'http://api.pubi.org/api/oauth/dialog'
    flow: implicit
    scopes:
      'write:pets': modify pets in your account
```

[8] https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#tagObject

```
      'read:pets': read your pets
  api_key:
    type: apiKey
    name: api_key
    in: header
```

For details on the security definition, please refer to the “Security Schema Object” section in the official OpenAPI 2.0 specification:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#securitySchemeObject

Later we will illustrate how to selectively apply these security definitions to the endpoints.

2.4 Object

Modelling a JSON payload in OpenAPI 2.0 specification is pretty straightforward. Consider the following HTTP response in JSON:

```
{
  “id”: 153,
  “petId”: 38,
  “quantity”: 2,
  “shipDate”: “2017-05-11T05:12:30+08:00”,
  “status”: “placed”,
  “Complete”: true
}
```

And here is how we model it in OpenAPI specification 2.0:

```
definitions:
  Order:
    title: Pet Order
    description: An order for a pets from the pet store
    type: object
    required:
      - petId
      - quantity
    properties:
      id:
        type: integer
        format: int64
      petId:
        type: integer
        format: int64
      quantity:
        type: integer
        format: int32
      shipDate:
        type: string
        format: date-time
```

```
    status:
      type: string
      description: Order Status
      enum:
        - placed
        - approved
        - delivered
    complete:
      type: boolean
      default: false
```

“**required**” lists out properties that must be present in the object. In this example, only “petId” and “quantity” are required. “**type**” describes the property type and “**format**” is an optional field to further define the type. We recommend including “**format**” to describe the type if possible. For a list of supported “type” and “format”, please refer to the “Data Types” section in the OpenAPI specification 2.0:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#data-types

If the field only allows certain values, we can use the “**enum**” attribute to list out all possible values. If the ***optional*** field has a default value, we can put the value in the “**default**” attribute.

2.5 Operation

Finally, we will show you how an endpoint is defined. Let’s start with the following example:

```
paths:
  '/pet/{petId}':
    post:
      tags:
        - pet
      summary: Updates a pet in the store with form data
      description: Update the name or status of a pet
      operationId: updatePetWithForm
      consumes:
        - application/x-www-form-urlencoded
      produces:
        - application/xml
        - application/json
      parameters:
        - name: petId
          in: path
          description: ID of pet that needs to be updated
          required: true
          type: integer
          format: int64
        - name: name
          in: formData
          description: Updated name of the pet
          required: false
```

```
      type: string
    - name: status
      in: formData
      description: Updated status of the pet
      required: false
      type: string
  responses:
    '405':
      description: Invalid input
  security:
    - petstore_auth:
        - 'write:pets'
        - 'read:pets'
```

The endpoint definition starts with the relative path (***/pet/{petId}*** in our case) to the endpoint's full URL. ***{petId}*** denotes a path variable, which is also known as path templating to allow variables in the path.

Under the same relative path, we can define different HTTP operations (or verbs) such as POST, GET, PUT, DELETE, OPTIONS, HEAD, PATCH. Then we use ***"tags"*** to classify this endpoint into the "pet" category. ***"operationId"*** can be used to uniquely identify this endpoint in the specification. It is optional but we strongly recommend defining one as OpenAPI Generator will use it to determine the method name and automatically generate one if it is not defined.

"consumes", which is optional, defines the MIME types expected by the API server while ***"produces"*** lists out the MIME types of the HTTP response returned by the API server. For a list of supported MIME types, please refer to the "Mime Types" sub-section in the official specification:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#mimeTypes

The "**parameters**" field lists all the parameters supported by this endpoint. "**in**" specifies the location of the parameter, which can be ***path, query, header, formData or body***.

The "path" parameter can only be required and it only supports primitive types such as string and integer but not object as there's no way to substitute the path variable with an object.

The "query" parameter is the URL query string appended to the URL. For the following relative path:

```
/pet/108?format=json&fields=all
```

"format" and "fields" are optional "query" parameters.

The "header" parameter defines the key-value pair in the header of the HTTP request. If the HTTP request contains "X-API-VERSION: v3.1.5" in the header, then the header parameter is "X-API-VERSION".

The “formData” parameter is usually found in the HTTP body for transmitting data to the API server with the “content-type” set to “application/x-www-form-urlencoded”. Here is an example:

```
POST /pet/108 HTTP/1.1
Host: api.pubi.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

name=Lion&status=sold
```

The form data contains 2 parameters: “name”, “status”, which are defined as “string”. Form parameters also support file upload and here is an example:

```
        - name: file
          in: formData
          description: file to upload
          required: false
          type: file
```

The last type of parameter is the “body” parameter. As the name implies, it’s used to model the payload in the HTTP request body. For the following HTTP request:

```
POST /pet/109 HTTP/1.1
Host: api.pubi.org
Content-Type: application/json
Transfer-Encoding:chunked

{
  "id": 109,
  "category": {
    "id": 2,
    "name": "animal"
  },
  "name": "Tiger",
  "photoUrls": [
    "bit.ly/2mzaoTL"
  ],
  "tags": [
    {
      "id": 3,
      "name": "big"
    }
  ],
  "status": "available"
}
```

We will model the HTTP request as follows:

```
    parameters:
```

```
    - in: body
      name: body
      description: Pet object that needs to be added
      required: true
      schema:
        $ref: '#/definitions/Pet'
```

While ‘#/definitions/Pet' in the is defined as

```
Pet:
  title: a Pet
  description: A pet for sale in the pet store
  type: object
  required:
    - name
    - photoUrls
  properties:
    id:
      type: integer
      format: int64
    category:
      $ref: '#/definitions/Category'
    name:
      type: string
      example: doggie
    photoUrls:
      type: array
      items:
        type: string
    tags:
      type: array
      items:
        $ref: '#/definitions/Tag'
    status:
      type: string
      description: pet status in the store
      enum:
        - available
        - pending
```

Tips: the “#” in “#/definitions/Pet” refers to the current document.

“body” parameter is the only type of parameters that support objects while “path”, “header”, “formData” and “query” parameters only support primitive type such as string, integer as defined in https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#data-types.

The “**response**” section defines the HTTP response from the server. In our example, the HTTP response body does not contain any data. To describe a JSON response, we can use the following:

```
    responses:
```

```
        '200':
          description: successful operation
          schema:
            $ref: '#/definitions/Pet'
```

Where “Pet” is defined as an object mentioned above. The raw HTTP response may look like the following:

```
HTTP/1.1 200 OK
Date: Mon, 11 Sep 2017 08:21:34 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Mon, 11 Sep 2017 07:11:20 GMT
Content-Length: 88
Content-Type: application/json
Connection: Closed

{
  "id": 109,
  "category": {
    "id": 2,
    "name": "animal"
  },
  "name": "Tiger",
  "photoUrls": [
    "bit.ly/2mzaoTL"
  ],
  "tags": [
    {
      "id": 3,
      "name": "big"
    }
  ],
  "status": "available"
}
```

To model an array of model (JSON or XML) in the response, we can use the following:

```
       responses:
        '200':
          description: successful operation
          schema:
            type: array
            items:
              $ref: '#/definitions/Pet'
```

What if the HTTP response is the following in which the JSON keys are not fixed?

```
{
  "cat": 2,
  "dog": 3,
  "rabbit": 5
}
```

We can use "**additionalProperties**" to describe a map or dictionary. Here is an example to describe the JSON payload above.

```
responses:
  '200':
    description: successful operation
    schema:
      type: object
      additionalProperties:
        type: integer
        format: int32
```

The last part of the endpoint definition is "**security**", which is optional. This section lists out security definitions(s) required for this endpoint. The following shows the endpoint only requires "petstore_auth" defined in the security definition section.

```
security:
  - petstore_auth:
      - 'write:pets'
      - 'read:pets'
```

If the endpoint does not require any authentication, you can simply omit the "**security**" section in the endpoint definition.

2.6 Editor

Instead of writing the specification file with a text pad or other YAML editor, you may want to try the following open-source tools to edit the specification file:

- **Swagger Editor** (https://editor.swagger.io/) is probably the easiest way to edit OpenAPI specification. All you need is a browser and there's no need to sign up for an account. It has been rewritten from the ground up. (If you prefer the old version of the editor, please go to https://editor2.swagger.io). It is completely free and open-source[9], which means you can run it locally in your machine.

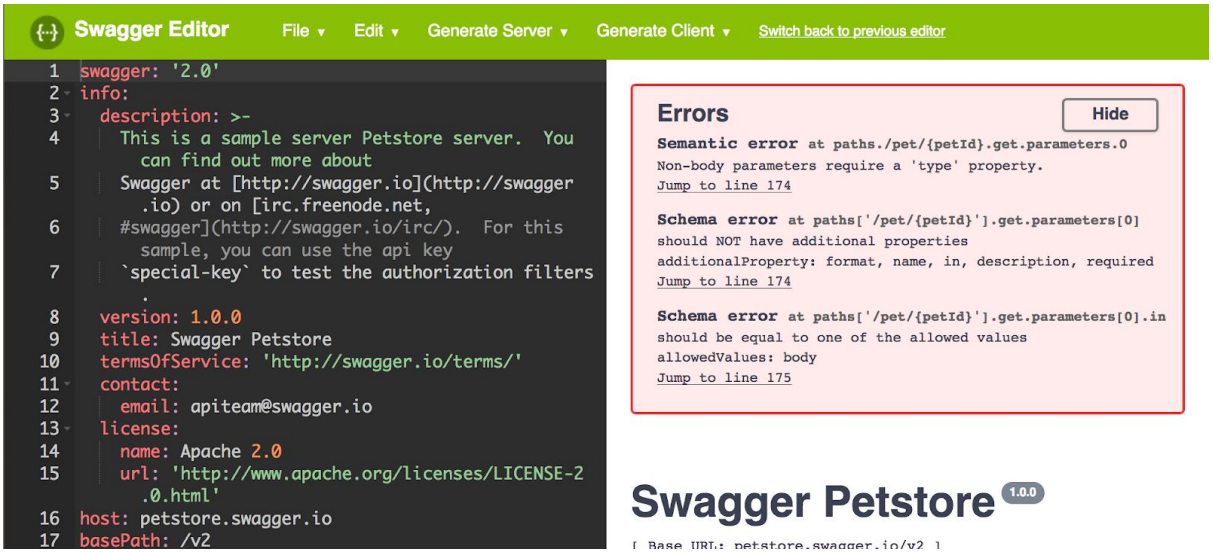
- **Apicurio Studio** (http://www.apicur.io/) by RedHat is a new contender to the OpenAPI editor space. Being free and open-source[10], it is still in beta release and for a list of upcoming features, please refer to the roadmap[11].

- If you use Eclipse for software development, you probably want to try **KaiZen OpenAPI Editor**[12] (also free, open-source), which can be easily installed from the

9 https://github.com/swagger-api/swagger-editor
10 https://github.com/apicurio/apicurio-studio
11 http://www.apicur.io/roadmap/
12 https://github.com/RepreZen/KaiZen-OpenAPI-Editor

Eclipse marketplace[13]. It is developed by RepreZen (https://www.reprezen.com/), which offers RepreZen Studio - an IDE focusing on API design.

The benefit of using these tools is that it will show you the errors or warnings in case it spots issues with the specification. The error message below shows Swagger Editor reporting some errors for an invalid specification[14]:



2.7 Converter

Before OpenAPI specification emerged as the industry standard, there were different formats to describe RESTful APIs. RAML[15], API Blueprint[16] just to name a few. To convert various formats into OpenAPI specification, here are some free, open-source tools to do the job for you:

- **API Spec Transformer** (https://github.com/stoplightio/api-spec-converter) by **Stoplight**[17], which is a SaaS provider focusing on building and testing APIs, supports transformation between various formats: OpenAPI 2.0, RAML (0.8, 1.0), Postman Collection 1.0[18].

- **API Spec Converter** by **LucyBot**[19] (a startup providing solutions to make it easy for developers to get up and running with APIs) is another excellent converter, which is available online via https://lucybot-inc.github.io/api-spec-converter/ and source code can be found in https://github.com/LucyBot-Inc/api-spec-converter. Currently, it

13 https://marketplace.eclipse.org/content/kaizen-openapi-editor
14 https://github.com/openapi-generator-ebook/introduction/blob/master/invalid_petstore.json
15 https://raml.org/
16 https://apiblueprint.org/
17 https://stoplight.io/
18 https://github.com/stoplightio/api-spec-converter#supported-conversions
19 http://lucybot.com/

supports 7 different formats: Swagger 1.x, OpenAPI 2.0 & 3.0, RAML 0.8, IO Docs[20], API Blueprint, Google API Discovery[21], WADL[22].

- **API-Flow** (https://github.com/luckymarmot/API-Flow) is another free and open-source alternative offered by **Paw**[23], which is an advanced API tool for Mac. As of Sep 2017, it supports OpenAPI 2.0, RAML v1.0, Postman Collection v2.0, Paw v3.1 and plans to support OpenAPI 3.0, RAML 0.8, Postman Collection v1.0, Postman Dump v1.0, Insomnia v3.0 and API Blueprint in the future.

2.8 Plugins

Instead of using the contract-first approach to write the specification manually, there are plugins that can generate OpenAPI/Swagger specification from the API server's source code so that the source code is always in sync with the specification. Here are a few examples:

- **Swashbuckle** (https://github.com/domaindrivendev/Swashbuckle): Seamlessly adds a Swagger/OpenAPI to C# WebApi projects.

- **swagger-inline** (https://github.com/readmeio/swagger-inline) by ReadMe[24]: Writes your OpenAPI file as comments.

- **transmute-core** (http://transmute-core.readthedocs.io/en/latest/): Converts Python functions into APIs that include OpenAPI/Swagger 2.0 schema validation and documentation.

- **ruby-grape** (https://github.com/ruby-grape/grape-swagger): Allows you to generate OpenAPI/Swagger 2.0 specifications from Grape APIs.

- **swagger-jsdoc** (https://www.npmjs.com/package/swagger-jsdoc): Keeps an up-to-date and reusable OpenAPI/Swagger 2.0 specification through documentation in the code.

20 https://github.com/mashery/iodocs
21 https://developers.google.com/discovery/v1/reference/apis
22 http://www.w3.org/Submission/wadl/
23 https://paw.cloud/
24 http://readme.io/

3. OpenAPI 3.0 Specification

In this chapter, we will walk you through the basics of OpenAPI 3.0 specification, which was officially released by OpenAPI Initiatives in July 2017. As of 2021, OpenAPI 3.0 specification has become a lot more popular as more tools provide better support. OpenAPI Generator supports both OpenAPI 2.0 and 3.0 specifications. In the future, we may drop support for OpenAPI 2.0 specification as the majority of the ecosystem has migrated to OpenAPI 3.0 specification.

We will start with Pet Store API specification 3.0, which describes a RESTful API for an online pet store with some operations and several models: https://pubspec.github.io/petstore_oas3.yaml. Both JSON and YAML can be used to represent the specification[25]. It is important to note that all field names are case-sensitive.

3.1 Basic

The full Pet Store API specification in JSON and YAML can be found in the Github page of this eBook[26]. Let's start with the basic information of the REST API:

```
openapi: 3.0.0
servers:
  - url: 'http://api.pubi.org/v2'
info:
  description: >-
    This is a sample server Petstore server. For this sample, you can use the api
key
    `special-key` to test the authorization filters.
  version: 1.0.0
  title: OpenAPI Petstore
  license:
    name: Apache-2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
```

"**openapi**" states the version of the specification, which is 3.0.0 for our examples.

"**info**" provides various information about the API such as version, contact, license and more. Only "**title**' and "**version**" are required fields and you can find out more from the link below:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md#infoObject

As the name implies, "**servers**" is a list of server URLs of the API. One can use an IP address as well, for example, http://192.168.7.5:8080/v2. Consider the following endpoint URL:

25 https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md#format
26 https://github.com/openapi-generator-ebook/introduction

Example: https://api.codegen.org:8080/v3/api/generator

The "**url**" is *https://api.codegen.org:8080/v3*.

For more information on these top-level fields, please refer to the "OpenAPI Object" section in the official OpenAPI specification 3.0:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md#openapi-object

3.2 Tags

The next section in the specification is "**tags**"[27], which can be used for classifying the endpoints. Here is an example:

```
tags:
  - name: pet
    description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: 'http://swagger.io'
```

Then we can tag all the operations related to Pet with the "pet" (case-sensitive) tag.

3.3 Security Requirement Object

"**securitySchemes**" describes what authentication mechanism is needed for the endpoint(s). Currently, 4 mechanisms are supported: API key in header or query, HTTP basic, OAuth2 token and OpenID Connect which is newly introduced in OpenAPI 3.0 specification. One endpoint can have more than one authentication mechanism. Here is an example:

```
securityDefinitions:
  petstore_auth:
    type: oauth2
    authorizationUrl: 'http://api.pubi.org/api/oauth/dialog'
    flow: implicit
    scopes:
      'write:pets': modify pets in your account
      'read:pets': read your pets
  api_key:
    type: apiKey
    name: api_key
    in: header
```

For details on the security definition, please refer to the "Security Schema Object" section in the official OpenAPI 3.0 specification:

[27] https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md#tagObject

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md#security-scheme-object

Later we will illustrate how to selectively apply these security definitions to the endpoints.

3.4 Object

Modelling a JSON payload in OpenAPI 3.0 specification is pretty straightforward. Consider the following HTTP response in JSON:

```
{
  “id”: 153,
  “petId”: 38,
  “quantity”: 2,
  “shipDate”: “2017-05-11T05:12:30+08:00”,
  “status”: “placed”,
  “Complete”: true
}
```

And here is how we model it in OpenAPI specification 3.0:

```
  schemas:
    Order:
      title: Pet Order
      description: An order for a pets from the pet store
      type: object
       required:
         - petId
         - quantity
      properties:
        id:
          type: integer
          format: int64
        petId:
          type: integer
          format: int64
        quantity:
          type: integer
          format: int32
        shipDate:
          type: string
          format: date-time
        status:
          type: string
          description: Order Status
          enum:
            - placed
            - approved
            - delivered
        complete:
          type: boolean
          default: false
```

```
    xml:
      name: Order
```

“**required**” lists out properties that must be present in the object. In this example, only “petId” and “quantity” are required. “**type**” describes the property type and “**format**” is an optional field to further define the type. We recommend including “**format**” to describe the type if possible. For a list of supported “type” and “format”, please refer to the “Data Types” section in the OpenAPI specification 3.0:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md#dataTypes

If the field only allows certain values, we can use the “**enum**” attribute to list out all possible values. If the ***optional*** field has a default value, we can put the value in the “**default**” attribute.

3.5 Operation

Finally, we will show you how an endpoint is defined. Let’s start with the following example:

```
paths:
  '/pet/{petId}':
    post:
      tags:
        - pet
      summary: Updates a pet in the store with form data
      description: ''
      operationId: updatePetWithForm
      parameters:
        - name: petId
          in: path
          description: ID of pet that needs to be updated
          required: true
          schema:
            type: integer
            format: int64
      responses:
        '405':
          description: Invalid input
      security:
        - petstore_auth:
            - 'write:pets'
            - 'read:pets'
      requestBody:
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                name:
                  description: Updated name of the pet
                  type: string
```

```
        status:
          description: Updated status of the pet
          type: string
```

The endpoint definition starts with the relative path (***/pet/{petId}*** in our case) to the endpoint's full URL. ***{petId}*** denotes a path variable, which is also known as path templating to allow variables in the path.

Under the same relative path, we can define different HTTP operations (or verbs) such as POST, GET, PUT, DELETE, OPTIONS, HEAD, PATCH. Then we use ***"tags"*** to classify this endpoint into the "pet" category. ***"operationId"*** can be used to uniquely identify this endpoint in the specification. It is optional but we strongly recommend defining one as OpenAPI Generator will use it to determine the method name and automatically generate one if it is not defined.

"requestBody" describes the body of the HTTP request. In this example, the body contains 2 optional values for "name" and "status" in the form of HTML form, which is denoted by the media type "application/x-www-form-urlencoded". Other supported media types can be found in https://tools.ietf.org/html/rfc7231#appendix-D. Here is what the HTTP request looks like:

```
POST /pet/108 HTTP/1.1
Host: api.pubi.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

name=Lion&status=sold
```

One can also use schema to model the payload such as JSON, XML documents.

For the following HTTP request:

```
POST /pet/109 HTTP/1.1
Host: pubi.org
Content-Type: application/json
Transfer-Encoding:chunked

{
  "id": 109,
  "category": {
    "id": 2,
    "name": "animal"
  },
  "name": "Tiger",
  "photoUrls": [
    "bit.ly/2mzaoTL"
  ],
  "tags": [
    {
```

```
      "id": 3,
      "name": "big"
    }
  ],
  "status": "available"
}
```

We will model the HTTP request as follows:

```
paths:
  /pet:
    post:
      tags:
        - pet
      summary: Add a new pet to the store
      description: ''
      operationId: addPet
      responses:
        '405':
          description: Invalid input
      security:
        - petstore_auth:
            - 'write:pets'
            - 'read:pets'
      requestBody:
        $ref: '#/components/requestBodies/Pet'
```

While ‘#/components/requestBodies/Pet' in the is defined as

```
components:
  requestBodies:
    Pet:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Pet'
        application/xml:
          schema:
            $ref: '#/components/schemas/Pet'
      description: Pet object that needs to be added to the store
      required: true
```

And the full definition of ‘#/components/schemas/Pet' is defined as

```
  schemas:
    Pet:
      title: a Pet
      description: A pet for sale in the pet store
      type: object
      required:
        - name
        - photoUrls
      properties:
```

```
      id:
        type: integer
        format: int64
      category:
        $ref: '#/components/schemas/Category'
      name:
        type: string
        example: doggie
      photoUrls:
        type: array
        xml:
          name: photoUrl
          wrapped: true
        items:
          type: string
      tags:
        type: array
        xml:
          name: tag
          wrapped: true
        items:
          $ref: '#/components/schemas/Tag'
      status:
        type: string
        description: pet status in the store
        enum:
          - available
          - pending
          - sold
    xml:
      name: Pet
```

Tips: the "#" in "#/definitions/Pet" refers to the current document.

For more information about "requestBody", please refer to the "Request Body Object" section in the OpenAPI specification 3.0:

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md#requestBodyObject

The "**parameters**" field lists all the parameters supported by this endpoint. "**in**" specifies the location of the parameter, which can be ***path, query or header***.

The "path" parameter can only be required and it only supports primitive types such as string and integer but not object as there's no way to substitute the path variable with an object.

The "query" parameter is the URL query string appended to the URL. For the following relative path:

```
/pet/108?format=json&fields=all
```

“format” and “fields” are optional “query” parameters.

The “header” parameter defines the key-value pair in the header of the HTTP request. If the HTTP request contains “X-API-VERSION: v3.1.5” in the header, then the header parameter is “X-API-VERSION”.

The “**response**” section defines the HTTP response from the server. In our example, the HTTP response body does not contain any data. To describe a response that can be either in JSON or XML, we can use the following:

```
responses:
  '200':
    description: successful operation
    content:
      application/xml:
        schema:
          $ref: '#/components/schemas/Pet'
      application/json:
        schema:
          $ref: '#/components/schemas/Pet'
```

Where “Pet” is defined as an object mentioned above. The raw HTTP response may look like the following:

```
HTTP/1.1 200 OK
Date: Mon, 11 Sep 2017 08:21:34 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Mon, 11 Sep 2017 07:11:20 GMT
Content-Length: 88
Content-Type: application/json
Connection: Closed

{
  "id": 109,
  "category": {
    "id": 2,
    "name": "animal"
  },
  "name": "Tiger",
  "photoUrls": [
    "bit.ly/2mzaoTL"
  ],
  "tags": [
    {
      "id": 3,
      "name": "big"
    }
  ],
  "status": "available"
}
```

To model an array of model (JSON or XML) in the response, we can use the following:

```
responses:
  '200':
    description: successful operation
    content:
      application/json:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/Pet'
```

What if the HTTP response is the following in which the JSON keys are not fixed?

```
{
  "cat": 2,
  "dog": 3,
  "rabbit": 5
}
```

We can use "**additionalProperties**" to describe a map or dictionary. Here is an example to describe the JSON payload above.

```
responses:
  '200':
    description: successful operation
    content:
      application/json:
        schema:
          type: object
          additionalProperties:
            type: integer
            format: int32
```

The last part of the endpoint definition is "**security**", which is optional. This section lists out security definitions(s) required for this endpoint. The following shows the endpoint only requires "petstore_auth" defined in the security definition section.

```
security:
  - petstore_auth:
      - 'write:pets'
      - 'read:pets'
```

If the endpoint does not require any authentication, you can simply omit the "**security**" section in the endpoint definition.

3.6 Editor

Instead of writing the specification file with a text pad or other YAML editor, you may want to try the following open-source tools to edit the specification file:

- **Apicurio Studio** (http://www.apicur.io/) by RedHat is a new contender to the OpenAPI editor space. Being free and open-source[28], it is still in beta release and for a list of upcoming features, please refer to the roadmap[29].

- If you use Eclipse for software development, you probably want to try **KaiZen OpenAPI Editor**[30] (also free, open-source), which can be easily installed from the Eclipse marketplace[31]. It is developed by RepreZen (https://www.reprezen.com/), which offers RepreZen Studio - an IDE focusing on API design.

The benefit of using these tools is that it will show you the errors or warnings in case it spots issues with the specification.

3.7 Converter

Before OpenAPI specification emerged as the industry standard, there were different formats to describe RESTful APIs. RAML[32], API Blueprint[33] just to name a few. To convert various formats into OpenAPI specification, here are some free, open-source tools to do the job for you:

- **API Spec Transformer** (https://github.com/stoplightio/api-spec-converter) by **Stoplight**[34], which is a SaaS provider focusing on building and testing APIs, supports transformation between various formats: OpenAPI 2.0, RAML (0.8, 1.0), Postman Collection 1.0[35].

- **API Spec Converter** by **LucyBot**[36] (a startup providing solutions to make it easy for developers to get up and running with APIs) is another excellent converter, which is available online via https://lucybot-inc.github.io/api-spec-converter/ and source code can be found in https://github.com/LucyBot-Inc/api-spec-converter. Currently, it supports 7 different formats: Swagger 1.x, OpenAPI 2.0 & 3.0, RAML 0.8, IO Docs[37], API Blueprint, Google API Discovery[38], WADL[39].

- **API-Flow** (https://github.com/luckymarmot/API-Flow) is another free and open-source alternative offered by **Paw**[40], which is an advanced API tool for Mac. As of Sep 2017, it supports OpenAPI 2.0, RAML v1.0, Postman Collection v2.0, Paw

28 https://github.com/apicurio/apicurio-studio
29 http://www.apicur.io/roadmap/
30 https://github.com/RepreZen/KaiZen-OpenAPI-Editor
31 https://marketplace.eclipse.org/content/kaizen-openapi-editor
32 https://raml.org/
33 https://apiblueprint.org/
34 https://stoplight.io/
35 https://github.com/stoplightio/api-spec-converter#supported-conversions
36 http://lucybot.com/
37 https://github.com/mashery/iodocs
38 https://developers.google.com/discovery/v1/reference/apis
39 http://www.w3.org/Submission/wadl/
40 https://paw.cloud/

v3.1 and plans to support OpenAPI 3.0, RAML 0.8, Postman Collection v1.0, Postman Dump v1.0, Insomnia v3.0 and API Blueprint in the future.

3.8 Plugins

Instead of using the contract-first approach to write the specification manually, there are plugins that can generate OpenAPI/Swagger specification from the API server's source code so that the source code is always in sync with the specification. Here are a few examples:

- **Swashbuckle** (https://github.com/domaindrivendev/Swashbuckle): Seamlessly adds an OpenAPI to C# WebApi projects.

- **swagger-inline** (https://github.com/readmeio/swagger-inline) by ReadMe[41]: Writes your OpenAPI file as comments.

- **swagger-php** (https://github.com/zircote/swagger-php): which generates interactive OpenAPI documentation for your RESTful API using doctrine annotations.

[41] http://readme.io/

4. Introduction to OpenAPI Generator

4.1 History

Before we deep dive into OpenAPI Generator, here is a little bit of history and how I got involved. As mentioned, the project "Swagger Codegen" was started by Tony Tam[42] back in 2011. The first version was written in Java but later they switched it to Scala so as to attract more open-source developers to contribute to the project. However, the exact opposite happened as it failed to attract Scala developers to work on the project. In early 2015, the project was rewritten in Java again as part of the major enhancement to support OpenAPI 2.0 specification.

I started voluntarily contributing to this project with bug fixes and enhancement in early 2015 right after the project was rewritten in Java. Looking back it's one of the best things I've ever done as so many developers benefit from this tool and many companies, from start-ups to Internet giants, are using OpenAPI Generator in their production environment to streamline software development. Later in this book, I will explain more on why you should also contribute and how to start contributing back to the project.

4.2 Installation

One of the reasons why OpenAPI Generator goes mainstream is that there are several ways to run it on different platforms (Windows, Mac, Linux), with or without installing Java Runtime.

4.2.1 Using OpenAPI Generator CLI JAR (stable release)

OpenAPI Generator comes with a command-line interface (CLI) under the sub-module modules/openapi-generator-cli. The CLI is a Java JAR and therefore you will need to install Java Runtime 8 or later. The stable releases can be downloaded from MVNRespository.com or other maven repositories by searching for "OpenAPI Generator". As of this writing, the latest release is 5.0.1. Here is the command to download the JAR and obtain the usage from the command prompt:

[Mac/Linux/Windows]

```
wget
https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli/5.0
.1/openapi-generator-cli-5.0.1.jar
java -jar openapi-generator-cli-5.0.1.jar help
```

Note: For Windows users, please install PowerShell 5.0 or later in order to follow the examples in this book.

This will show the following in the output:

[42] https://www.linkedin.com/in/tonytam/

```
usage: openapi-generator-cli <command> [<args>]

The most commonly used openapi-generator-cli commands are:
    config-help   Config help for chosen lang
    generate      Generate code with the specified generator.
    help          Display help information about openapi-generator
    list          Lists the available generators
     meta            MetaGenerator. Generator for creating a new template
set and configuration for Codegen.   The output will be based on the
language you specify, and includes default templates to include.
    validate      Validate specification
    version       Show version information used in tooling

See 'openapi-generator-cli help <command>' for more information on a
specific
command.
```

We're not going to walk through every single option here. At this point, you have a working CLI (Java JAR) ready for you to generate some codes.

You may find it useful to obtain the OpenAPI Generator CLI version by running

```
java -jar openapi-generator-cli-5.0.1.jar version
```

which will return *5.0.1*

We are also adding the OpenAPI Generator version to the auto-generated files so that users can easily tell from the files which version of OpenAPI Generator was used to generate the code.

4.2.2 Using OpenAPI Generator CLI JAR (SNAPSHOT version)

Starting in July 2017, we have updated our continuous integration workflow to publish a SNAPSHOT version of current master, which is version 5.0.1 as of this writing, to SonaType SNAPSHOT repositories[43]. Here are the URLs to the latest SNAPSHOT version of openapi-generator-cli:

[5.0.1-SNAPSHOT] -
https://oss.sonatype.org/content/repositories/snapshots/org/openapitools/openapi-generator-cli/5.0.1-SNAPSHOT/

If the developers cannot install the dependencies for building the JAR locally for whatever reasons (for instance lack of administrator privilege to install software), using the SNAPSHOT version is a good alternative before the next stable release becomes available.

[43] https://oss.sonatype.org/content/repositories/snapshots/

4.2.3 Building JAR from the latest master (recommended)

This is the recommended approach as the project is extremely active and building the JAR from the latest master allows you to enjoy the latest enhancement and bug fixes contributed by OpenAPI Generator core team and the community. This also allows you to roll back OpenAPI Generator to a particular commit if the latest contains undesired enhancements that you prefer not to use at the moment.

To build the project, you will need to have Java Runtime 8 or later, Apache Maven and git installed. Maven is well-known for building Java-based projects and the minimum version is 3.3.3.

For Mac, we recommend using Homebrew: *brew install maven*

For Linux, *sudo apt-get install maven* should do the job.

For Windows, please follow the instructions in this answer in StackOverflow as the instructions are quite detailed and we are not going to repeat them here.

After a successful installation, *mvn -version* will display something similar to the following:

```
Apache Maven 3.3.3 (7994120775791599e205a5524ec3e0dfe41d4a06;
2015-04-22T19:57:37+08:00)
Maven home: /usr/local/Cellar/maven/3.3.3/libexec
Java version: 1.8.0_112, vendor: Oracle Corporation
Java home:
/Library/Java/JavaVirtualMachines/jdk1.8.0_112.jdk/Contents/Home/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "10.12.4", arch: "x86_64", family: "mac"
```

Another dependency is ***git***[44], which is by far the most popular version control system in the open source community. For installation of git in different OS, please refer to the tutorial by Atlassian. We will not cover the basic of git in this book and here are 2 easy-to-follow tutorials in case you are new to git:

1. Github: Hello Word
2. Atlassian: Setting up a repository

Even though you are new to git, our examples will show the exact commands being used so that everyone can follow easily.

[44] https://git-scm.com/

In the following, we will show you how to clone the OpenAPI Generator project from Github and build the project locally on your machine. To begin, open a new terminal or shell and run the following commands one by one:

```
git clone https://github.com/OpenAPITools/openapi-generator.git
cd openapi-generator
mvn clean install -DskipTests
```

Tips: "mvn install" will install the JARs in the local maven repository and "-DskipTests" will skip all the tests so as to speed up the JAR building process. "-pl modules/openapi-generator-cli" will only build the JAR for the OpenAPI Generator CLI module.

After the commands run successfully, you will see the following:

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] openapi-generator-project 5.0.1-SNAPSHOT ........... SUCCESS [ 24.417 s]
[INFO] openapi-generator-core ............................. SUCCESS [ 21.044 s]
[INFO] openapi-generator (core library) ................... SUCCESS [02:15 min]
[INFO] openapi-generator (executable) ..................... SUCCESS [ 22.867 s]
[INFO] openapi-generator (maven-plugin) ................... SUCCESS [ 28.357 s]
[INFO] openapi-generator-gradle-plugin (maven wrapper) .... SUCCESS [01:47 min]
[INFO] openapi-generator-online 5.0.1-SNAPSHOT ............ SUCCESS [ 27.391 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 06:11 min
[INFO] Finished at: 2021-01-12T16:41:34+08:00
[INFO] ------------------------------------------------------------------------
```

Which means the project has been built successfully and you can find the latest JAR for openapi-generator-cli in *modules/openapi-generator-cli/target/openapi-generator-cli.jar*. Using the JAR you just built, you can run the following command to obtain the usage:

```
java -jar modules/openapi-generator-cli/target/openapi-generator-cli.jar help
```

which will display the usage shown in the previous section.

What if we do not want to build the JAR based on the latest master but instead based on a particular change made to the project? We can easily do so by checking out the project at a particular commit, which represents a change in git and comes with a commit hash to uniquely identify the change. To get a list of commits for OpenAPI Generator, please refer to "commits"[45] in the project homepage[46]. As an example, we will checkout the commit da9ba4cd7d5461647492b61aaf74cb850ded2456[47]:

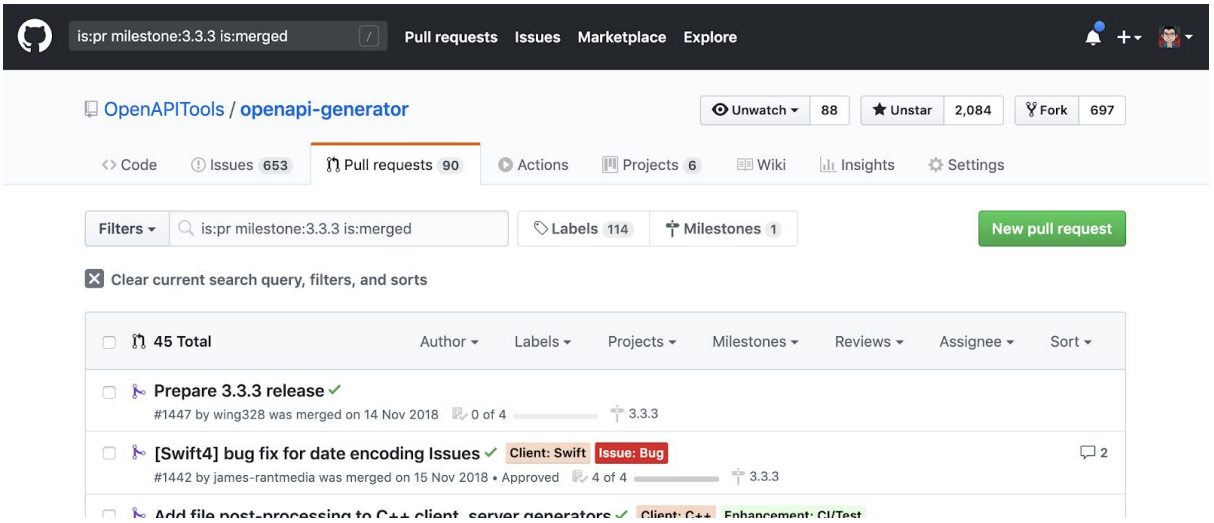
45 https://github.com/openapitools/openapi-generator/commits/master
46 https://github.com/openapitools/openapi-generator
47 http://bit.ly/31AjCjk

```
git clone https://github.com/openapitools/openapi-generator.git
cd openapi-generator
git checkout da9ba4cd7d5461647492b61aaf74cb850ded2456
mvn clean install -DskipTests
```

Tips: instead of using da9ba4cd7d5461647492b61aaf74cb850ded2456, you can also use its short form da9ba4c: git checkout da9ba4c

This way the JAR will be built based on a particular change to the project. This trick is useful when you do not want to use the latest enhancements or bug fixes, which may break your current usage of OpenAPI Generator.

As of Dec 2020, the latest stable release is 5.0.0, which means the enhancements, bug fixes will NOT be backward compatible with the previous stable release 4.3.1. The current master is 5.0.1, which contains backward compatible changes. The release note in the release page highlights all the important changes[48]. To get a list of enhancements or bug fixes merged into the 3.3.3 release, you can filter the pull requests with "***is:pr milestone:3.3.3 is:merged***" and here is the result:

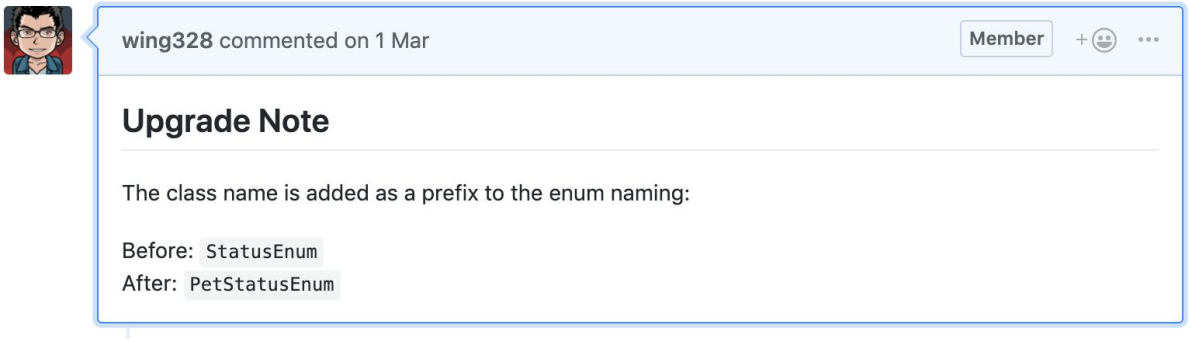


https://github.com/openapitools/openapi-generator/pulls?utf8=%E2%9C%93&q=is%3Apr%20milestone%3A3.3.3%20is%3Amerged

Tips: "pull requests" are change requests in Git terms. If you want to make a change to a Git repo that you do not own, you will need to fork the repo first and then submit a pull request. We will go into details later when we talk about contributing back to OpenAPI Generator.

To understand more about the breaking changes, we also provide an upgrade note to ease the migration. Here is an example of upgrade note:

[48] https://github.com/openapitools/openapi-generator/releases



Ref:
https://github.com/OpenAPITools/openapi-generator/pull/2242#issuecomment-468630877

If you need help or have any questions with the changes (breaking or non-breaking), just reply to the pull request. Of course, you will need to register[49] a Github account in order to leave a comment.

4.2.4 Online generator (https://api.openapi-generator.tech)

If you do not want to install any of the dependencies or have troubles with the installation, you can still use OpenAPI Generator with the online generator: https://api.openapi-generator.tech. All you need is the "curl"[50] command, which is the Swiss-army knife for interacting with HTTP servers. For Windows users, you can download it via https://curl.haxx.se/download.html. Here is a quick example to generate a Ruby API client for the OpenAPI spec "http://pubi.org/petstore_oas3.json":

```
curl -X POST -H "content-type:application/json" -d
'{"openAPIUrl":"http://pubi.org/petstore_oas3.json"}'
https://api.openapi-generator.tech/api/gen/clients/ruby
```

And the result will look like the following:

```
{"code":"ff1fcd6b-e062-46fd-9e7b-8fff99514571","link":"https://api.openap
i-generator.tech/api/gen/download/ff1fcd6b-e062-46fd-9e7b-8fff99514571"}
```

The zipped file, which contains the Ruby SDK, can be downloaded via https://api.openapi-generator.tech/api/gen/download/ff1fcd6b-e062-46fd-9e7b-8fff99514571 as shown in the result.

***IMPORTANT**: https://api.openapi-generator.tech usually runs **the latest stable version of OpenAPI Generator**, which is 5.0.1 as of this writing. To enjoy the latest enhancements and bug fixes, you can use https://api-latest-master.openapi-generator.tech instead or run the server locally.*

[49] https://github.com/join
[50] https://curl.haxx.se/

In the next chapter, we will go into details on how to pass different options to customize the auto-generated code by the online generator.

At this point, we are able to use OpenAPI Generator with just “curl”, nothing else.

If you do not want your zipped source codes hosted on a public server or want more control over the server such as a different port to listen on or which generator version to run, you can also run the online OpenAPI Generator locally on any machine you want. The OpenAPI Generator online project lives under the sub-module “modules/openapi-generator-online/”[51]. Here are step by step instructions to build the generator and run it locally.

```
git clone https://github.com/openapitools/openapi-generator.git
cd openapi-generator
mvn clean install -DskipTests
cd modules/openapi-generator-online
mvn jetty:run
```

If the server runs successfully, it should show the following:

```
[INFO] Started ServerConnector@498b697{HTTP/1.1}{0.0.0.0:8080}
[INFO] Started @15872ms
[INFO] Started Jetty Server
```

To test the local online OpenAPI Generator, we can rerun the curl command with a different host:

```
curl -X POST -H "content-type:application/json" -d
'{"openAPIUrl":"http://pubi.org/petstore_oas3.json"}'
https://localhost:8080/api/gen/clients/ruby
```

Then it should show similar results.

For companies with multiple teams leveraging OpenAPI Generator, you may find hosting an online OpenAPI Generator internally in your intranet desirable as it would allow you to manage OpenAPI Generator in a centralized manner. Having different teams running the OpenAPI Generator CLI (JAR) locally on their machines may run into potential issues that teams are not using the same version of OpenAPI Generator to generate codes.

4.2.5 Docker

If you have Docker[52] installed, you can start using OpenAPI Generator right away. Most server stub generators in OpenAPI Generator provide out-of-the-box support for Docker. For those who are not familiar with Docker, it is a very popular tool to create and run applications using containers, which package all the dependencies required to run the application. Unlike virtual machines (VM), Docker allows the containers to share the same system kernel to

[51] https://github.com/openapitools/openapi-generator/tree/master/modules/openapi-generator-online
[52] https://www.docker.com/

avoid the overhead in building a separate virtual operating system. The “Get started” guide in Docker.com is a good starting point to familiarize yourself with Docker[53]. (For the following examples, we are using Docker version 17.09.0-ce, build afdb6d4)

The easiest way to use OpenAPI Generator with Docker is to use the “openapi-generator-cli” image: https://hub.docker.com/r/openapitools/openapi-generator-cli/. Just run the following command in “/tmp” directory to generate Ruby API client for Pet Store API specification:

```
$ docker run --rm -v ${PWD}:/local openapitools/openapi-generator-cli
generate \
    -i http://pubi.org/petstore_oas3.json \
    -g ruby \
    -o /local/out/ruby
```

and it should output the following:

```
$ docker run --rm -v ${PWD}:/local openapitools/openapi-generator-cli
generate \
    -i http://pubi.org/petstore_oas3.json \
    -g ruby \
    -o /local/out/ruby
Unable to find image 'openapitools/openapi-generator-cli:latest' locally
latest: Pulling from openapitools/openapi-generator-cli
709515475419: Pull complete
38a1c0aaa6fd: Pull complete
cd134db5e982: Pull complete
614b16f7a66e: Pull complete
a881d9a2ace8: Pull complete
401f1f948074: Pull complete
Digest:
sha256:317aac3ea1b69405aa60064a10ad013d06132b816bb51a5def1a5162d45d43fb
Status: Downloaded newer image for
openapitools/openapi-generator-cli:latest
[main] WARN  o.o.c.ignore.CodegenIgnoreProcessor - Output directory does
not exist, or is inaccessible. No file (.openapi-generator-ignore) will
be evaluated.
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
```

[53] https://docs.docker.com/get-started/

```
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found in
content, returning only the first one
[main] INFO  o.o.c.languages.AbstractRubyCodegen - Hint: Environment
variable 'RUBY_POST_PROCESS_FILE' (optional) not defined. E.g. to format
the source code, please try 'export
RUBY_POST_PROCESS_FILE="/usr/local/bin/rubocop -a"' (Linux/Mac)
[main] INFO  o.o.codegen.AbstractGenerator - writing file
/local/out/ruby/lib/openapi_client/models/api_response.rb
[main] INFO  o.o.codegen.AbstractGenerator - writing file
/local/out/ruby/spec/models/api_response_spec.rb
[main] INFO  o.o.codegen.AbstractGenerator - writing file
/local/out/ruby/docs/ApiResponse.md
[main] INFO  o.o.codegen.AbstractGenerator - writing file
/local/out/ruby/lib/openapi_client/models/category.rb
[main] INFO  o.o.codegen.AbstractGenerator - writing file
/local/out/ruby/spec/models/category_spec.rb
[main] INFO  o.o.codegen.AbstractGenerator - writing file
/local/out/ruby/docs/Category.md

(result omitted...)
```

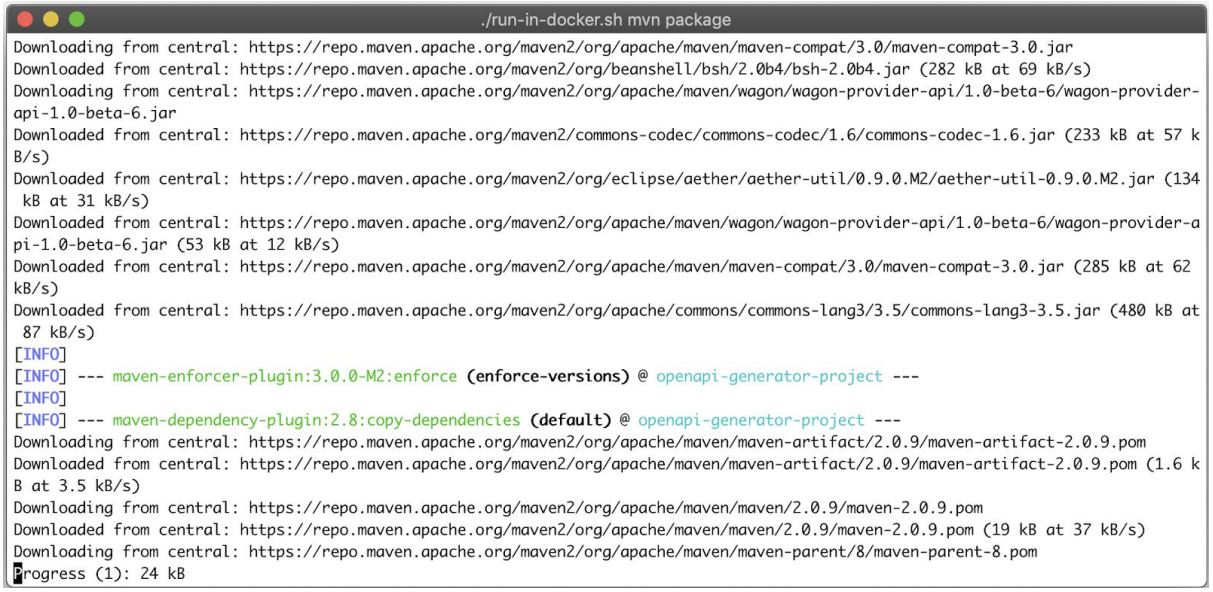
Docker tried to find the “openapitools/openapi-generator-cli” image locally but couldn’t and then automatically pull the latest from https://hub.docker.com. After finishing the setup, it generated the Ruby API client for the Pet Store API specification. The Ruby client can be found in /tmp/out/ruby.

What if you want to play with OpenAPI Generator and apply some customization to the generators? You can also do that with Docker using a Bash script included in the project. We will start with the following commands to build the project inside the Docker container:

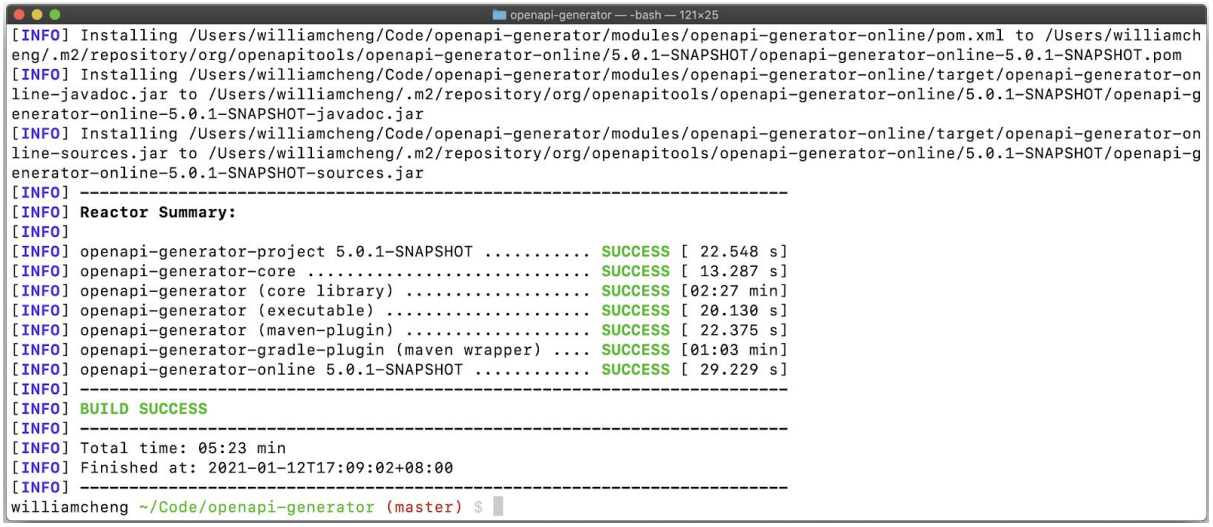
```
git clone https://github.com/openapitools/openapi-generator
cd openapi-generator
```

```
./run-in-docker.sh mvn package
```

The “run-in-docker.sh” acts like a shell or session to run the command “mvn package”, which builds the OpenAPI Generator project using Maven. Here is what the output (partial) looks like:



This may take awhile as Maven tries to download all the dependencies from the Maven repositories for the first time. Eventually, it should show the following:



“run-in-docker.sh” also acts like the OpenAPI Generator executable installed via Homebrew. It can run various options directly such as “help”, generate” and more. Here are some examples

To show the usage, just run

```
./run-in-docker.sh help
```

To list out all the generators available, run with the command line option “list”

```
./run-in-docker.sh list
```

To run the Ruby Petstore sample script, you will need to call the “bin/generate-samples.sh” script under the “/gen” mount point. You can think of “/gen” as the root directory to the container.

```
./run-in-docker.sh /gen/bin/generate-samples.sh ./bin/config/ruby.yaml
```

Or you can “generate” Ruby API client directly

```
./run-in-docker.sh generate \
 -i http://pubi.org/petstore_oas3.json \
 -g ruby -o /gen/out/ruby-petstore --additional-properties
gemVersion=1.2.3
```

Another Docker image worth mentioning is “openapi-generator-online”[54], which allows you to fire up an instance of openapi-generator-online locally via Docker. We will walk through a simple example on generating Ruby API client below:

To start the generator, it is as simple as running the following command, which will pull the latest container image of openapi-generator-online if not found in the local cache:

```
docker run -e "GENERATOR_HOST=http://localhost:8080" -p 8080:8080 -d
openapitools/openapi-generator-online
```

Please wait for 10 seconds to let the generator start at port 8080. Then we will make an HTTP request to generate Ruby API client for Pet Store API specification:

```
curl -X POST -H "content-type:application/json" -d
'{"openAPIUrl":"http://pubi.org/petstore_oas3.json"}'
http://localhost:8080/api/gen/clients/ruby
```

The response is a JSON file with an URL to download the auto-generated code.

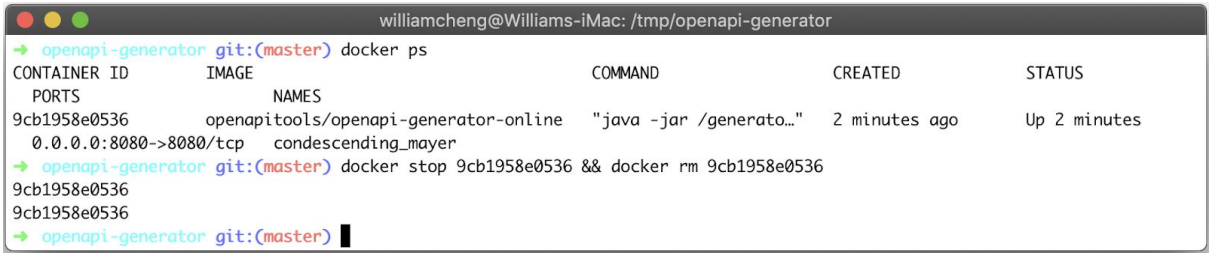
```
{"code":"0260466d-cc7c-4349-93be-22e900baef61","link":"http://localhost:8
080/api/gen/download/0260466d-cc7c-4349-93be-22e900baef61"}
```

We can easily “curl” the link directly to get the zipped Ruby API client.

```
curl
http://localhost:8080/api/gen/download/0260466d-cc7c-4349-93be-22e900baef
61 > result.zip
```

54 https://hub.docker.com/r/openapitools/openapi-generator-online

Finally, we stop the generator and clear all the temporary files by first running “docker ps” to identify the container ID and then using the “stop”, “rm” option to terminate the process and clear all the temporary files:



Docker support in OpenAPI Generator can come in handy if you are already familiar with Docker.

4.2.6 Maven/Gradle plug-in

You can also integrate OpenAPI Generator into your existing workflow if you are using Maven or Gradle. The full documentation on the OpenAPI Generator Maven plug-in can be found in [openapi-generator/tree/master/modules/openapi-generator-maven-plugin](). Here is a sample pom.xml to generate Ruby API client for Pet Store API specification using OpenAPI Generator Maven plugin:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.openapitools.Petstore</groupId>
    <artifactId>ruby-demo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <build>
        <plugins>
            <!-- activate the plugin -->
            <plugin>
                <groupId>org.openapitools</groupId>
                <artifactId>openapi-generator-maven-plugin</artifactId>
                <version>5.0.1</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>generate</goal>
                        </goals>
                        <configuration>
                            <!-- specify the location of the spec -->

<inputSpec>http://pubi.org/petstore_oas3.json</inputSpec>
                            <!-- target to generate ruby client code -->
```

```
                            <language>ruby</language>
                            <!-- general option -->

<gitUserId>openapi-generator-ebook</gitUserId>
                            <gitRepoId>OnlinePetstore</gitRepoId>
                            <!-- language specified options -->
                            <configOptions>
                                <gemName>online_petstore</gemName>
                                <moduleName>OnlinePetstore</moduleName>
                                <gemLicense>MIT</gemLicense>
                                <gemVersion>0.3.1</gemVersion>

<gemRequiredRubyVersion>0.3.1</gemRequiredRubyVersion>
                            </configOptions>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

(Ref: https://github.com/openapi-generator-ebook/introduction/blob/master/pom.xml)

The XML comments explain what each section does. It is essentially the same as generating the code via the OpenAPI Generator CLI and put the options in the form of XML.

Then after running “mvn package”, you will find the Ruby API client under the folder “target/generated-sources/openapi/”

The OpenAPI Generator project also comes with an official Gradle plug-in:

https://github.com/OpenAPITools/openapi-generator/blob/master/modules/openapi-generator-gradle-plugin/README.adoc

5. Customizing the autogenerated code

In this chapter, we will walk you through several commonly used options to customize the auto-generated code. The project has matured over the years with the help of contributors and the OpenAPI Generator core team to add various options to customize the output so as to meet different requirements.

5.1 The Basic

First and foremost, how can we get a list of languages supported by OpenAPI Generator? It is as easy as running the following command:

```
java -jar modules/openapi-generator-cli/target/openapi-generator-cli.jar
list
```

and you will get the following:

```
CLIENT generators:
    - ada
    - android
    - apex
    - bash
    - c
    - clojure
    - cpp-qt5-client
    - cpp-restsdk
    - cpp-tizen
    - cpp-ue4 (beta)
    - csharp
    - csharp-netcore
    - dart
    - dart-dio
    - dart-jaguar
    - eiffel
    - elixir
    - elm
    - erlang-client
    - erlang-proper
    - go
    - groovy
    - haskell-http-client
    - java
    - javascript
    - javascript-apollo (beta)
    - javascript-closure-angular
    - javascript-flowtyped
    - jaxrs-cxf-client
```

```
    - jmeter
    - k6 (beta)
    - kotlin
    - lua (beta)
    - nim (beta)
    - objc
    - ocaml
    - perl
    - php
    - powershell (beta)
    - python (experimental)
    - python-legacy
    - r
    - ruby
    - rust
    - scala-akka
    - scala-gatling
    - scala-sttp (beta)
    - scalaz
    - swift5 (beta)
    - typescript (experimental)
    - typescript-angular
    - typescript-aurelia
    - typescript-axios
    - typescript-fetch
    - typescript-inversify
    - typescript-jquery
    - typescript-node
    - typescript-redux-query
    - typescript-rxjs


SERVER generators:
    - ada-server
    - aspnetcore
    - cpp-pistache-server
    - cpp-qt5-qhttpengine-server
    - cpp-restbed-server
    - csharp-nancyfx
    - erlang-server
    - fsharp-functions (beta)
    - fsharp-giraffe-server (beta)
    - go-gin-server
    - go-server
    - graphql-nodejs-express-server
    - haskell
    - java-inflector
    - java-msf4j
    - java-pkmst
    - java-play-framework
    - java-undertow-server
```

```
    - java-vertx
    - java-vertx-web (beta)
    - jaxrs-cxf
    - jaxrs-cxf-cdi
    - jaxrs-cxf-extended
    - jaxrs-jersey
    - jaxrs-resteasy
    - jaxrs-resteasy-eap
    - jaxrs-spec
    - kotlin-server
    - kotlin-spring
    - kotlin-vertx (beta)
    - nodejs-express-server (beta)
    - php-laravel
    - php-lumen
    - php-mezzio-ph
    - php-slim4
    - php-symfony
    - python-aiohttp
    - python-blueplanet
    - python-flask
    - ruby-on-rails
    - ruby-sinatra
    - rust-server
    - scala-akka-http-server (beta)
    - scala-finch
    - scala-lagom-server
    - scala-play-server
    - scalatra
    - spring


DOCUMENTATION generators:
    - asciidoc
    - cwiki
    - dynamic-html
    - html
    - html2
    - markdown (beta)
    - openapi
    - openapi-yaml
    - plantuml (beta)


SCHEMA generators:
    - avro-schema (beta)
    - graphql-schema
    - mysql-schema
    - protobuf-schema (beta)
```

```
CONFIG generators:
    - apache2
```

which contains more than 100 languages (or generators).

To use the latest generator, you will need to build the project locally following the instructions described in the previous chapter.

If you want to add a new generator, we would strongly recommend you to search in the "Issues"[55] section of the Github repo or open a new issue[56] to start the discussion as someone in the community may have already started working on that and starting a discussion to begin with will help avoid duplicated efforts.

Besides the "**version**" option we covered in the previous chapter to show the OpenAPI Generator version, you may also find the "**validate**" option useful. As the name implies, the "**validate**" option is for validating the input OpenAPI 2.0 or 3.0 specification. Here is an example:

```
$ java -jar openapi-generator-cli.jar validate -i
https://raw.githubusercontent.com/openapi-generator-ebook/introduction/ma
ster/invalid_petstore.json
```

If the OpenAPI 2.0 or 3.0 specification contains invalid definitions, the command will show warnings and/or errors about the issues as shown below:

```
Validating spec
(https://raw.githubusercontent.com/openapi-generator-ebook/introduction/m
aster/invalid_petstore.json)
Errors:
      -attribute paths.'/pet/{petId}'(get).[petId].type is missing

[error] Spec has 1 errors.
```

Tips: OpenAPI Generator leverages another open-source project "Swagger-Parser"[57] to validate the OpenAPI specification and parse the specification into Java POJOs.

There are essentially 2 types of options to control the output: general options and language-specific options. General options are available to all generators while language-specified options are for a particular generator only. To get a list of general options, just run *java -jar openapi-generator-cli.jar help generate* and you will find the output in Appendix A.

[55] https://github.com/openapitools/openapi-generator/issues
[56] https://github.com/openapitools/openapi-generator/issues/new
[57] https://github.com/swagger-api/swagger-parser

There are more than 30 options and we will not go through all of them in this book but we will instead illustrate the usage for some of the options as we later walk through use cases of using OpenAPI Generator.

For language-specific options, we need to use the "-g" command line switch with "config-help". Here is an example:

```
$ java -jar openapi-generator-cli.jar config-help -g python

CONFIG OPTIONS

      generateSourceCodeOnly
           Specifies that only a library source code is to be generated.
(Default: false)

      hideGenerationTimestamp
            Hides the generation timestamp when files are generated.
(Default: true)

      library
            library template (sub-template) to use: asyncio, tornado,
urllib3 (Default: urllib3)

      packageName
             python package name (convention: snake_case). (Default:
openapi_client)

      packageUrl
          python package URL.

      packageVersion
          python package version. (Default: 1.0.0)

      projectName
          python project name in setup.py (e.g. petstore-api).

      pythonAttrNoneIfUnset
            when accessing unset attribute, return `None` instead of
raising `ApiAttributeError` (Default: false)

      recursionLimit
           Set the recursion limit. If not set, use the system default
value.

      useNose
          use the nose test framework (Default: false)
```

For the Python API client generator, there are 8 options available at the moment while other generators may have more.

5.2 Example: Ruby SDK for Pet Store API

In this example, we will be generating Ruby SDK for Pet Store API and its specification can be found in http://pubi.org/petstore_oas3.json.

We will start with the following command without any options:

[Mac/Linux]

```
java -jar openapi-generator-cli.jar generate \
   -i http://pubi.org/petstore_oas3.json \
   -g ruby \
   -o /var/tmp/petstore_api/ruby/
```

[Windows]

```
java -jar openapi-generator-cli.jar generate ^
   -i http://pubi.org/petstore_oas3.json ^
   -g ruby ^
   -o C:\temp\petstore_api\ruby\
```

To output the code in the "/var/tmp/petstore_api/ruby" folder in Mac/Linux or "C:\temp\petstore_api\ruby" in Windows, we use the "-o" command line switch. Otherwise, the autogenerated files will be put in the current folder by default.

This will give you a Ruby SDK for Pet Store API using the default values and here is console output (partial):

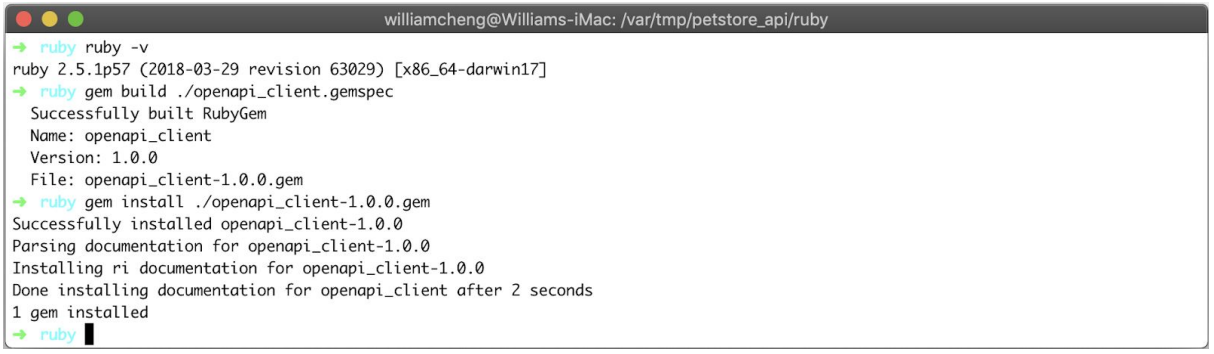
```
williamcheng@Williams-iMac: /tmp
➜ /tmp java -jar openapi-generator-cli.jar generate \
  -i http://petstore.swagger.io/v2/swagger.json \
  -g ruby \
  -o /var/tmp/petstore_api/ruby/

[main] WARN  o.o.c.ignore.CodegenIgnoreProcessor - Output directory does not exist, or is inaccessible. No file (.openapi-generator
-ignore) will be evaluated.
[main] INFO  o.o.c.languages.AbstractRubyCodegen - Hint: Environment variable 'RUBY_POST_PROCESS_FILE' (optional) not defined. E.g.
 to format the source code, please try 'export RUBY_POST_PROCESS_FILE="/usr/local/bin/rubocop -a"' (Linux/Mac)
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] INFO  o.o.codegen.AbstractGenerator - writing file /var/tmp/petstore_api/ruby/lib/openapi_client/models/api_response.rb
[main] INFO  o.o.codegen.AbstractGenerator - writing file /var/tmp/petstore_api/ruby/spec/models/api_response_spec.rb
```

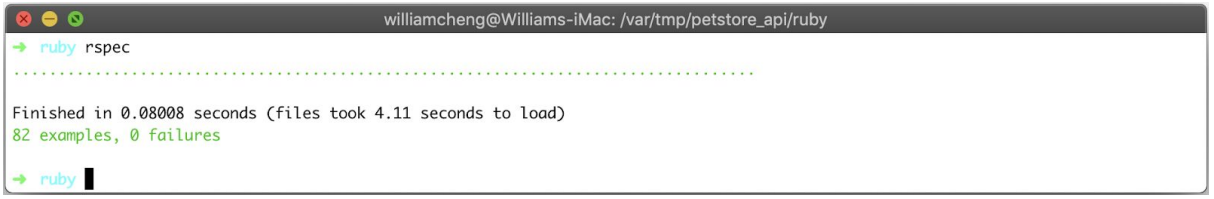
This is definitely a good start as we now have a working Ruby SDK. We can follow the instruction in the README.md to build the gem and install it locally by running

```
gem build ./openapi_client.gemspec
gem install ./openapi_client-1.0.0.gem
```

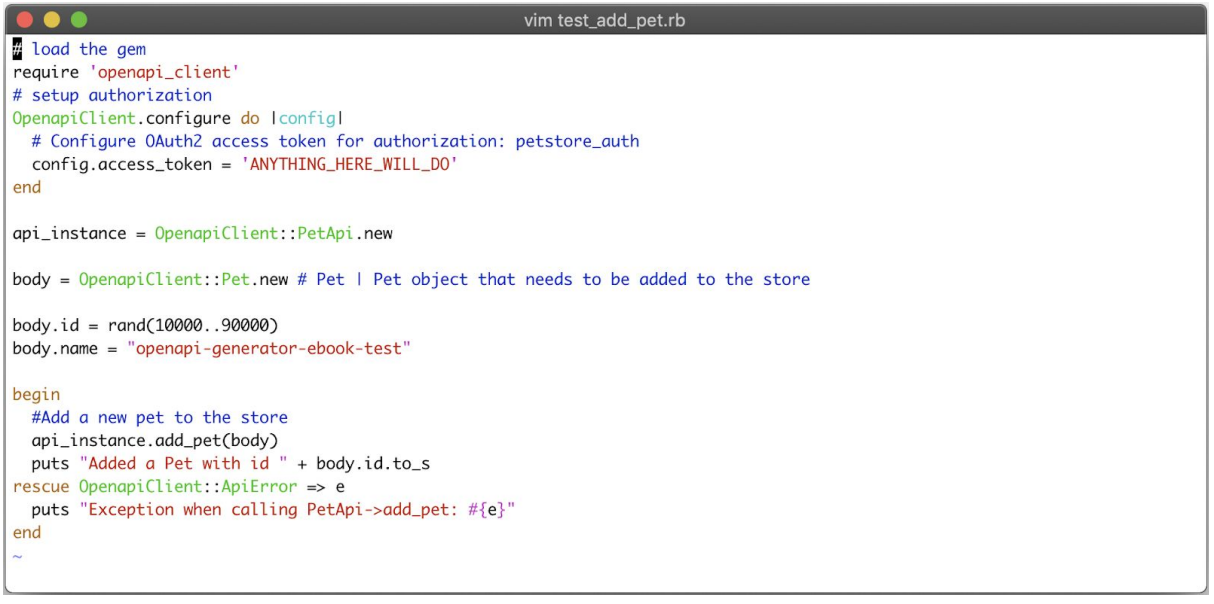
If the installation succeeds, you will see the following output:



The Ruby SDK comes with auto-generated test files under the “spec” folder. To run the test, simply follow the instruction in the README.md to run “rspec” and you will see the following test results:

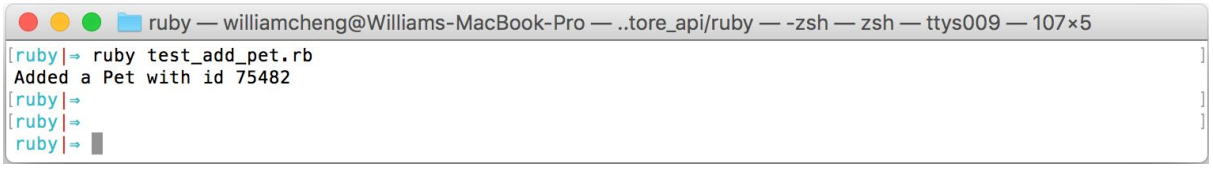


We can also write a very simple Ruby script to test the SDK. The code sample in the auto-generated documentation is a good starting point. For example, the documentation for “add_pet” can be found in the autogenerated markdown documentation “docs/PetApi.md” and the following is a Ruby script to call the “addPet” function so as to add a new Pet with a random ID and the name “openapi-generator-ebook-test” to the online pet store.

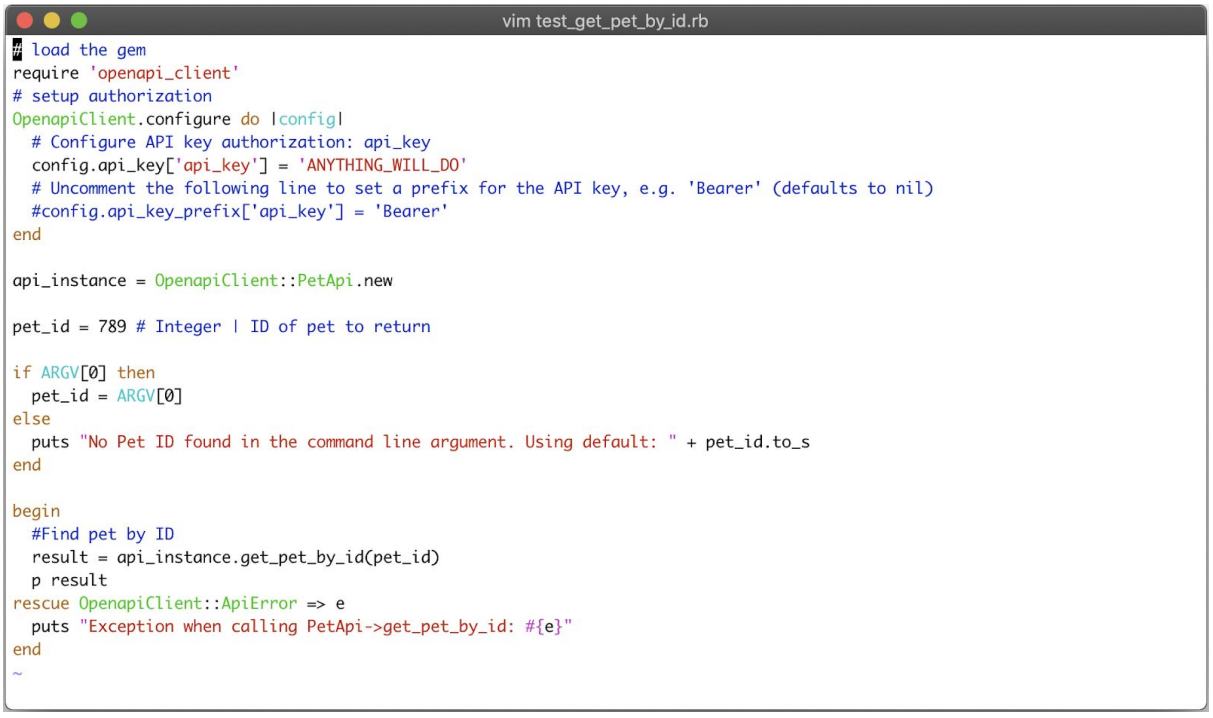


Ref: https://github.com/openapi-generator-ebook/introduction/blob/master/test_add_pet.rb

Here is the result running it:

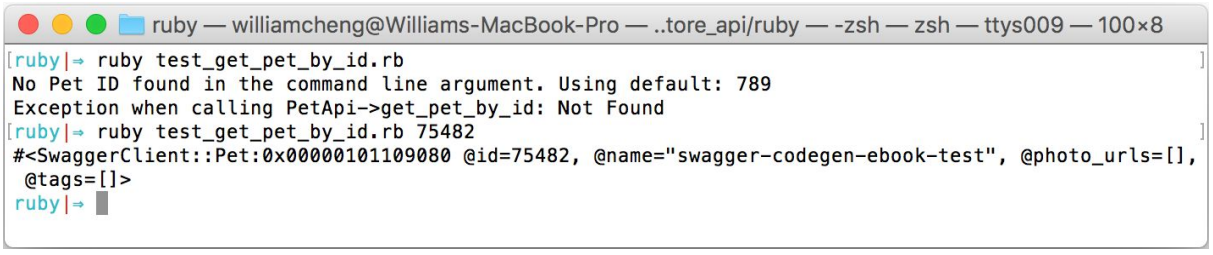


To test whether the new Pet object is stored successfully in the Petstore server, we can call the “get_pet_by_id” function. Here is another Ruby script using the “get_pet_by_id” code sample found in “docs/PetApi.md”:



Ref:
https://github.com/openapi-generator-ebook/introduction/blob/master/test_get_pet_by_id.rb

Here is the result running the script with and without the Pet ID 75482 in the command line argument:



Everything works right out-of-the-box and we have a full-featured Ruby SDKs ready for production usage in less than a minute but can we make it even better such as a better GEM name instead of “OpenapiClient”? The answer is Yes. In the next section, we will explore different ways to customize the Ruby SDK.

5.3 Customization using built-in options

Assuming we get the following requirements from our supervisor:

- The Ruby SDK should be published to Github repo “OnlinePetstore” under the account “openapi-generator-ebook” for easier distribution of the SDK.
- The Ruby SDK should be named “OnlinePetstore”
- The first public release version should be 0.3.1
- The Ruby Gem version should be 2.1
- The license of the SDK should be MIT

To begin customizing the Ruby SDK, we will explore what options are available by running the “config-help” command for Ruby and it should display the following options:

```
java -jar openapi-generator-cli.jar config-help -g ruby

CONFIG OPTIONS

      allowUnicodeIdentifiers
           boolean, toggles whether unicode identifiers are allowed in
names or not, default is false (Default: false)

      disallowAdditionalPropertiesIfNotPresent
           If false, the 'additionalProperties' implementation (set to
true by default) is compliant with the OAS and JSON schema
specifications. If true (default), keep the old (incorrect) behaviour
that 'additionalProperties' is set to false by default. (Default: true)
                  false - The 'additionalProperties' implementation is
compliant with the OAS and JSON schema specifications.
                    true - Keep the old (incorrect) behaviour that
'additionalProperties' is set to false by default.

      ensureUniqueParams
           Whether to ensure parameter names are unique in an operation
(rename parameters that are not). (Default: true)

      gemAuthor
          gem author (only one is supported).

      gemAuthorEmail
          gem author email (only one is supported).

      gemDescription
```

```
        gem description.  (Default: This gem maps to a REST API)

    gemHomepage
        gem homepage.  (Default: http://org.openapitools)

    gemLicense
        gem license.  (Default: unlicense)

    gemName
                gem name (convention: underscore_case). (Default:
openapi_client)

    gemRequiredRubyVersion
        gem required Ruby version.  (Default: >= 1.9)

    gemSummary
        gem summary.  (Default: A ruby wrapper for the REST APIs)

    gemVersion
        gem version. (Default: 1.0.0)

    hideGenerationTimestamp
           Hides the generation timestamp when files are generated.
(Default: true)

    legacyDiscriminatorBehavior
             Set to true for generators with better support for
discriminators. (Python, Java, Go, PowerShell, C#have this enabled by
default). (Default: true)
             true - The mapping in the discriminator includes descendent
schemas that allOf inherit from self and the discriminator mapping
schemas in the OAS document.
              false - The mapping in the discriminator includes any
descendent schemas that allOf inherit from self, any oneOf schemas, any
anyOf schemas, any x-discriminator-values, and the discriminator mapping
schemas in the OAS document AND Codegen validates that oneOf and anyOf
schemas contain the required discriminator and throws an error if the
discriminator is missing.

    library
        HTTP library template (sub-template) to use (Default: typhoeus)
              faraday - Faraday (https://github.com/lostisland/faraday)
(Beta support)
                                    typhoeus - Typhoeus >= 1.0.1
(https://github.com/typhoeus/typhoeus)

    moduleName
         top module name (convention: CamelCase, usually corresponding
to gem name). (Default: OpenAPIClient)

    prependFormOrBodyParameters
```

```
          Add form or body parameters to the beginning of the parameter
list. (Default: false)

      sortModelPropertiesByRequiredFlag
            Sort model properties to place required parameters before
optional parameters. (Default: true)

      sortParamsByRequiredFlag
            Sort method arguments to place required parameters before
optional parameters. (Default: true)
```

The options are self-explanatory but in case you have questions or suggestions regarding these options, please feel free to open a ticket (issue) in the Github repository[58]. We will customize these options with a JSON configuration file, e.g. config.json, as follows:

```
{
    “gemName”: “online_petstore”,
    “moduleName”: “OnlinePetstore”,
    “gemLicense”: “MIT”,
    “gemVersion”, “0.3.1”,
    “gemRequiredRubyVersion”, “2.1”
}
```

What about the requirement to publish the SDK to Github? We will need to provide the general options: gitUserId, gitRepoId:

```
{
    "gemName": "online_petstore",
    "moduleName": "OnlinePetstore",
    "gemLicense": "MIT",
    "gemVersion": "0.3.1",
    "gemRequiredRubyVersion": "2.1",
    "gitUserId": "openapi-generator-ebook",
    "gitRepoId": "OnlinePetstore"
}
```

Ref: https://github.com/openapi-generator-ebook/introduction/blob/master/config.json

To supply the configuration file as part of the code generation, we will need the “-c” option:

```
-c <configuration file>, --config <configuration file>
    Path to json configuration file. File content should be in a json
    format {"optionKey":"optionValue", "optionKey1":"optionValue1"...}
    Supported options can be different for each language. Run
    config-help -g {lang} command for language specific config options.
```

Here is the full command to generate a customized Ruby SDK to meet the requirement:

58 https://github.com/openapitools/openapi-generator/issues/new

[Mac/Linux]

```
java -jar openapi-generator-cli.jar generate \
   -i http://pubi.org/petstore_oas3.json \
   -g ruby \
   -o /var/tmp/petstore_api2/ruby/ \
   -c config.json
```

[Windows]

```
java -jar openapi-generator-cli.jar generate ^
   -i http://pubi.org/petstore_oas3.json ^
   -g ruby ^
   -o C:\temp\petstore_api2\ruby\ ^
   -c config.json
```

Note: we manually changed the output to a different folder “petstore_api2” instead of “petstore_api”.

Here is the partial output from the console:

```
williamcheng@Williams-iMac: /tmp
➜ /tmp java -jar openapi-generator-cli.jar generate \
  -i http://petstore.swagger.io/v2/swagger.json \
  -g ruby \
  -o /var/tmp/petstore_api2/ruby/ \
  -c config.json

[main] WARN  o.o.c.ignore.CodegenIgnoreProcessor - Output directory does not exist, or is inaccessible. No file (.openapi-generator
-ignore) will be evaluated.
[main] INFO  o.o.c.languages.AbstractRubyCodegen - Hint: Environment variable 'RUBY_POST_PROCESS_FILE' (optional) not defined. E.g.
 to format the source code, please try 'export RUBY_POST_PROCESS_FILE="/usr/local/bin/rubocop -a"' (Linux/Mac)
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] WARN  o.o.codegen.utils.ModelUtils - Multiple schemas found, returning only the first one
[main] INFO  o.o.codegen.AbstractGenerator - writing file /var/tmp/petstore_api2/ruby/lib/online_petstore/models/api_response.rb
[main] INFO  o.o.codegen.AbstractGenerator - writing file /var/tmp/petstore_api2/ruby/spec/models/api_response_spec.rb
[main] INFO  o.o.codegen.AbstractGenerator - writing file /var/tmp/petstore_api2/ruby/docs/ApiResponse.md
```

As you can see, the folder name is now “online_petstore”. By inspecting the gemspec file, we can confirm all the desired changes such as version name, package name and more are now in place:

vim openapi_client.gemspec

```
$:.push File.expand_path("../lib", __FILE__)
require "openapi_client/version"

Gem::Specification.new do |s|
  s.name        = "openapi_client"
  s.version     = OpenapiClient::VERSION
  s.platform    = Gem::Platform::RUBY
  s.authors     = ["OpenAPI-Generator"]
  s.email       = ["apiteam@swagger.io"]
  s.homepage    = "https://openapi-generator.tech"
  s.summary     = "Swagger Petstore Ruby Gem"
  s.description = "This is a sample server Petstore server.  You can find out more about Swagger at [http://swagger.io](http://swag
ger.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."
  s.license     = "Unlicense"
  s.required_ruby_version = ">= 1.9"

  s.add_runtime_dependency 'typhoeus', '~> 1.0', '>= 1.0.1'
  s.add_runtime_dependency 'json', '~> 2.1', '>= 2.1.0'

  s.add_development_dependency 'rspec', '~> 3.6', '>= 3.6.0'
  s.add_development_dependency 'vcr', '~> 3.0', '>= 3.0.1'
  s.add_development_dependency 'webmock', '~> 1.24', '>= 1.24.3'
  s.add_development_dependency 'autotest', '~> 4.4', '>= 4.4.6'
  s.add_development_dependency 'autotest-rails-pure', '~> 4.1', '>= 4.1.2'
  s.add_development_dependency 'autotest-growl', '~> 0.2', '>= 0.2.16'
  s.add_development_dependency 'autotest-fsevent', '~> 0.2', '>= 0.2.12'
```

Besides the configuration file, we can also customize the Ruby SDK through the command line. For example, we can specify the Github user ID and repository ID as follows:

[Mac/Linux]

```
java -jar openapi-generator-cli.jar generate \
   -i http://pubi.org/petstore_oas3.json \
   -g ruby \
   -o /var/tmp/petstore_api2/ruby/ \
   -c config.json \
   --git-user-id openapi-generator-ebook --git-repo-id OnlinePetstore
```

[Windows]

```
java -jar openapi-generator-cli.jar generate ^
   -i http://pubi.org/petstore_oas3.json ^
   -g ruby ^
   -o C:\temp\petstore_api2\ruby\ ^
   -c config.json ^
   --git-user-id openapi-generator-ebook --git-repo-id OnlinePetstore
```

We still keep the config.json for the remaining settings and only pass the Github user ID and repository ID via the general option through command line arguments. This way we can easily change the Ruby SDK without modifying the configuration file.

Another way to customize the output is to use the “--additional-properties” command switch:

```
--additional-properties <additional properties>
    sets additional properties that can be referenced by the mustache
    templates in the format of name=value,name=value
```

Variables in mustache templates are denoted in the form of {{variable_name}}, for example {{gemVersion}} in the gemspec.mustache file[59]. Here is the command to customize the "gemVersion" and "gemName" through the command line:

[Mac/Linux]
```
java -jar openapi-generator-cli.jar generate \
   -i http://pubi.org/petstore_oas3.json \
   -g ruby \
   -o /var/tmp/petstore_api2/ruby/ \
   -c config.json \
  --additional-properties gemVersion=0.3.1,gemName=online_petstore
```

[Windows]
```
java -jar openapi-generator-cli.jar generate ^
   -i http://pubi.org/petstore_oas3.json ^
   -g ruby ^
   -o C:\temp\petstore_api2/ruby\ ^
   -c config.json ^
   --additional-properties gemVersion=0.3.1,gemName=online_petstore
```

Similar to providing the command line options directly, the "--additional-properties" allows us to customize the output with ease. Personally, I prefer using the configuration file as it is easier to keep track of changes in the configuration file to understand how the auto-generated code has evolved and changed over time.

5.4 Deploying the Ruby SDK to GitHub

As part of the customization, we set the value of "gitUserId" and "gitRepoId" to "openapi-generator-ebook" and "OnlinePetstore" respectively. To deploy the Ruby SDK to GitHub, we will need to follow the simple steps below:

1. Login to Github with credential "openapi-generator-ebook"
2. Create the repository "OnlinePetstore" as shown below:

[59] https://github.com/openapitools/openapi-generator/blob/master/modules/openapi-generator/src/main/resources/ruby-client/gemspec.mustache

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner openapi-generator-ebook / **Repository name** * OnlinePetstore

Great repository names are short and memorable. Need inspiration? How about **psychic-telegram**?

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

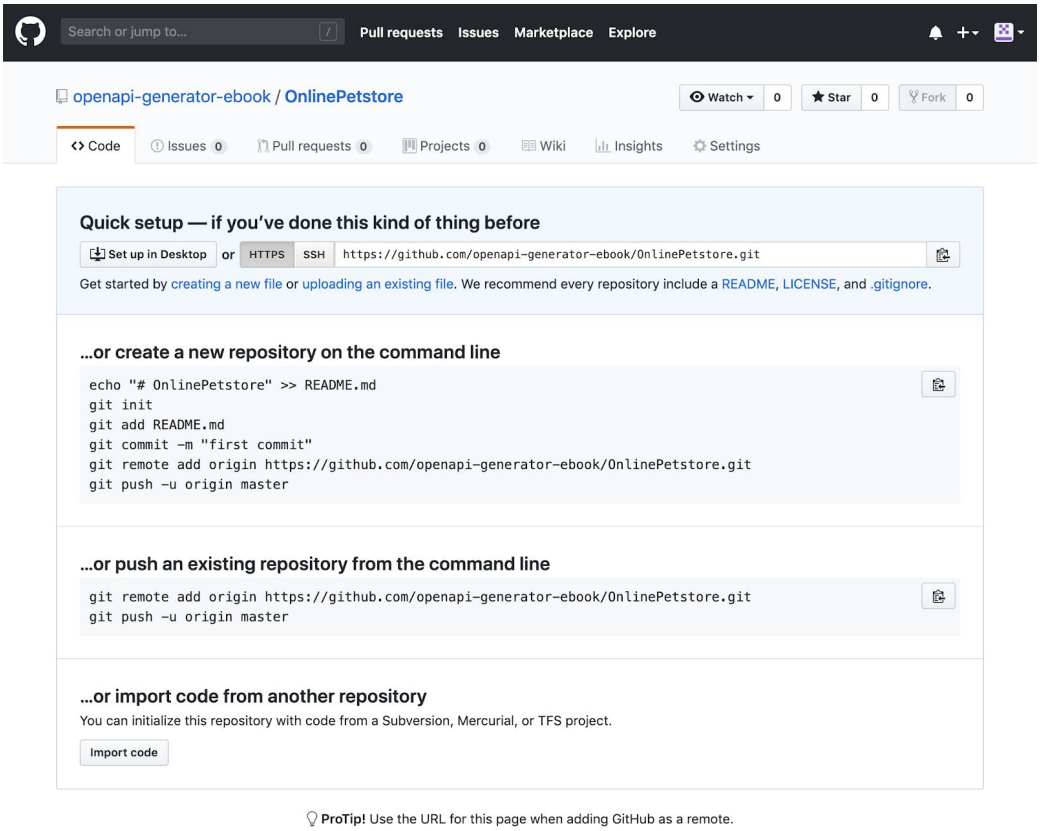
Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

3. Then Github will show you instructions to add files to the repository you just created:



4. For Mac/Linux users, you can instead go to the folder "/var/tmp/petstore_api2/ruby/" and running "/bin/sh git_push.sh" will output the following:

```
openapi@dev01:/var/tmp/petstore_api2/ruby$ /bin/sh git_push.sh
[INFO] No command line input provided. Set $git_user_id to
openapi-generator-ebook
[INFO] No command line input provided. Set $git_repo_id to OnlinePetstore
[INFO] No command line input provided. Set $release_note to Minor update
Initialized empty Git repository in /var/tmp/petstore_api2/ruby/.git/
[master (root-commit) 812e8a2] Minor update
 44 files changed, 5793 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .rspec
(... result omitted)
 create mode 100644 spec/spec_helper.rb
[INFO] $GIT_TOKEN (environment variable) is not set. Using the git
credential in your environment.
fatal: Couldn't find remote ref master
Unexpected end of command stream
Git pushing to
https://github.com/openapi-generator-ebook/OnlinePetstore.git
Username for 'https://github.com': openapi-generator-ebook
Password for 'https://openapi-generator-ebook@github.com':
 * [new branch]      master -> master
openapi@dev01:/var/tmp/petstore_api2/ruby$
```

Ref:
https://github.com/openapi-generator-ebook/OnlinePetstore/commit/812e8a297bba44b50bcd157f7bcfb8e71d66d581

The "git_push.sh" script can come in handy as part of your CI/CD (continuous integration and continuous delivery) workflow to push the updated SDK to the Github repository

5.5 Customization using the templates

So far we have explored using different options to customize the output. However, these options cannot completely customize every single line of the output.

What if we want to add the following text to every auto-generated file so that developers can more easily reach out to us in case they have any feedback on the Ruby SDK or want to report issues when using the Ruby SDK in their environments.

Please contact support@online-petstore.email if you have any feedback for us.

The only way to achieve this is to fully customize the templates. OpenAPI Generator CLI comes with the following option to provide customized templates:

```
-t <template directory>, --template-dir <template directory>
```

```
    folder containing the template files
```

The “-t” (or --template-dir) argument takes a path to the folder with the customized templates.

The default templates can be obtained from the OpenAPI Generator Github repository under the “modules/openapi-generator/src/main/resources” folder[60]. Here are the steps to use the customized templates:

[Mac/Linux]

```
mkdir templates
git clone https://github.com/openapitools/openapi-generator.git
cp -R
openapi-generator/modules/openapi-generator/src/main/resources/ruby-clien
t ./templates/
[Edit the file “templates/ruby/api_info.mustache” to insert the
customized message we want to display in the autogenerated files.]
java -jar openapi-generator-cli.jar generate \
   -i http://pubi.org/petstore_oas3.json \
   -g ruby \
   -o /var/tmp/petstore_api3/ruby/ \
   -c config.json \
   -t ./templates/ruby-client/
```

[Windows]

```
mkdir templates
git clone https://github.com/openapitools/openapi-generator.git
xcopy
openapi-generator\modules\openapi-generator\src\main\resources\ruby-clien
t templates /e /i /h
[Edit the file “templates\ruby\api_info.mustache” to insert the
customized message we want to display in the autogenerated files.]
java -jar openapi-generator-cli.jar generate ^
   -i http://pubi.org/petstore_oas3.json ^
   -g ruby ^
   -o C:\temp\petstore_api3\ruby\ ^
   -c config.json ^
   -t templates\ruby-client\
```

The file “api_info.mustache” is what we call a partial in a mustache, which is just another mustache template and it is imported by other files. Here is an example showing how it’s included in the second line of another template file (showing first 15 lines only):

```
=begin
{{> api_info}}
=end
```

60 https://github.com/openapitools/openapi-generator/tree/master/modules/openapi-generator/src/main/resources

```
require 'date'
require 'json'
require 'logger'
require 'tempfile'
require 'typhoeus'
require 'uri'

module {{moduleName}}
  class ApiClient
      # The Configuration object holding settings to be used in the API
client.
    attr_accessor :config
```

That way we can reuse common texts or codes among other files. For more information on mustache partial, please refer to the official manual[61].

To conclude, we have covered 4 different ways to customize the output via the OpenAPI Generator CLI:

1. Configuration file with “-c” switch
2. Command line switches (e.g. --git-user-id)
3. Customising the mustache variables via the “--additional-properties” switch
4. Customising the mustache templates via the “-t” switch

We recommend using the configuration file to start with and customizing the templates as the last resort if the built-in options and the “--additional-properties” switch cannot fully meet your requirement.

[61] https://mustache.github.io/mustache.5.html

6. OpenAPI Generator upgrade

OpenAPI Generator is a very active open-source project with more than 1100 contributors and hundreds of companies using it in production. Every week, there are enhancements and bug fixes merged into the main branch “master”. As of this writing, the latest stable release is 5.0.0. There are 3 important branches: master (for upcoming 5.0.1 release), 5.1.x and 6.0.x. The latest stable release is 5.0.0 and contains breaking changes compared with the previous stable version 4.3.1, which means developers who upgrade from 4.3.1 to 5.0.0 may need to manually make some minor adjustments to their program to fix issues due to breaking changes without fallbacks.

All backward-compatible enhancements and bug fixes made to the current master are synchronized (copied) to the 5.1.x and 6.0x branch, which means these branches will also include those backward-compatible changes.

6.1 Release note

Before upgrading to a newer version of OpenAPI Generator, we strongly recommend reviewing the release note first so as to better understand the impact of the upgrade. For stable versions, the detailed and informative release notes can be found on the release page of the Github repository[62].

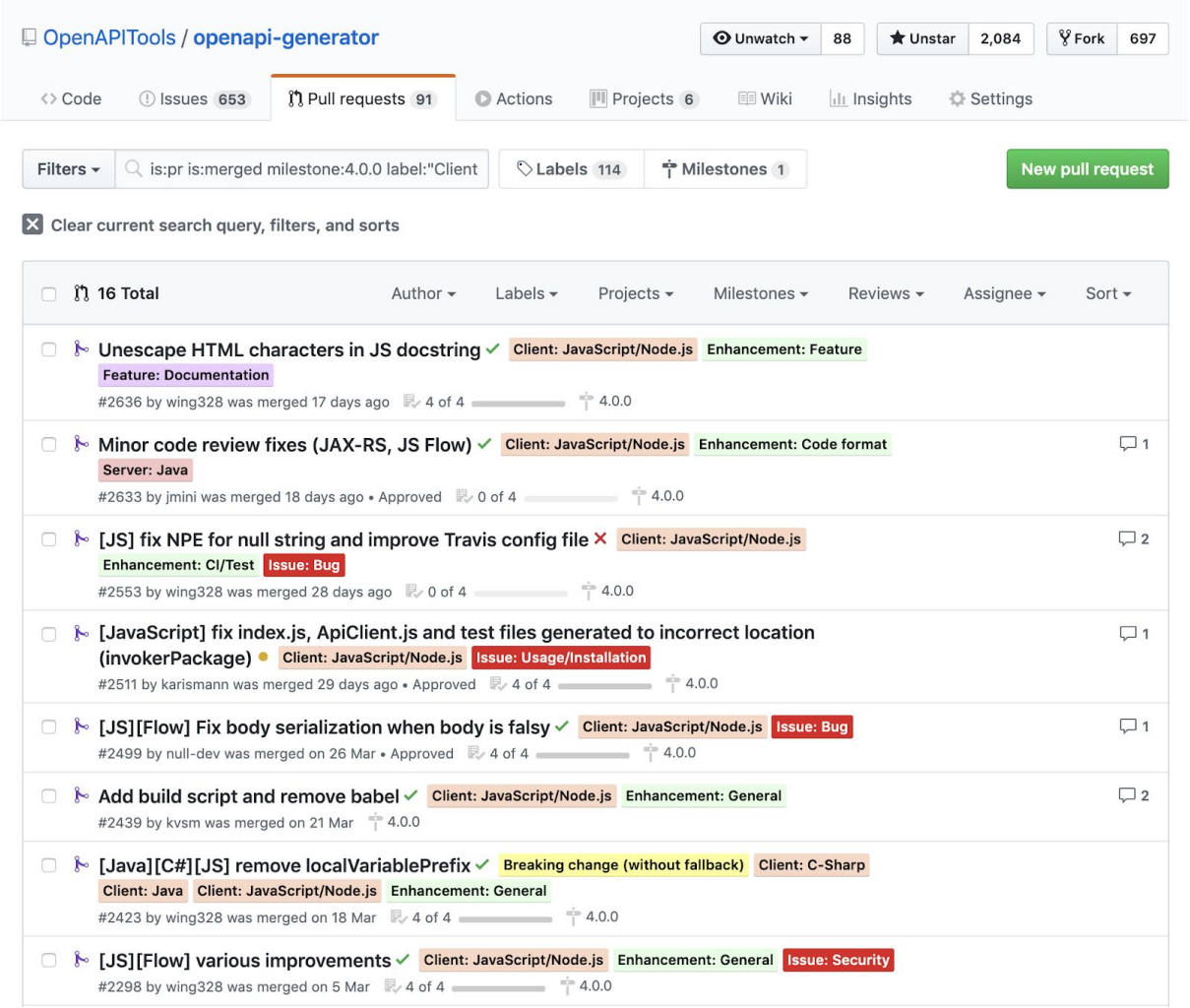
What if you want to review the changes to the stable release 4.0.0? Github provides an excellent way for users to filter changes for a particular release or category. Assuming we need to review all the breaking changes made to the JavaScript API client in the 4.0.0 release, we can easily do so by going to the pull request page and apply the following filter:

```
is:pr is:merged label:"Client: JavaScript/Node.js" milestone:4.0.0
```

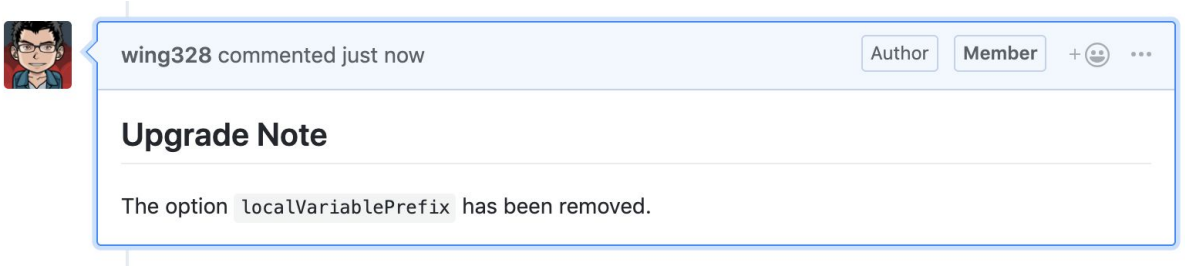
In addition to typing the search string above, we can also click on the “Labels” button to select “Client: JavaScript/Node.js” and then the “Milestones” button to select “4.0.0” to achieve the same result.

Below is what the result looks like:

[62] https://github.com/openapitools/openapi-generator/releases



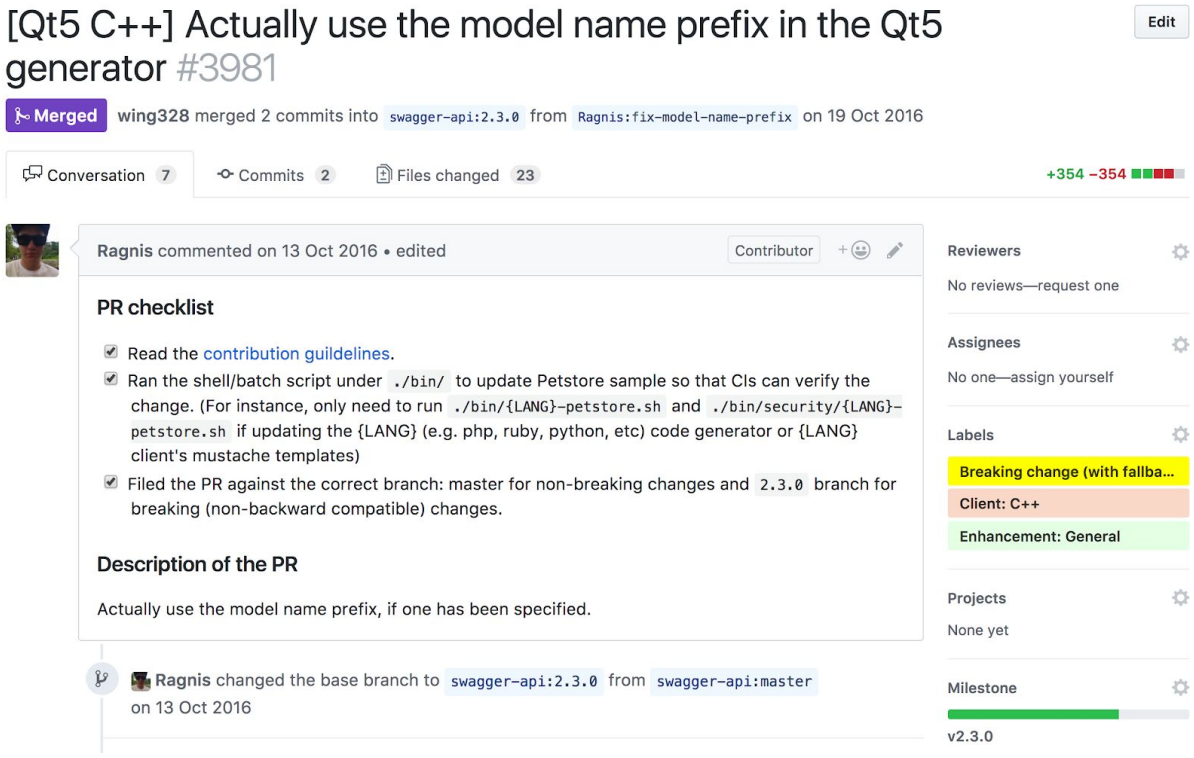
Some changes are labeled with "**Breaking changes (without fallback)**". Clicking on these changes will show more details about the breaking changes. Below shows the upgrade note for the issue in the list above:



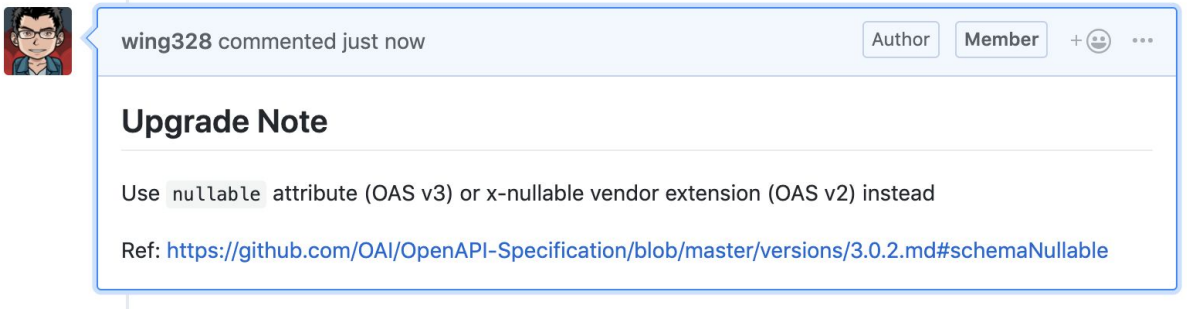
Ref:
https://github.com/OpenAPITools/openapi-generator/pull/2423#issuecomment-487370990

By reviewing the upgrade note, you will find it much easier to perform OpenAPI Generator upgrade as you now have a much better understanding of the change. If you still have questions after reviewing the upgrade note or want to share some upgrade tips for this particular change, please feel free to do so by replying to the pull request.

For changes labeled as "**Breaking changes (with fallback)**", they usually include an upgrade note on how to fallback to the previous version using command line options or other ways. Here is an example of breaking changes with fallback for the Qt5 C++ generator:



Ref: https://github.com/OpenAPITools/openapi-generator/pull/2559



Ref:
https://github.com/OpenAPITools/openapi-generator/pull/2559#issuecomment-487371277

6.2 Release schedule

There's no fixed schedule on how often we release a stable version. For patch release (e.g. 3.3.4), it usually takes 1 month. For minor releases such as 3.3.0, it may take 2 to 3 months or a year. For major release (4.0.0), it may take more than a year.

For the latest schedule, please refer to the compatibility matrix[63]:

1.1 - Compatibility

The OpenAPI Specification has undergone 3 revisions since initial creation in 2010. The openapi-generator project has the following compatibilities with the OpenAPI Specification:

OpenAPI Generator Version	Release Date	Notes
6.0.0 (upcoming major release) SNAPSHOT	Nov/Dec 2021	Minor release with breaking changes (no fallback)
5.1.0 (upcoming minor release) SNAPSHOT	Mar/Apr 2021	Minor release with breaking changes (with fallback)
5.0.1 (upcoming patch release) SNAPSHOT	Jan/Feb 2021	Patch release with enhancements, bug fixes, etc
5.0.0 (latest stable release)	21.12.2020	Major release with breaking changes (no fallback)
4.3.1	06.05.2020	Patch release (enhancements, bug fixes, etc)

OpenAPI Spec compatibility: 1.0, 1.1, 1.2, 2.0, 3.0

For old releases, please refer to the **Release** page.

63 https://github.com/openapitools/openapi-generator#compatibility

7. Common use cases and issues

7.1 Removing prefix from “operationId” (method name)

For some specifications we looked at, the ”operationId”, which is used by OpenAPI Generator to determine the API method name, is in the form of “{prefix}_{method_name}”, for example, “User_get_id”, “student_addResult”, which will be generated as “UserGetId” and “StudentAddResult” accordingly in C# API client by OpenAPI Generator. (C# method name follows the “Pascal Case” naming convention). The C# code to use these functions would look like the following:

```
student.StudentAddResult(result1);
user.UserGetId(291);
```

These methods are still functional but do not look idiomatic with the object name prefixing the method name. To remove the object prefix, we can use the “--remove-operation-id-prefix” option, when generating the API client. For example:

```
[Mac/Linux]
java -jar openapi-generator-cli.jar generate \
   -i http://pubi.org/petstore_oas3.json \
   -g csharp --remove-operation-id-prefix \
   -o /var/tmp/petstore_api/csharp/

[Windows]
java -jar openapi-generator-cli.jar generate ^
   -i http://pubi.org/petstore_oas3.json ^
   -g csharp --remove-operation-id-prefix ^
   -o C:\temp\petstore_api\csharp\
```

Then “User_”, “student_” or anything before the first underscore will be truncated to have a more idiomatic method name.

7.2 Add prefix to models

Suppose we are integrating the auto-generated Java SDK for a Bitcoin Exchange API into an existing enterprise application that already contains a lot of models defined in third-party libraries. One of the third-party models contains a model named “Component” and coincidentally our auto-generated Java SDK also contains a model with the same name. One possible way is to add a prefix to all models generated by OpenAPI Generator using the “--model-name-prefix“ option. For example:

```
[Mac/Linux]
java -jar openapi-generator-cli.jar generate \
   -i http://pubi.org/petstore_oas3.json \
   -g java --model-name-prefix Bitcoin \
```

```
    -o /var/tmp/petstore_api/java/
```

[Windows]

```
java -jar openapi-generator-cli.jar generate ^
    -i http://pubi.org/petstore_oas3.json ^
    -g java --model-name-prefix Bitcoin ^
    -o C:\temp\petstore_api\java\
```

Then all the auto-generated models will be prefixed with “Bitcoin”, for example, “BitcoinComponent”

7.3 Customize the User-Agent for easier call trace

User-Agent in the HTTP header specifies the name of the applications that make the HTTP call. It can be very useful for tracing the HTTP requests. If we need to debug an issue with the HTTP request through a proxy, one way to easily and uniquely identify the HTTP requests among many others going through the same proxy is to set a different User-Agent using “--http-user-agent”. For example, the default User-Agent for Ruby API client is “openapi-generator/#{VERSION}/ruby" in which #{VERSION} is a Ruby constant set to the version of the Ruby Gem. To set it to a different value such as “Ruby/Debug/User123”, we can run the following:

[Mac/Linux]

```
java -jar openapi-generator-cli.jar generate \
    -i http://pubi.org/petstore_oas3.json \
    -g ruby --http-user-agent “Ruby/Debug/User123” \
    -o /var/tmp/petstore_api/ruby/
```

[Windows]

```
java -jar openapi-generator-cli.jar generate ^
    -i http://pubi.org/petstore_oas3.json ^
    -g ruby --http-user-agent “Ruby/Debug/User123” ^
    -o C:\temp\petstore_api\ruby\
```

Then in the proxy or HTTP server log, one can more easily trace the request with “Ruby/Debug/User123”.

7.4 Test the API client in different deployment environments

One of the most common requirements is the ability to test the API clients with different deployment environments such as development, testing, staging before rolling out to production. There are several ways to achieve this. The simplest one is to change the OpenAPI 2.0 or 3.0 specification. Using the Petstore example, we can replace the following line:

```
  host: api.pubi.org
```

with

```
host: staging.api.pubi.org
```

and then generate a new API client with the updated specification. This works but there is overhead in maintaining multiple specifications and it is just a matter of time the specifications for different environments become out-of-sync.

Another less obvious way is to modify the host table on your machine. Here are the files storing the host tables in different operating systems:

- Linux: /etc/hosts
- Mac: /private/etc/hosts
- Windows: c:\windows\system32\drivers\etc\hosts

To map api.pubi.org to the IP address of the staging server (e.g. 192.168.8.19), we can add the following entry to the host table:

```
192.168.8.19      api.pubi.org
```

This method can be handy for testing the API clients against different deployment environments without modifying the specification but we do not recommend this approach as from time to time people forgot to remove the entry in the host table and it took them a long time to troubleshoot before they realized the API client was actually talking to a different server.

The recommended approach is to change the target host during runtime. Most API clients allow you to change the base URL (e.g. https://api.pubi.org/v1) before making HTTP calls to the API servers. Using the Ruby SDK as an example, one can change the base URL easily in the configuration object and pass it when creating a new instance of ApiClient. Here is the sample code to change the base URL with a different port for Ruby Petstore SDK:

```
config2 = Petstore::Configuration.new do |c|
  c.host = 'stagging.api.pubi.org:8080'
  c.base_path = 'v2'
end
api_client2 = Petstore::ApiClient.new(config2)
```

The configuration object also allows you to change other settings such as HTTP timeout, TLS verification and more.

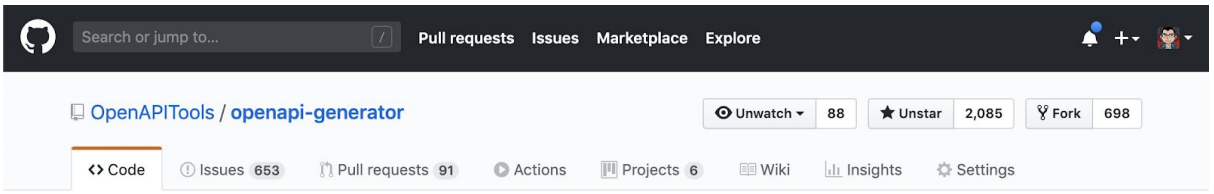
8. Contribute back

OpenAPI Generator would not have achieved its success and wide adoption today without a great community. We strongly encourage users to contribute back to the project in whatever ways they feel comfortable and here are some suggestions on how users can easily make this project even better.

8.1 Report issues

Given that OpenAPI Generator supports more than 35 server frameworks and nearly 40 API client generators, there may be some edge cases not covered by certain generators, especially the new ones, which are likely less mature. In case you find any issues or have any questions, please try the following:

1. Perform a search in Github to confirm no one has yet reported similar issues.



Tips: You may also want to try the Github advanced search as well[64]

2. Create a new issue and provide all the required details such as OpenAPI Generator version, OpenAPI/Swagger spec to reproduce the issue and more.

8.2 Help the others

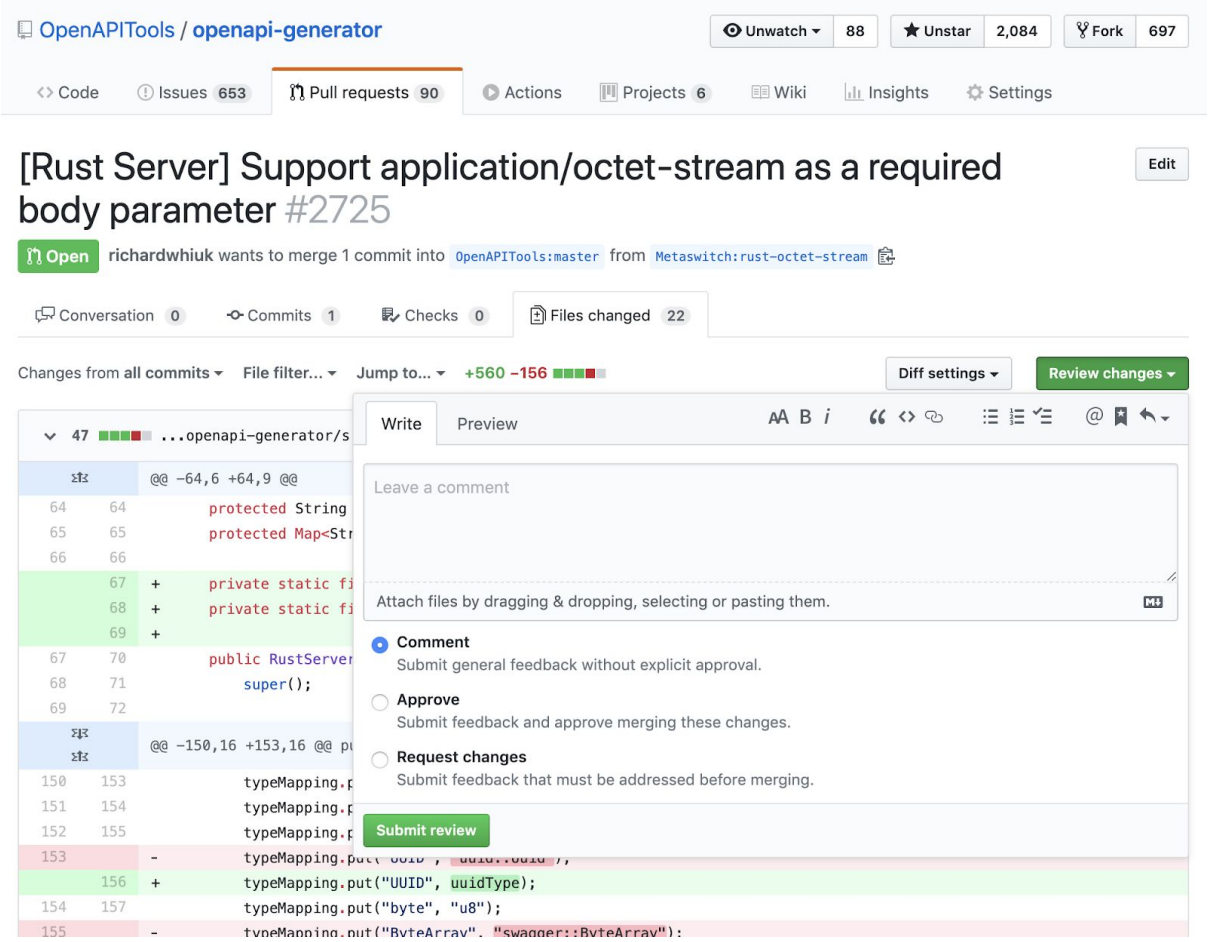
The issue tracker in the Github repository is the primary channel to report issues, provide feedback and start a discussion related to OpenAPI Generator. There are other channels such as StackOverflow[65] to discuss OpenAPI Generator but we prefer using the Github issue tracker to consolidate all the discussions into a single place so as to build a stronger and bigger community. If you have time, please help out by answering questions related to OpenAPI Generator. This will also help you to gain a better understanding of OpenAPI Generator by sharing what you know and find out OpenAPI Generator features that you have not yet discovered.

8.3 Review pull requests

64 https://github.com/search/advanced
65 https://stackoverflow.com/questions/tagged/openapi-generator

In 2010, there are more than 1500 change requests merged into OpenAPI Generator[66], which means there are more than 4 change requests merged on average per day. With so many change requests for enhancements and bug fixes, I want to take this opportunity to thank the OpenAPI Generator core team for helping out in reviewing and testing changes using their own time. To bring the project to the next level, we strongly recommend everyone in the community to help review the changes and test locally if possible. Everyone is encouraged to share their view on the change and suggest better alternatives for bug fixes or enhancements.

Github offers an easy-to-use workflow for reviewing change requests. Using https://github.com/OpenAPITools/openapi-generator/pull/2725 as an example, just click on the “Files changed” tab and then “Review changes” as shown below



You can simply make a comment on the change. If the change looks good to you, you can approve it so that the OpenAPI Generator team knows that this PR is good from your perspective. If you have better suggestions or find issues with the pull request, you can select “Request changes” and provide details on what you want to change.

66 https://github.com/OpenAPITools/openapi-generator/pulls?q=is%3Apr+is%3Amerged+merged%3A2020-01-01..2020-12-31

8.4 Enhancements or bug fixes

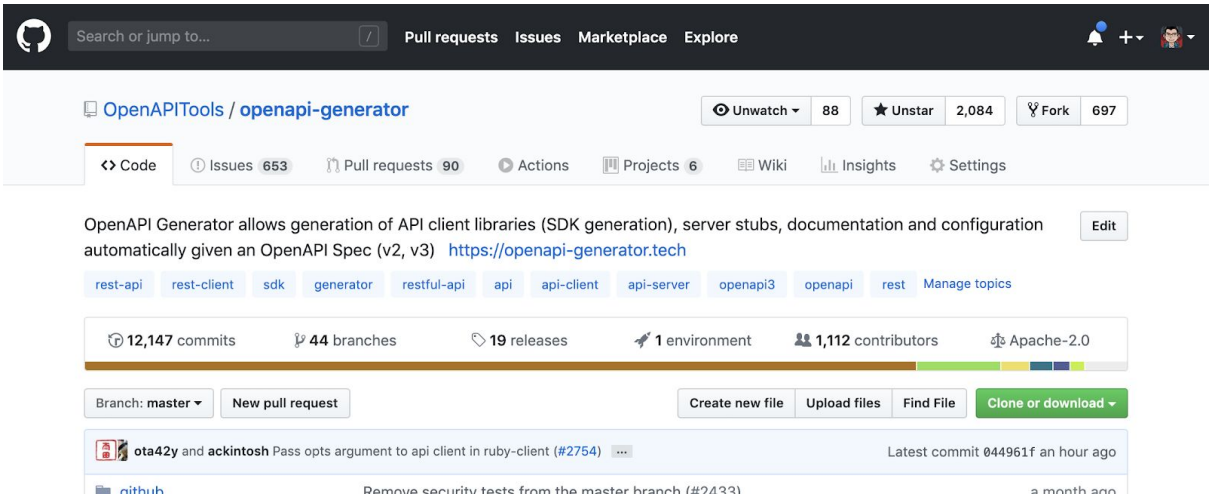
OpenAPI Generator would not have achieved its success today without the vibrant community. More than 1700 developers have contributed back various enhancements, bug fixes and new generators, and you can be one of them too. If you can think of any way to make OpenAPI Generator better, you are welcome to start or join the discussion in the issue tracker. One easy way to filter out tasks that need community contribution is to use the search filter below:

```
is:open is:issue label:"help wanted"
```

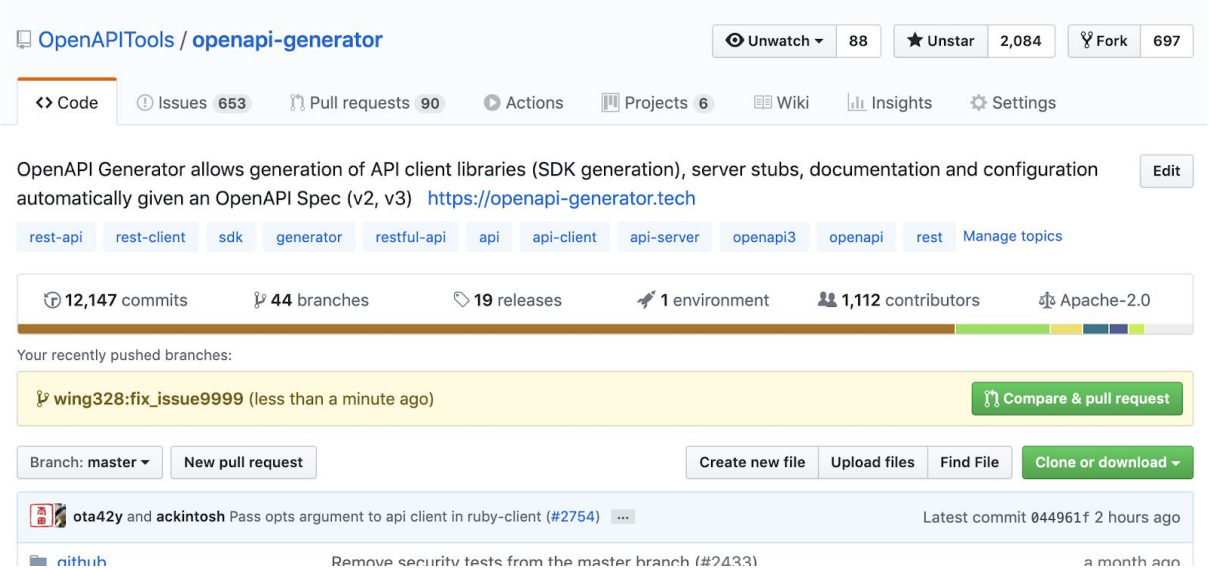
Then you can further filter tasks based on programming languages.

To file a pull request for enhancements or bug fixes, here are a few simple steps:

1. Visit https://github.com/openapitools/openapi-generator and then click on the "Fork" button in the top-right corner as shown below. Then in your local machine (Linux, Mac, Windows), open a terminal and run the following (assuming your github ID is "your_user_id")



2. git clone https://github.com/your_user_id/openapi-generator.git
3. cd openapi-generator
4. git checkout -b fix_issue9999
5. Make changes with your favorite text editor
6. git commit -a (you may need to use “git add filename” to add new files)
7. git push origin fix_issue9999
8. Visit https://github.com/openapitools/openapi-generator in the browser and click on the button to file a new PR based on fix_issue9999 as shown below



Ref: https://github.com/OpenAPITools/openapi-generator/wiki/FAQ#how-can-i-submit-a-pr-to-fix-bugs-or-make-enhancements

Contributing to OpenAPI Generator can not only enhance your software development skills but also make your resume look more attractive because there is a growing demand for IT professionals who have experience with the open-source software. Do a Google search and you will find job openings looking for candidates with experience in OpenAPI and OpenAPI Generator so we strongly recommend you add OpenAPI and OpenAPI Generator to your resume.

Reviewing other PRs is also a good way to understand how changes are made to the OpenAPI Generator. In the merged pull requests, you will find what files are modified to fix a bug or add a new feature. To find all merged pull requests related to the Python API client, we can apply the following filter:

```
is:pr is:merged label:"Client: Python"
```

If you have any questions on the merged PRs, please feel free to reply with your question and the community will try their best to help you out.

8.5 Technical committees

For those who want to participate in the future development of OpenAPI Generator, you may consider joining the technical committees, who shoulder the following responsibilities:

- Provides guidance and direction to other users
- Reviews pull requests and issues
- Improves the generator by making enhancements, fixing bugs or updating documentation
- Sets the technical direction of the generator

To join the technical committee, one must have at least 3 merged PRs for a particular generator. To apply, please refer to the latest instruction in the project's README[67].

67 https://github.com/OpenAPITools/openapi-generator#62---openapi-generator-technical-committee

9. What's next?

Congratulations! You have completed this book to gain a better understanding of OpenAPI (formerly known as Swagger) 2.0 and 3.0 specification and how to leverage OpenAPI Generator to streamline your software development experience. This is just the beginning. Both OpenAPI and OpenAPI Generator can do a lot more than you would have imagined so please do not stop here. Keep exploring the unlimited possibilities with the toolings around OpenAPI and OpenAPI Generator, which has new generators contributed by the vibrant community almost every month. We also strongly encourage OpenAPI Generator users to share their experience via blog posts, video and more. Some materials shared by the OpenAPI Generator community can be found in the project's README[68]. Please feel free to add yours with a pull request.

We sincerely hope you found this book, OpenAPI Generator and OpenAPI useful and we truly wish you all the best with your API journey.

68 https://github.com/OpenAPITools/openapi-generator#4---companiesprojects-using-openapi-generator

Appendix A

The output below is for reference only as there are new options added to OpenAPI Generator thanks to the active community.

OpenAPI Generator general options:

NAME

openapi-generator-cli generate - Generate code with the specified generator.

SYNOPSIS

openapi-generator-cli generate

[(-a <authorization> | --auth <authorization>)]
[--api-name-suffix <api name suffix>] [--api-package <api package>]
[--artifact-id <artifact id>] [--artifact-version <artifact version>]
[(-c <configuration file> | --config <configuration file>)] [--dry-run]
[(-e <templating engine> | --engine <templating engine>)]
[--enable-post-process-file]
[(-g <generator name> | --generator-name <generator name>)]
[--generate-alias-as-model] [--git-host <git host>]
[--git-repo-id <git repo id>] [--git-user-id <git user id>]
[--global-property <global properties>...] [--group-id <group id>]
[--http-user-agent <http user agent>]
[(-i <spec file> | --input-spec <spec file>)]
[--ignore-file-override <ignore file override location>]
[--import-mappings <import mappings>...]
[--instantiation-types <instantiation types>...]
[--invoker-package <invoker package>]
[--language-specific-primitives <language specific primitives>...]
[--legacy-discriminator-behavior] [--library <library>]
[--log-to-stderr] [--minimal-update]
[--model-name-prefix <model name prefix>]
[--model-name-suffix <model name suffix>]
[--model-package <model package>]
[(-o <output directory> | --output <output directory>)] [(-p <additional properties> | --additional-properties <additional properties>)...]
[--package-name <package name>] [--release-note <release note>]
[--remove-operation-id-prefix]
[--reserved-words-mappings <reserved word mappings>...]
[(-s | --skip-overwrite)] [--server-variables <server variables>...]
[--skip-validate-spec] [--strict-spec <true/false strict behavior>]
[(-t <template directory> | --template-dir <template directory>)]
[--type-mappings <type mappings>...] [(-v | --verbose)]

OPTIONS

-a <authorization>, --auth <authorization>
adds authorization headers when fetching the OpenAPI definitions remotely. Pass in a URL-encoded string of name:header with a comma separating multiple values

--api-name-suffix <api name suffix>
Suffix that will be appended to all API names ('tags'). Default: Api. e.g. Pet => PetApi. Note: Only ruby, python, jaxrs generators support this feature at the moment.

--api-package <api package>
package for generated api classes

--artifact-id <artifact id>
artifactId in generated pom.xml. This also becomes part of the generated library's filename

--artifact-version <artifact version>
artifact version in generated pom.xml. This also becomes part of the generated library's filename

-c <configuration file>, --config <configuration file>
Path to configuration file. It can be JSON or YAML. If file is JSON, the content should have the format {"optionKey":"optionValue", "optionKey1":"optionValue1"...}. If file is YAML, the content should have the format optionKey: optionValue. Supported options can be different for each language. Run config-help -g {generator name} command for language-specific config options.

--dry-run
Try things out and report on potential changes (without actually making changes).

-e <templating engine>, --engine <templating engine>
templating engine: "mustache" (default) or "handlebars" (beta)

--enable-post-process-file
Enable post-processing file using environment variables.

-g <generator name>, --generator-name <generator name>
generator to use (see list command for list)

--generate-alias-as-model
Generate model implementation for aliases to map and array schemas. An 'alias' is an array, map, or list which is defined inline in a

OpenAPI document and becomes a model in the generated code. A 'map' schema is an object that can have undeclared properties, i.e. the 'additionalproperties' attribute is set on that object. An 'array' schema is a list of sub schemas in a OAS document

--git-host <git host>
Git host, e.g. gitlab.com.

--git-repo-id <git repo id>
Git repo ID, e.g. openapi-generator.

--git-user-id <git user id>
Git user ID, e.g. openapitools.

--global-property <global properties>
sets specified global properties (previously called 'system properties') in the format of name=value,name=value (or multiple options, each with name=value)

--group-id <group id>
groupId in generated pom.xml

--http-user-agent <http user agent>
HTTP user agent, e.g. codegen_csharp_api_client, default to 'OpenAPI-Generator/{packageVersion}/{language}'

-i <spec file>, --input-spec <spec file>
location of the OpenAPI spec, as URL or file (required if not loaded via config using -c)

--ignore-file-override <ignore file override location>
Specifies an override location for the .openapi-generator-ignore file. Most useful on initial generation.

--import-mappings <import mappings>
specifies mappings between a given class and the import that should be used for that class in the format of type=import,type=import. You can also have multiple occurrences of this option.

--instantiation-types <instantiation types>
sets instantiation type mappings in the format of type=instantiatedType,type=instantiatedType.For example (in Java): array=ArrayList,map=HashMap. In other words array types will get instantiated as ArrayList in generated code. You can also have multiple occurrences of this option.

--invoker-package <invoker package>
    root package for generated code

--language-specific-primitives <language specific primitives>
    specifies additional language specific primitive types in the format
    of type1,type2,type3,type3. For example:
    String,boolean,Boolean,Double. You can also have multiple
    occurrences of this option.

--legacy-discriminator-behavior
    Set to true for generators with better support for discriminators.
    (Python, Java, Go, PowerShell, C#have this enabled by default).

--library <library>
    library template (sub-template)

--log-to-stderr
    write all log messages (not just errors) to STDOUT. Useful for
    piping the JSON output of debug options (e.g. `--global-property
    debugOperations`) to an external parser directly while testing a
    generator.

--minimal-update
    Only write output files that have changed.

--model-name-prefix <model name prefix>
    Prefix that will be prepended to all model names.

--model-name-suffix <model name suffix>
    Suffix that will be appended to all model names.

--model-package <model package>
    package for generated models

-o <output directory>, --output <output directory>
    where to write the generated files (current dir by default)

-p <additional properties>, --additional-properties <additional
properties>
    sets additional properties that can be referenced by the mustache
    templates in the format of name=value,name=value. You can also have
    multiple occurrences of this option.

--package-name <package name>
    package for generated classes (where supported)

--release-note <release note>
    Release note, default to 'Minor update'.

--remove-operation-id-prefix
    Remove prefix of operationId, e.g. config_getId => getId

--reserved-words-mappings <reserved word mappings>
    specifies how a reserved name should be escaped to. Otherwise, the
    default _<name> is used. For example id=identifier. You can also
    have multiple occurrences of this option.

-s, --skip-overwrite
    specifies if the existing files should be overwritten during the
    generation.

--server-variables <server variables>
    sets server variables overrides for spec documents which support
    variable templating of servers.

--skip-validate-spec
    Skips the default behavior of validating an input specification.

--strict-spec <true/false strict behavior>
    'MUST' and 'SHALL' wording in OpenAPI spec is strictly adhered to.
    e.g. when false, no fixes will be applied to documents which pass
    validation but don't follow the spec.

-t <template directory>, --template-dir <template directory>
    folder containing the template files

--type-mappings <type mappings>
    sets mappings between OpenAPI spec types and generated code types in
    the format of OpenAPIType=generatedType,OpenAPIType=generatedType.
    For example: array=List,map=Map,string=String. You can also have
    multiple occurrences of this option.

-v, --verbose
    verbose mode

Appendix B

Below is the change log of this eBook:

2017/11/08 – first release.

2017/11/10 – first revision (fix typos, revised wordings, etc) based on feedback from the readers.

2017/11/30 – added page breaks.

2017/12/09 – minor fix to “formData” parameter and chapter numbering.

2017/12/27 – minor fix to command line arguments.

2019/04/29 – updated with OpenAPI Generator.

2019/05/22 – minor fix to URLs.

2019/05/26 – minor change to font size of command, output, etc.

2019/07/07 – update petstore.json URL

2021/01/12 – added new section about OpenAPI specification 3.0