# A Beginner's Guide to Code Generation for REST APIs

How to use Swagger Codegen to streamline RESTful API development

William Cheng
@wing328

# Table of Contents

# 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 starts contributing to Swagger Codegen in 2014 and has been leading the project on a voluntary basis with more than 700 merged pull requests and over 1.5 million line changes. Since the inception in 2011, the project has grown to one of the most active open-source projects with more than 750 contributors and many companies, from start-ups to IT conglomerates, are using it in their technology stack 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 Swagger Codegen.

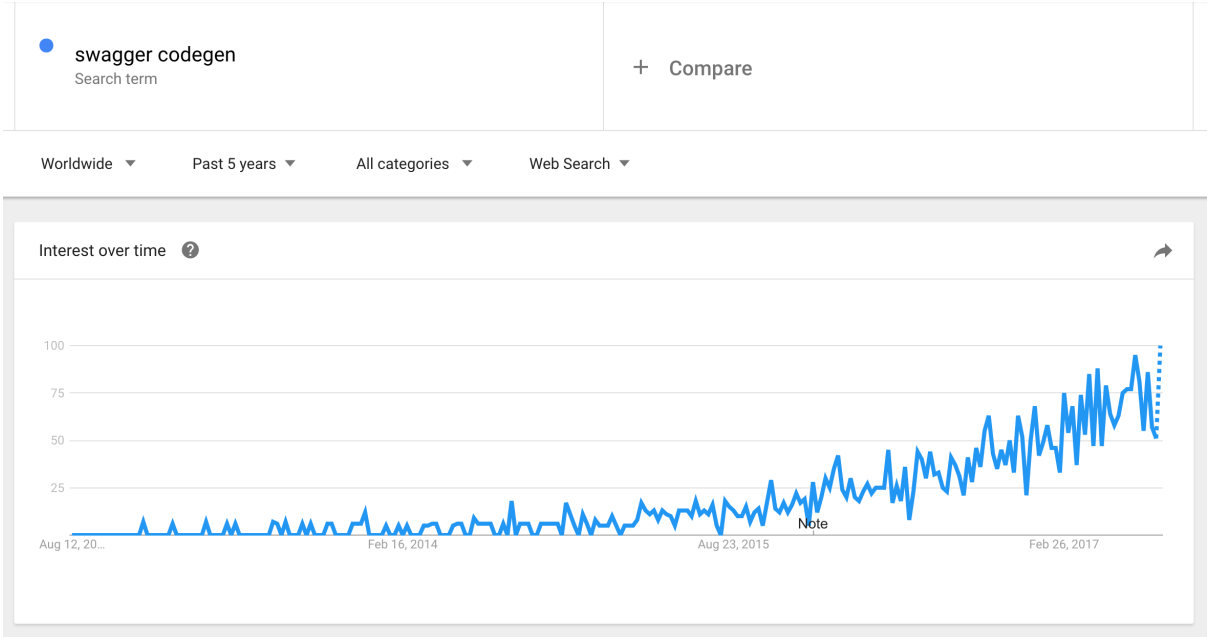
# 1. Introduction

Swagger Codegen has been gaining significant popularity since 2015 (Swagger is a registered trademark of SmartBear Software, Inc[1]). Companies, big or small, are using it in production to streamline the software development process and to reduce maintenance cost. Here are some listed companies with their market capitalization as of August 2017 using Swagger Codegen in production:

Atlassian (~US$7.8B)
Autodesk (~US$23.8B)
Box (~US$2.4B)
Cisco (~US$157B)
Cummins (~US$26.8B)
General Electric (~US$223.2B)
Hewlett Packard Enterprise (~US$29B)
IBM (~US$135.2B)
Kabuku (~JPY71.6B)
Mporium (~GBP58.1m)
Rapid7 (~US$649.9M)
Red Hat (~US$17.4B)
Royal Bank of Canada (~CAD135.2B)
SAS (~SEK6.041B)
Stingray (~CAD428.2M)
VMware (~US$38B)
Yelp (~US$3.2B)
Zalando (~EUR9.7B)

Below is a graph showing the Google Trend for “Swagger Codegen” in the past 5 years[2].

1 https://smartbear.com/
2 https://trends.google.com/trends/explore?date=today%205-y&q=swagger%20codegen




The Swagger Codegen community is also growing bigger and bigger every day with 9200+ commits and 2700+ merged pull requests, 700+ contributors who have at least 1 merged pull request.

This book aims to show IT professionals how they can leverage code generation with Swagger Codegen 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 "Swagger Codegen"?

Swagger Codegen is a completely free and open-source (Apache License v2) project to generate REST[3] API clients, server stubs and documentation based on an OpenAPI (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[4]. For the full specification of OpenAPI 2.0, please refer to the project page in Github[5].

The Swagger project was originally started by Tony Tam[6] in 2011 when he was the CTO of Wordnik. The project contains several core components:

- Swagger specification - an easy-to-understand format to describe RESTful APIs.
- Swagger UI - an interactive online explorer to allow developers to quickly play with the RESTful API in a browser.
- Swagger Editor - an online editor to write Swagger specification

3 https://en.wikipedia.org/wiki/Representational_state_transfer
4 OpenAPI members: https://www.openapis.org/membership/members
5 https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md
6 https://www.linkedin.com/in/tonytam/

- Swagger Codegen - a code generator to generate API clients, server stubs and API documentation with ease.

Later the project was acquired by SmartBear in 2015 and the Swagger specification was donated to Open API Initiative (OAI) as the foundation of OpenAPI specification.

## 1.2 Why “Swagger Codegen”?

Swagger Codegen 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 Swagger Codegen comes into play. Swagger Codegen 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 Swagger Codegen is not the only code generator available on the market so why choose Swagger Codegen? Here are a couple of points to convince you Swagger Codegen is the obvious choice:

1. Swagger Codegen 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. Swagger Codegen is used by many companies in production, from listed companies such as Atlassian, Box, Cisco, HP Enterprise, IBM, VMWare, Yelp and more to startups that have raised rounds of venture capital fundings including ActiveHours[7], DocuSign[8] and PagerDuty[9] just to name a few, which means the project itself is mature and production ready.

3. Swagger Codegen, with 5000+ stars on the Github project page[10] and 9000+ commits, has a vibrant and growing community. More than 700 software developers all over the world have filed a Pull Request, which is the standard way to propose code changes in git, to make Swagger Codegen better. You will find many activities related to Swagger Codegen in StackOverflow, Twitter, Github and other social networks. For a list of presentations, videos, tutorials and eBooks on Swagger Codegen, please refer to the README in the Github repository[11].

7 https://www.activehours.com/
8 https://www.docusign.com/
9 https://www.pagerduty.com/
10 https://github.com/swagger-api/swagger-codegen
11 https://github.com/swagger-api/swagger-codegen#presentationsvideostutorialsbooks

4. Swagger Codegen supports more than 30 API clients, 25 server stubs and API documentation. Recently, we just added support for Kotlin client generator[12] and more generators such as PowerShell[13], Lua[14], Rust[15] and Ada[16] are being worked on by the awesome community.

## 1.3 Who needs “Swagger Codegen”?

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

1. Backend developers - Swagger Codegen 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 specification file.

2. Frontend developers - Swagger Codegen 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 Swagger Codegen, 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 Swagger Codegen, one can easily generate full-featured SDKs in more than 10 programming languages to reduce the friction for onboarding new developers.

12 https://github.com/swagger-api/swagger-codegen/issues/5071
13 https://github.com/swagger-api/swagger-codegen/issues/4320
14 https://github.com/swagger-api/swagger-codegen/issues/4794
15 https://github.com/swagger-api/swagger-codegen/pull/6105
16 https://github.com/swagger-api/swagger-codegen/pull/6602

## 1.4 Who should read this book?

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

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

# 2. OpenAPI 2.0 Specification

In this chapter, we will walk you through the basic of OpenAPI 2.0 specification as it would be difficult to explain Swagger Codegen 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://petstore.swagger.io/v2/swagger.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[17]. 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[18]. 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: petstore.swagger.io
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[19].

"**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

17 https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#format
18 https://github.com/swagger-codegen-ebook/introduction/blob/master/petstore.yaml
19 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 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**"[20], 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://petstore.swagger.io/api/oauth/dialog'
    flow: implicit
    scopes:
      'write:pets': modify pets in your account
      'read:pets': read your pets
  api_key:
```

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

```
    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 Swagger Codegen 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: petstore.swagger.io
```

```
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: petstore.swagger.io
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", "form" 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[21], which means you can run it locally in your machine.

- If you are looking for ways to collaborate with your team on designing API following the contract-first approach, you should try **SwaggerHub** (https://swaggerhub.com/) by SmartBear. It allows you to work on the same specification with your teammates, to generate interactive documentation, to mock API services easily and more.

- **Apicurio Studio** (http://www.apicur.io/) by RedHat is a new contender to the OpenAPI editor space. Being free and open-source[22], it is still in beta release and for a list of upcoming features, please refer to the roadmap[23].

- If you use Eclipse for software development, you probably want to try **KaiZen OpenAPI Editor**[24] (also free, open-source), which can be easily installed from the Eclipse marketplace[25]. 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[26]:

21 https://github.com/swagger-api/swagger-editor
22 https://github.com/apicurio/apicurio-studio
23 http://www.apicur.io/roadmap/
24 https://github.com/RepreZen/KaiZen-OpenAPI-Editor
25 https://marketplace.eclipse.org/content/kaizen-openapi-editor
26 https://github.com/swagger-codegen-ebook/introduction/blob/master/invalid_petstore.json


## 2.7 Converter

Before OpenAPI specification emerges as the industry standard, there are different formats to describe RESTful APIs. RAML[27], API Blueprint[28] 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**[29], 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[30].

- **API Spec Converter** by **LucyBot**[31] (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[32], API Blueprint, Google API Discovery[33], WADL[34].

- **API-Flow** (https://github.com/luckymarmot/API-Flow) is another free and open-source alternative offered by **Paw**[35], 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

27 https://raml.org/
28 https://apiblueprint.org/
29 https://stoplight.io/
30 https://github.com/stoplightio/api-spec-converter#supported-conversions
31 http://lucybot.com/
32 https://github.com/mashery/iodocs
33 https://developers.google.com/discovery/v1/reference/apis
34 http://www.w3.org/Submission/wadl/
35 https://paw.cloud/

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[36]: Writes your swagger 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.

36 http://readme.io/

# 3. Introduction to Swagger Codegen

## 3.1 History

Before we deep dive into Swagger Codegen, here is a little bit of history and how I got involved. As mentioned, the project was started by Tony Tam 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. There is an ongoing thread[37] to discuss the possibility of switching the project from Java to other languages so feel free to join the discussion to suggest other languages.

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 Swagger Codegen 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.

## 3.2 Installation

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

### 3.2.1 Using Swagger Codegen CLI JAR (stable release)

Swagger Codegen comes with a command-line interface (CLI) under the sub-module modules/swagger-codegen-cli. The CLI is a Java JAR and therefore you will need to install Java Runtime 7 or later. The stable releases can be downloaded from MVNRespository.com or other maven repositories by searching for "Swagger Codegen". As of this writing, the latest release is 2.2.3. Here is the command to download the JAR and obtain the usage from the command prompt:

[Mac/Linux/Windows]

```
wget http://central.maven.org/maven2/io/swagger/swagger-codegen-cli/2.2.3/swagger-codegen-cli-2.2.3.jar
java -jar swagger-codegen-cli-2.2.3.jar
```

Note: For Windows users, please install PowerShell 3.0 or later in order to follow the examples in this book.

This will show the following in the output:

37 https://github.com/swagger-api/swagger-codegen/issues/3948

```
usage: swagger-codegen-cli <command> [<args>]

The most commonly used swagger-codegen-cli commands are:
    config-help   Config help for chosen lang
    generate      Generate code with chosen lang
    help          Display help information
    langs         Shows available langs
     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


See 'swagger-codegen-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 Swagger Codegen CLI version by running

```
java -jar swagger-codegen-cli-2.2.3.jar version
```

which will return *2.2.3*

We are also adding Swagger Codegen version to the auto-generated files so that users can easily tell from the files which version of Swagger Codegen was used to generate the code[38].

### 3.2.2 Using Swagger Codegen 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 2.2.3 as of this writing, and 2.3.0 to SonaType SNAPSHOT repositories[39]. Here are the URLs to the latest SNAPSHOT version of swagger-codegen-cli:

[2.2.3-SNAPSHOT] -
https://oss.sonatype.org/content/repositories/snapshots/io/swagger/swagger-codegen-cli/2.2.3-SNAPSHOT/
[2.3.0-SNAPSHOT] -
https://oss.sonatype.org/content/repositories/snapshots/io/swagger/swagger-codegen-cli/2.3.0-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.

38 https://github.com/swagger-api/swagger-codegen/issues/5998
39 https://oss.sonatype.org/content/repositories/snapshots/

### 3.2.3 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 Swagger Codegen core team and the community. This also allows you to roll back Swagger Codegen 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 7 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***[40], 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.

In the following, we will show you how to clone the Swagger Codegen 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/swagger-api/swagger-codegen.git
cd swagger-codegen
mvn clean install -DskipTests
```

40 https://git-scm.com/

***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/swagger-codegen-cli" will only build the JAR for the Swagger Codegen CLI module.***

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

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] swagger-codegen-project ........................... SUCCESS [  3.118 s]
[INFO] swagger-codegen (core library) .................... SUCCESS [ 16.141 s]
[INFO] swagger-codegen (executable) ...................... SUCCESS [  4.266 s]
[INFO] swagger-codegen (maven-plugin) .................... SUCCESS [  4.561 s]
[INFO] swagger-generator ................................. SUCCESS [  9.972 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 38.759 s
[INFO] Finished at: 2017-05-27T23:12:44+08:00
[INFO] Final Memory: 74M/1144M
[INFO] ------------------------------------------------------------------------
```

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

```
java -jar modules/swagger-codegen-cli/target/swagger-codegen-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 Swagger Codegen, please refer to "commits"[41] in the project homepage[42]. As an example, we will checkout the commit 999f901d2bdcc83b220f5c038c8352479112c8fb[43]:

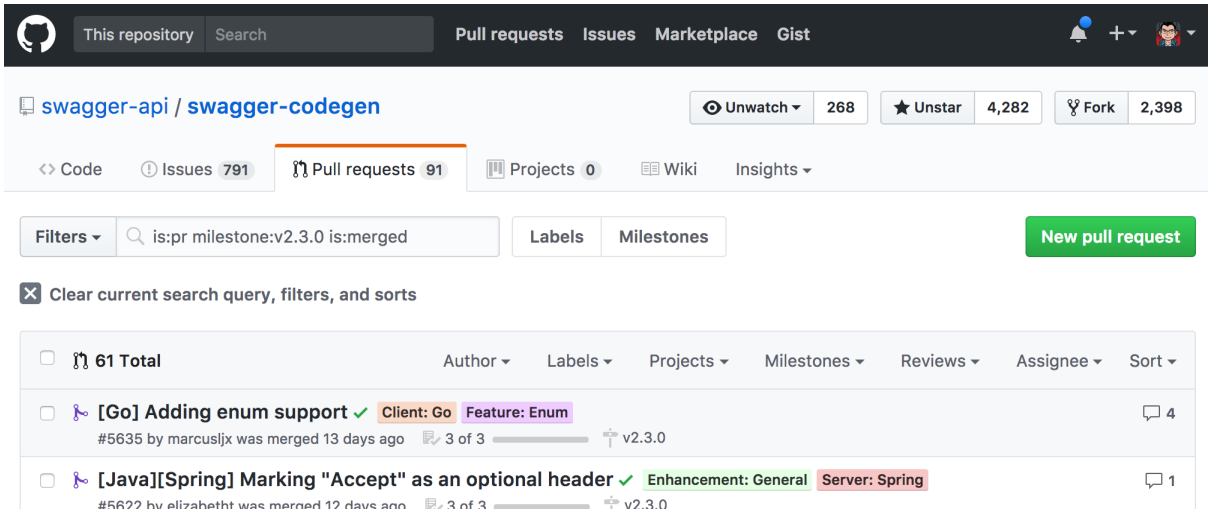
```
git clone https://github.com/swagger-api/swagger-codegen.git
cd swagger-codegen
git checkout 999f901d2bdcc83b220f5c038c8352479112c8fb
mvn clean install -DskipTests
```

*Tips: instead of using 999f901d2bdcc83b220f5c038c8352479112c8fb, you can also use its short form 999f901: git checkout 999f901*

41 https://github.com/swagger-api/swagger-codegen/commits/master
42 https://github.com/swagger-api/swagger-codegen
43 http://bit.ly/2yNzhxk

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 Swagger Codegen.

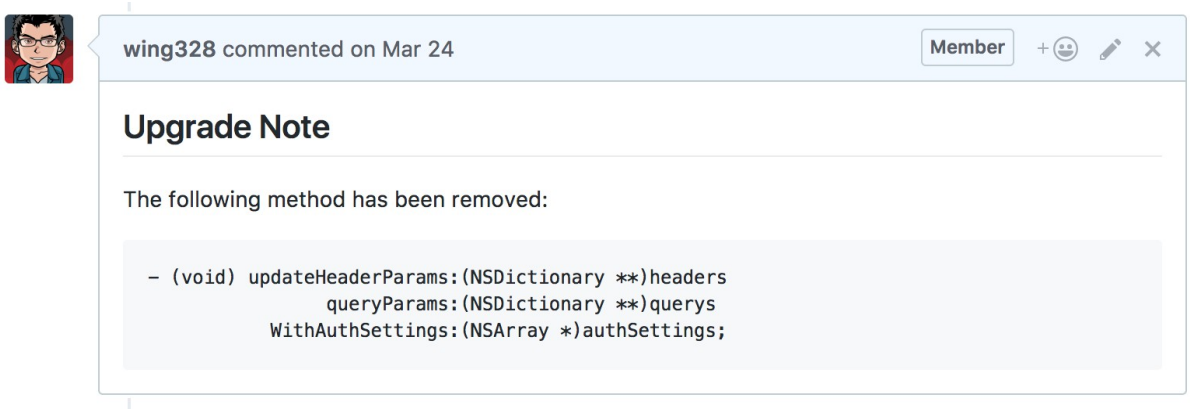
The latest stable release is 2.2.3, which means the enhancements, bug fixes will be backward compatible with the previous patch release 2.2.2. The current master is 2.3.0, which contains breaking changes. The release note in the release page highlights all the important changes[44]. To get a list of enhancements or bug fixes merged into 2.3.0 branch, you can filter the pull requests with "***is:pr milestone:v2.3.0 is:merged***" and here is the result:




https://github.com/swagger-api/swagger-codegen/pulls?utf8=%E2%9C%93&q=is%3Apr%20milestone%3Av2.3.0%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 Swagger Codegen.*

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




44 https://github.com/swagger-api/swagger-codegen/releases

Ref: https://github.com/swagger-api/swagger-codegen/pull/3798#issuecomment-288790226

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[45] a Github account in order to leave a comment.

### 3.2.4 Online generator (https://generator.swagger.io)

If you do not want to install any of the dependencies or having troubles with the installation, you can still use Swagger Codegen with the online generator: https://generator.swagger.io. All you need to do is the "curl"[46] 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://petstore.swagger.io/v2/swagger.json":

```
curl -X POST -H "content-type:application/json" -d
'{"swaggerUrl":"http://petstore.swagger.io/v2/swagger.json"}'
https://generator.swagger.io/api/gen/clients/ruby
```

And the result will look like the following:

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

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

***IMPORTANT**: https://generator.swagger.io usually runs **the latest stable version of Swagger Codegen**, which is 2.2.3 as of this writing. To enjoy the latest enhancement and bug fixes, you will need to build and run the generator locally as described above.*

Instead of using curl or other HTTP tools to submit requests to https://generator.swagger.io, you can also use Swagger Editor, which is a web-based GUI to edit OpenAPI Specification. One drawback is that it does not allow you to provide any options at all to customize the output. In other words, it always generates codes using the default setting, for example, using "SwaggerClient" as the package name. There's an ongoing discussion[47] about adding a menu in Swagger Editor to allow users more easily customizing the auto-generated code.

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.

45 https://github.com/join
46 https://curl.haxx.se/
47 https://github.com/swagger-api/swagger-editor/issues/713

At this point, we are able to use Swagger Codegen 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 Swagger Codegen generator locally on any machine you want. The Swagger Codegen generator project lives under the sub-module “modules/swagger-generator/”[48]. Here are step by step instructions to build the generator and run it locally.

```
git clone https://github.com/swagger-api/swagger-codegen.git
cd swagger-codegen
mvn clean install -DskipTests
cd models/swagger-generator
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 Swagger Codegen generator, we can rerun the curl command with a different host:

```
curl -X POST -H "content-type:application/json" -d
'{"swaggerUrl":"http://petstore.swagger.io/v2/swagger.json"}'
https://localhost:8080/api/gen/clients/ruby
```

Then it should show similar results.

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

### 3.2.5 Docker

If you have Docker[49] installed, you can start using Swagger Codegen right away. Most server stub generators in Swagger Codegen 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 machine (VM), Docker allows the containers to share the same system kernel to 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[50]. (For the following examples, we are using Docker version 17.09.0-ce, build afdb6d4)

48 https://github.com/swagger-api/swagger-codegen/tree/master/modules/swagger-generator
49 https://www.docker.com/
50 https://docs.docker.com/get-started/

The easiest way to use Swagger Codegen with Docker is to use the "swagger-codegen-cli" image: https://hub.docker.com/r/swaggerapi/swagger-codegen-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 swaggerapi/swagger-codegen-cli generate \
    -i http://petstore.swagger.io/v2/swagger.json \
    -l ruby \
    -o /local/out/ruby
```

and it should output the following:

```
$ docker run --rm -v ${PWD}:/local swaggerapi/swagger-codegen-cli generate \
    -i http://petstore.swagger.io/v2/swagger.json \
    -l ruby \
    -o /local/out/ruby
Unable to find image 'swaggerapi/swagger-codegen-cli:latest' locally
latest: Pulling from swaggerapi/swagger-codegen-cli
709515475419: Pull complete
38a1c0aaa6fd: Pull complete
cd134db5e982: Pull complete
7ca04cccce43: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:07807d55a33a7ffc56f858e1ab5ce569177b57e2b3233aa1f4765e91b25fd0aa
Status: Downloaded newer image for swaggerapi/swagger-codegen-cli:latest
[main] INFO io.swagger.parser.Swagger20Parser - reading from
http://petstore.swagger.io/v2/swagger.json
[main] WARN io.swagger.codegen.ignore.CodegenIgnoreProcessor - Output directory
does not exist, or is inaccessible. No file (.swager-codegen-ignore) will be
evaluated.
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/lib/swagger_client/models/api_response.rb
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/spec/models/api_response_spec.rb
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/docs/ApiResponse.md
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/lib/swagger_client/models/category.rb
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/spec/models/category_spec.rb
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/docs/Category.md
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/lib/swagger_client/models/order.rb
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/spec/models/order_spec.rb
[main] INFO io.swagger.codegen.AbstractGenerator - writing file
/local/out/ruby/docs/Order.md
(result omitted...)
```

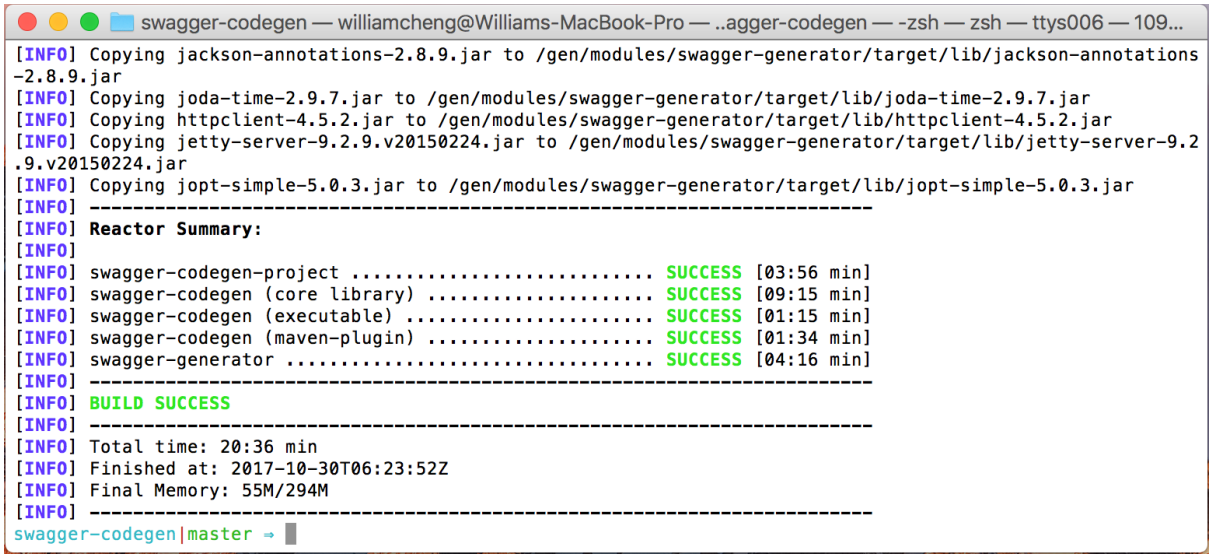
Docker tried to find the "swaggerapi/swagger-codegen-cli" image locally but couldn't and then automatically pull the latest from https://hub.docker.com. After finishing the setup, it generated the Ruby API client for the Pet Store API specification. The Ruby client can be found in /tmp/out/ruby.

What if you want to play with Swagger Codegen 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/swagger-api/swagger-codegen
cd swagger-codegen
./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 Swagger Codegen 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 Swagger Codegen 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 “langs”

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

To run the Ruby Petstore sample script, you will need to call the “bin/ruby-petstore.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/ruby-petstore.sh
```

Or you can “generate” Ruby API client directly

```
./run-in-docker.sh generate \
 -i http://petstore.swagger.io/v2/swagger.json \
 -l ruby -o /gen/out/ruby-petstore -DgemVersion=1.2.3
```

Another Docker image worth mentioning is “swagger-generator”[51], which allows you to fire up an instance of Swagger generator 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 swagger-generator if not found in the local cache:

```
docker run -e "GENERATOR_HOST=http://localhost:8080" -p 8080:8080 -d
swaggerapi/swagger-generator
```

51 https://hub.docker.com/r/swaggerapi/swagger-generator/

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
'{"swaggerUrl":"http://petstore.swagger.io/v2/swagger.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:8080/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-22e900baef61 >
result.zip
```

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:

```
swagger-codegen|master⚡ ⇒ docker ps
CONTAINER ID        IMAGE                             COMMAND                  CREATED             STAT
271fd93c3b16        swaggerapi/swagger-generator      "java -jar /genera..."   3 hours ago         Up 3
swagger-codegen|master⚡ ⇒ docker stop 271fd93c3b16 && docker rm 271fd93c3b16
271fd93c3b16
271fd93c3b16
swagger-codegen|master⚡ ⇒
```

Docker support in Swagger Codegen can come in handy if you are already familiar with Docker.

### 3.2.6 Maven/Gradle plug-in

You can also integrate Swagger Codegen into your existing workflow if you are using Maven or Gradle. The full documentation on the Swagger Codegen Maven plug-in can be found in https://github.com/swagger-api/swagger-codegen/tree/master/modules/swagger-codegen-maven-plugin. Here is a sample pom.xml to generate Ruby API client for Pet Store API specification using Swagger Codegen 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>io.swagger.Petstore</groupId>
  <artifactId>ruby-demo</artifactId>
  <version>1.0-SNAPSHOT</version>
```

```
  <build>
    <plugins>
      <!-- activate the plugin -->
      <plugin>
        <groupId>io.swagger</groupId>
        <artifactId>swagger-codegen-maven-plugin</artifactId>
        <version>2.2.3</version>
        <executions>
          <execution>
            <goals>
              <goal>generate</goal>
            </goals>
            <configuration>
              <!-- specify the location of the spec -->
              <inputSpec>http://petstore.swagger.io/v2/swagger.json</inputSpec>
              <!-- target to generate ruby client code -->
              <language>ruby</language>
              <!-- general option -->
              <gitUserId>swagger-codegen-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/swagger-codegen-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 Swagger Codegen 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/swagger/”

There are other third-party Maven plugins such as the one by Zalando[52] that you may want to give it a try as well.

The Swagger Codegen project does not come with any official Gradle plug-in but here are some developed and maintained by the vibrant Swagger Codegen community:

https://github.com/int128/gradle-swagger-generator-plugin by 岩田 英丈[53]
https://github.com/thebignet/swagger-codegen-gradle-plugin by Jean Detoeuf[54]

52 https://github.com/zalando-stups/swagger-codegen-tooling
53 https://int128.github.io/
54 https://github.com/thebignet

# 4. 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 been matured over the years with the help of contributors and the Swagger Codegen core team to add various options to customize the output so as to meet different requirements.

## 4.1 The Basic

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

```
java -jar modules/swagger-codegen-cli/target/swagger-codegen-cli.jar langs
```

(or without "**langs**" as "**langs**" is the default option)

and you will get the following:

```
Available languages: [android, aspnet5, aspnetcore, async-scala, bash, cwiki,
csharp, cpprest, dart, elixir, flash, python-flask, go, groovy, java, jaxrs, jaxrs-
cxf-client, jaxrs-cxf, jaxrs-resteasy, jaxrs-resteasy-eap, jaxrs-spec, jaxrs-cxf-
cdi, inflector, javascript, javascript-closure-angular, jmeter, nancyfx, nodejs-
server, objc, perl, php, python, qt5cpp, ruby, scala, scalatra, finch, silex-PHP,
sinatra, rails5, slim, spring, dynamic-html, html, html2, swagger, swagger-yaml,
swift, swift3, tizen, typescript-angular2, typescript-angular, typescript-node,
typescript-fetch, akka-scala, CsharpDotNet2, clojure, haskell, lumen, go-server,
erlang-server, undertow, msf4j, ze-ph]
```

which contains more than 65 languages (or generators).

Apex, a strongly-typed, object-oriented programming language developed by Salesforces, was just added to Swagger Codegen by the community on 2017/05/19 via the pull request https://github.com/swagger-api/swagger-codegen/pull/5669, which is the 3000th pull request submitted to Swagger Codegen. 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 Swagger Codegen version, you may also find the "**validate**" option useful. As the name implies, the "**validate**" option is for validating the input OpenAPI 2.0 specification. Here is an example:

55 http://github.com/swagger-api/swagger-codegen/issues
56 https://github.com/swagger-api/swagger-codegen/issues/new

```
$ java -jar swagger-codegen-cli.jar validate -i
https://raw.githubusercontent.com/swagger-codegen-
ebook/introduction/master/invalid_petstore.json
```

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

```
Validating spec file (https://raw.githubusercontent.com/swagger-codegen-
ebook/introduction/master/invalid_petstore.json)
attribute paths.'/pet/{petId}'(get).[petId].type is missing
Exception in thread "main" io.swagger.codegen.cmd.ValidateException
       at io.swagger.codegen.cmd.Validate.run(Validate.java:34)
       at io.swagger.codegen.SwaggerCodegen.main(SwaggerCodegen.java:35)
```

*Tips: Swagger Codegen leverages another Swagger 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 swagger-codegen-cli.jar he*lp generate and you will find the output in Appendix A.

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 Swagger Codegen.

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

```
$ java -jar swagger-codegen-cli.jar config-help -l python

CONFIG OPTIONS
       packageName
           python package name (convention: snake_case). (Default: swagger_client)

       packageVersion
           python package version. (Default: 1.0.0)

       packageUrl
           python package URL.

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

       hideGenerationTimestamp
           hides the timestamp when files were generated (Default: true)
```

57 https://github.com/swagger-api/swagger-parser

For Python API client generator, there are 5 options available at the moment while other generators may have more.

## 4.2 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://petstore.swagger.io/v2/swagger.json.

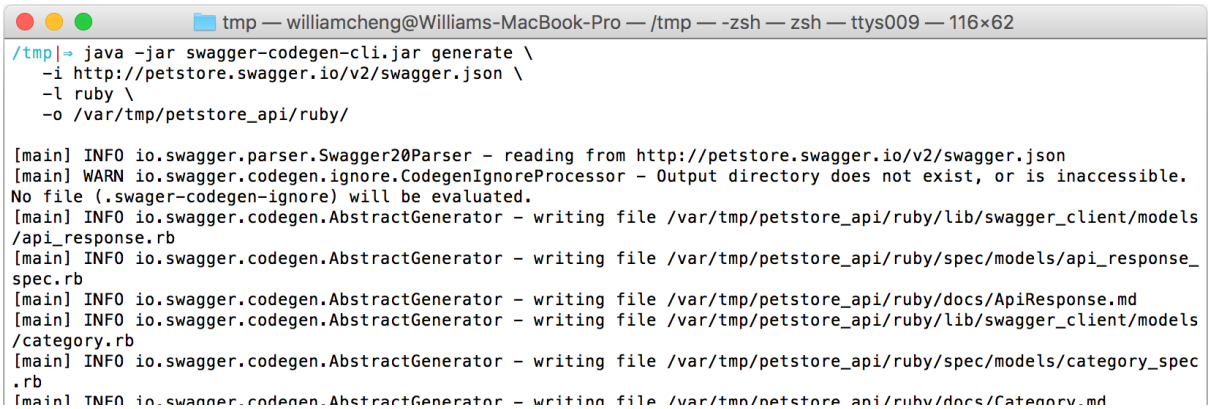
We will start with the following command without any options:

[Mac/Linux]
```
java -jar swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l ruby \
   -o /var/tmp/petstore_api/ruby/
```

[Windows]
```
java -jar swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l 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.

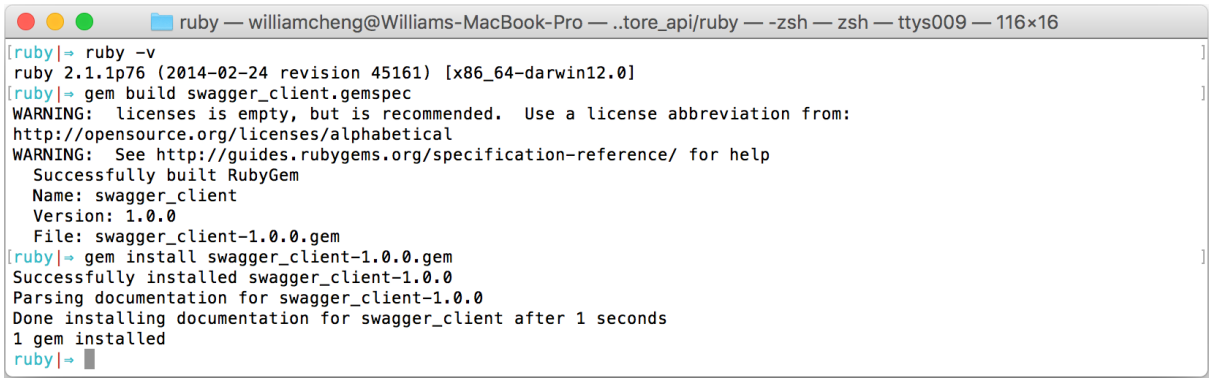
This will give you a Ruby SDK for Pet Store API using the default values and here is console output (partial):



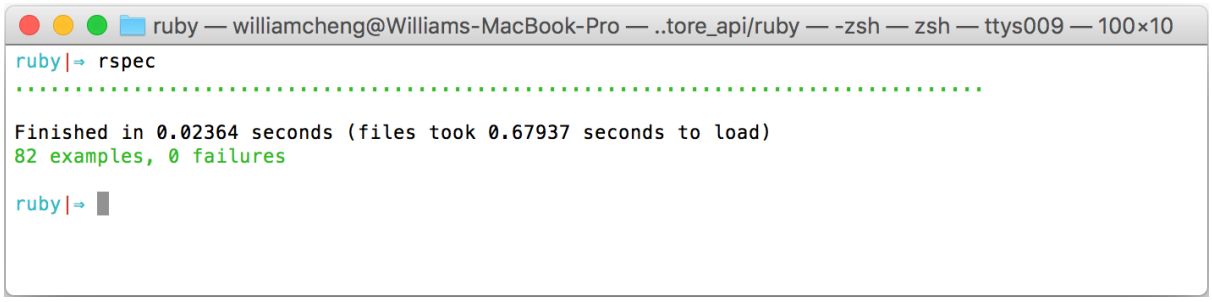

This is definitely a good start as we now have a working Ruby SDK. We can follow the instruction in the README.md to build the gem and install it locally by running

```
gem build swagger_client.spec
gem install ./swager_client.gem
```

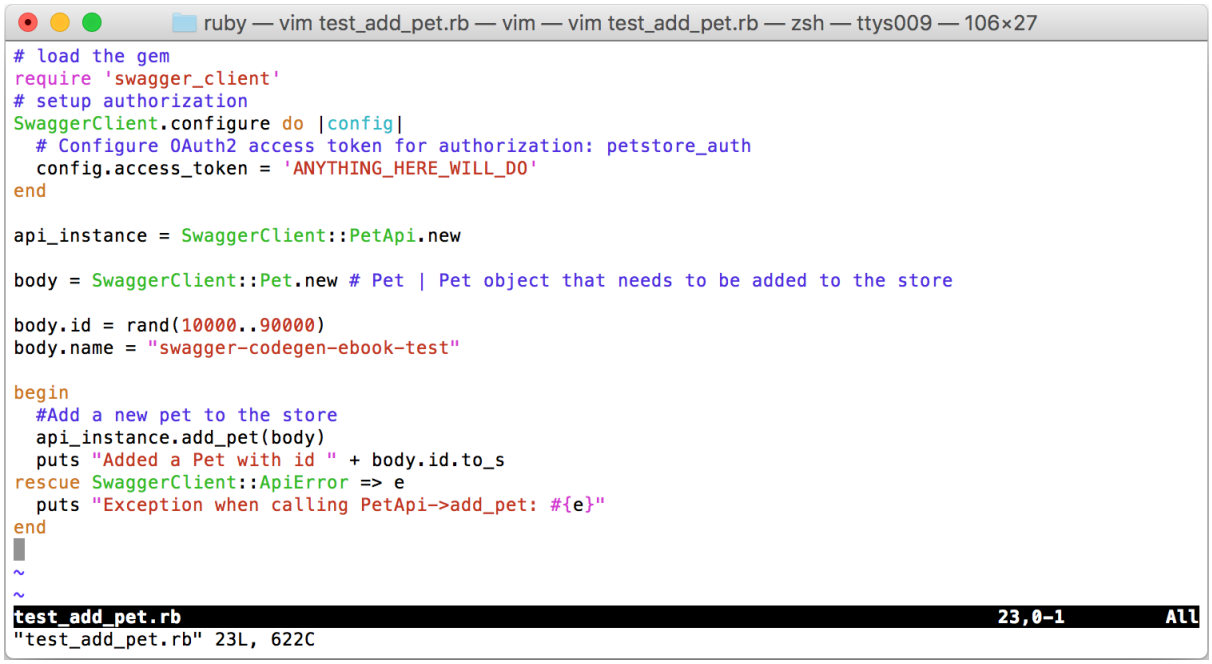
If the installation succeeds, you will see the following output:




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 autogenerated 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 “swagger-codegen-ebook-test” to the online pet store.




Ref: https://github.com/swagger-codegen-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”:

```
ruby — vim test_get_pet_by_id.rb — vim — vim test_get_pet_by_id.rb — zsh — ttys009 — 100×32
# load the gem
require 'swagger_client'
# setup authorization
SwaggerClient.configure do |config|
  # Configure API key authorization: api_key
  config.api_key['api_key'] = 'ANYTHING_WILL_DO'
  # Uncomment the following line to set a prefix for the API key, e.g. 'Bearer' (defaults to nil)
  #config.api_key_prefix['api_key'] = 'Bearer'
end

api_instance = SwaggerClient::PetApi.new

pet_id = 789 # Integer | ID of pet to return

if ARGV[0] then
  pet_id = ARGV[0]
else
  puts "No Pet ID found in the command line argument. Using default: " + pet_id.to_s
end
begin
  #Find pet by ID
  result = api_instance.get_pet_by_id(pet_id)
  p result
rescue SwaggerClient::ApiError => e
  puts "Exception when calling PetApi->get_pet_by_id: #{e}"
end
~
~
~
test_get_pet_by_id.rb                                          20,0-1          All
"test_get_pet_by_id.rb" 27L, 741C
```

Ref: [https://github.com/swagger-codegen-ebook/introduction/blob/master/test_get_pet_by_id.rb](https://github.com/swagger-codegen-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:

```
ruby — williamcheng@Williams-MacBook-Pro — ..tore_api/ruby — -zsh — zsh — ttys009 — 100×8
[ruby|⇒ ruby test_get_pet_by_id.rb                                                    ]
No Pet ID found in the command line argument. Using default: 789
Exception when calling PetApi->get_pet_by_id: Not Found
[ruby|⇒ ruby test_get_pet_by_id.rb 75482                                              ]
#<SwaggerClient::Pet:0x00000101109080 @id=75482, @name="swagger-codegen-ebook-test", @photo_urls=[],
 @tags=[]>
ruby|⇒
```

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 “SwaggerClient”? The answer is Yes. In the next section, we will explore different ways to customize the Ruby SDK.

## 4.3 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 “swagger-codegen-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 swagger-codegen-cli.jar config-help -l ruby

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

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

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

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

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

       gemVersion
           gem version. (Default: 1.0.0)

       gemLicense
           gem license.  (Default: proprietary)

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

       gemHomepage
           gem homepage.  (Default: http://swagger.io)

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

       gemDescription
           gem description.  (Default: This gem maps to a swagger API)

       gemAuthor
           gem author (only one is supported).

       gemAuthorEmail
           gem author email (only one is supported).

       hideGenerationTimestamp
```

```
          hides the timestamp when files were generated (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": "swagger-codegen-ebook",
    "gitRepoId": "OnlinePetstore"
}
```

Ref: https://github.com/swagger-codegen-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 -l {lang} command for language specific config options.
```

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

[Mac/Linux]

```
java -jar swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l ruby \
   -o /var/tmp/petstore_api2/ruby/ \
   -c config.json
```

[Windows]

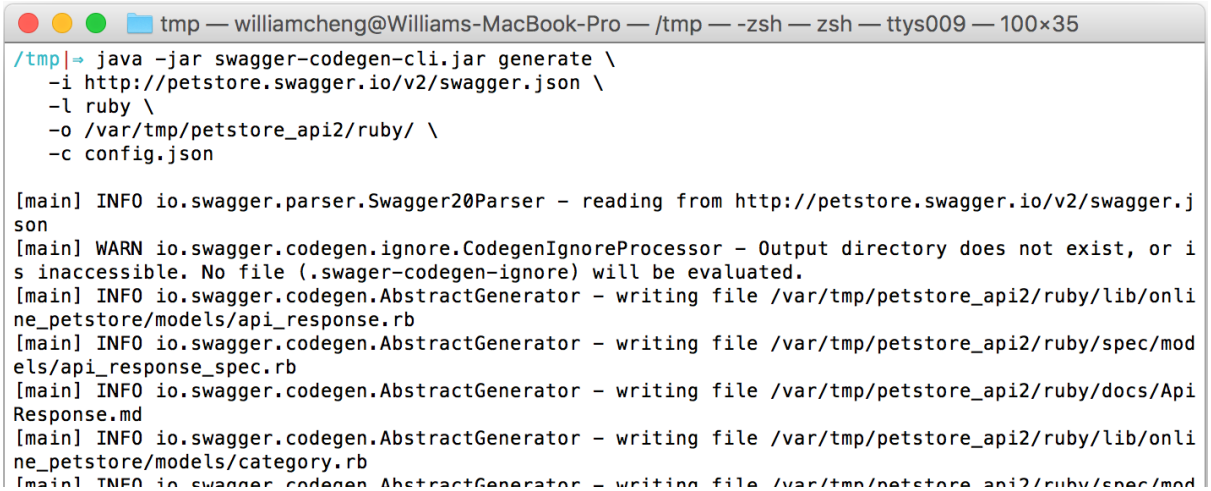
```
java -jar swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l ruby ^
```

58 https://github.com/swagger-api/swagger-codegen/issues/new

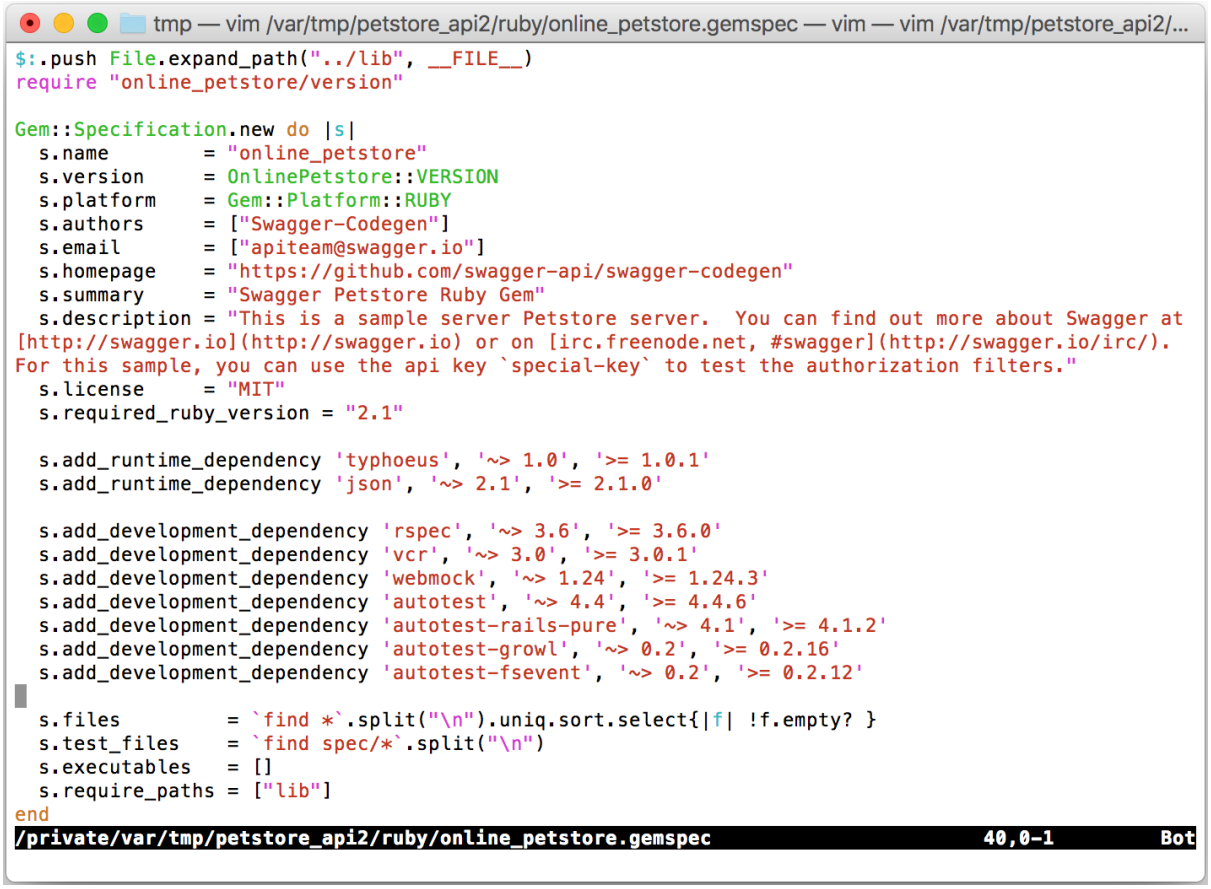
```
  -o C:\temp\petstore_api2\ruby\ ^
  -c config.json
```

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

Here is the partial output from the console:




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:




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 swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l ruby \
   -o /var/tmp/petstore_api2/ruby/ \
   -c config.json \
   --git-user-id swagger-codegen-ebook --git-repo-id OnlinePetstore
```

[Windows]

```
java -jar swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l ruby ^
   -o C:\temp\petstore_api2\ruby\ ^
   -c config.json ^
   --git-user-id swagger-codegen-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 swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l ruby \
   -o /var/tmp/petstore_api2/ruby/ \
   -c config.json \
  --additional-properties gemVersion=0.3.1,gemName=online_petstore
```

[Windows]

```
java -jar swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l ruby ^
   -o C:\temp\petstore_api2/ruby\ ^
   -c config.json ^
   --additional-properties gemVersion=0.3.1,gemName=online_petstore
```

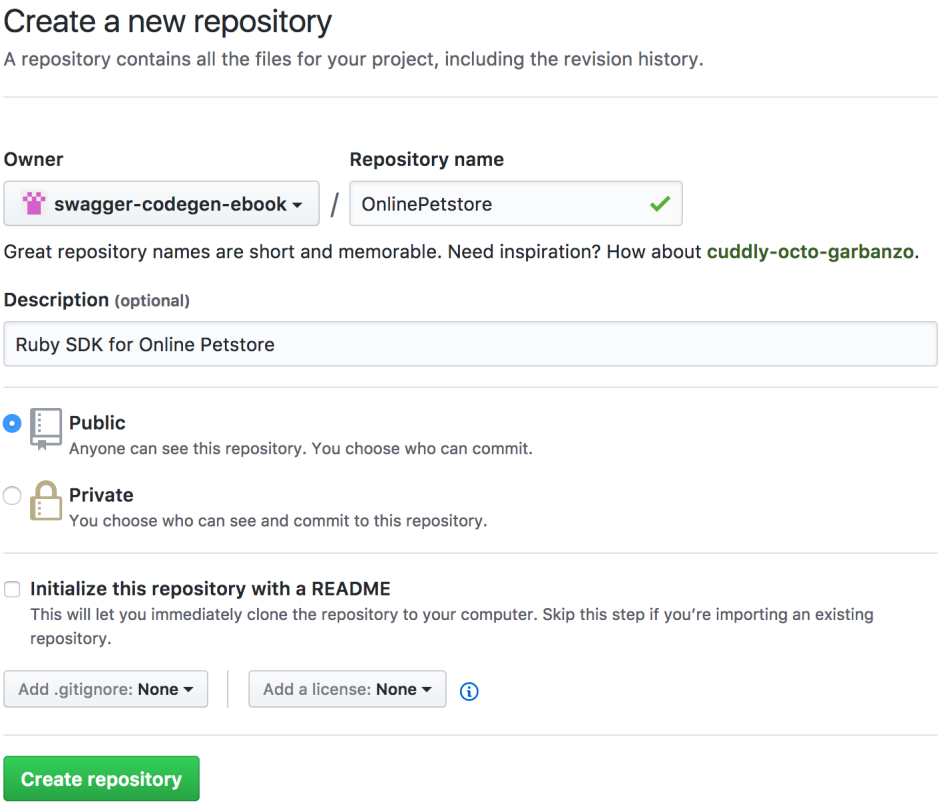
Similar to providing the command line options directly, the "--additional-properties" allows us to customize the output with ease. Personally, I prefer using the configuration file as it is easier to keep track of changes in the configuration file to understand how the auto-generated code has evolved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

## 4.4 Deploying the Ruby SDK to GitHub

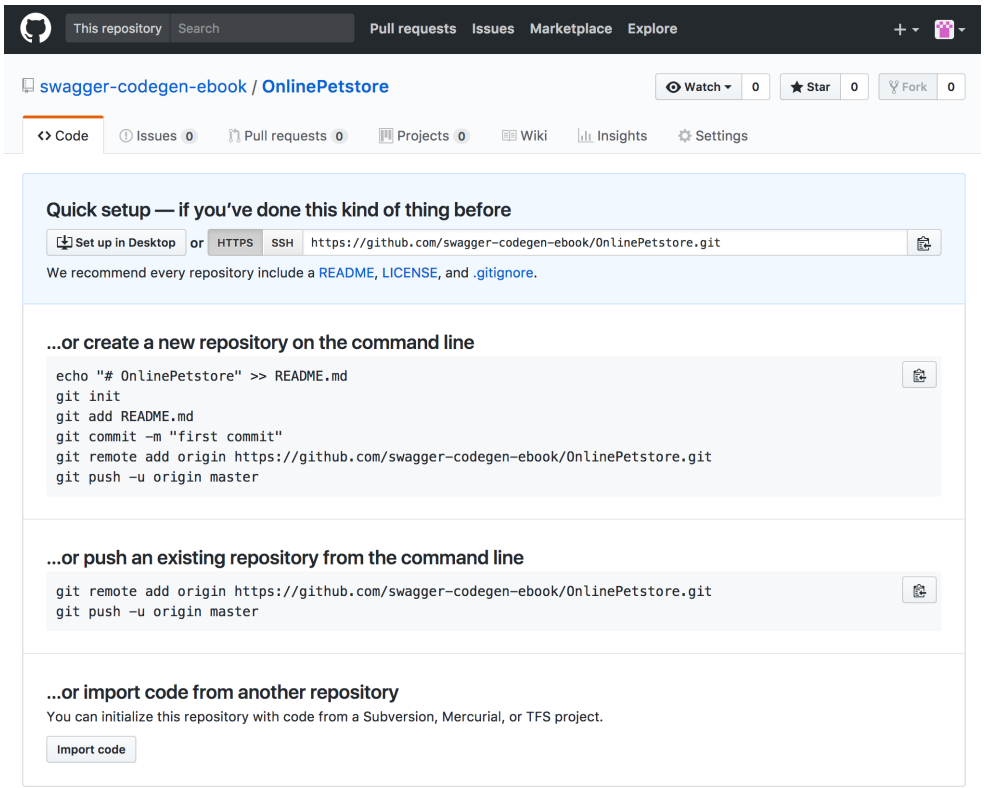
As part of the customization, we set the value of "gitUserId" and "gitRepoId" to "swagger-codegen-ebook" and "OnlinePetstore" respectively. To deploy the Ruby SDK to GitHub, we will need to follow the simple steps below:

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

59 https://github.com/swagger-api/swagger-codegen/blob/master/modules/swagger-codegen/src/main/resources/ruby/gemspec.mustache




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:

```
swagger@dev01:/var/tmp/petstore_api2/ruby$ /bin/sh git_push.sh
[INFO] No command line input provided. Set $git_user_id to swagger-codegen-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] $GIT_TOKEN (environment variable) is not set. Using the git credential in
your environment.
fatal: Couldn't find remote ref master
Unexpected end of command stream
Git pushing to https://github.com/swagger-codegen-ebook/OnlinePetstore.git
Username for 'https://github.com': swagger-codegen-ebook
Password for 'https://swagger-codegen-ebook@github.com':
 * [new branch]      master -> master
swagger@dev01:/var/tmp/petstore_api2/ruby$
```

Ref: [https://github.com/swagger-codegen-ebook/OnlinePetstore/commit/812e8a297bba44b50bcd157f7bcfb8e71d66d581](https://github.com/swagger-codegen-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

## 4.5 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 files 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. Swagger Codegen 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 Swagger Codegen Github repository under the "modules/swagger-codegen/src/main/resources" folder[60]. Here are the steps to use the customized templates:

[Mac/Linux]

```
mkdir templates
git clone https://github.com/swagger-api/swagger-codegen.git
cp -R swagger-codegen/modules/swagger-codegen/src/main/resources/ruby ./templates/
[Edit the file "templates/ruby/api_info.mustache" to insert the customized message
we want to display in the autogenerated files.]
java -jar swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l ruby \
   -o /var/tmp/petstore_api3/ruby/ \
   -c config.json \
   -t ./templates/ruby/
```

[Windows]

```
mkdir templates
git clone https://github.com/swagger-api/swagger-codegen.git
xcopy swagger-codegen\modules\swagger-codegen\src\main\resources\ruby 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 swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l ruby ^
   -o C:\temp\petstore_api3\ruby\ ^
   -c config.json ^
   -t templates\ruby\
```

The file "api_info.mustache" is what we call a partial in 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

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
```

60 https://github.com/swagger-api/swagger-codegen/tree/master/modules/swagger-codegen/src/main/resources

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 Swagger Codegen 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

# 5. Swagger Codegen upgrade

Swagger Codegen is a very active open-source project with more than 700+ 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 2.2.3. There are 2 important branches: master and 3.0.0. The current master will be released as 2.3.0 and contains breaking changes compared with the latest stable version 2.2.3, which means developers who upgrade from 2.2.3 to 2.3.0 are likely required to manually make some minor adjustments to their program to fix compilation errors.

All backward-compatible enhancements and bug fixes made to the current master are synchronized (copied) to the 3.0.0 branch, which means 3.0.0 branch will also include those backward-compatible changes.

## 5.1 Release note

Before upgrading to a newer version of Swagger Codegen, 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].

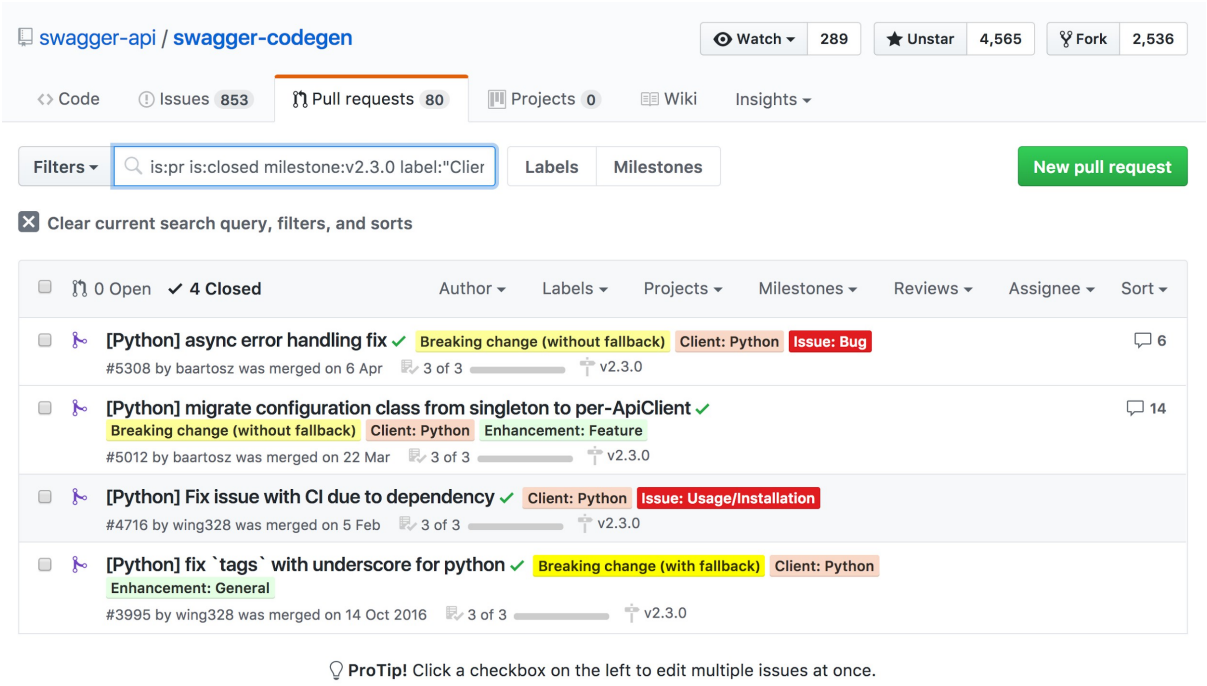
What if you want to review the changes to the latest master (2.3.0), which has not yet been released? Github provides an excellent way for users to filter changes for a particular release or category. Assuming we need to review all the breaking changes made to the Python API client in the latest master (2.3.0), we can easily do so by going to the pull request page and apply the following filter:

```
is:pr is:closed milestone:v2.3.0 label:"Client: Python"
```

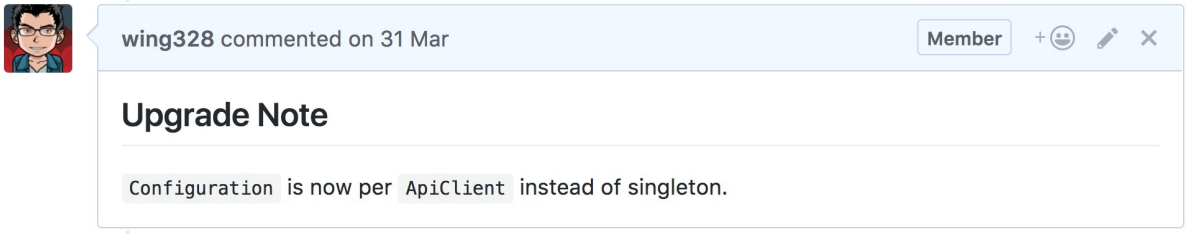
In addition to typing the search string above, we can also click on the “Labels” button to select “Client: Python” and then the “Milestones” button to select “v2.3.0” to achieve the same result.

Below is what the result looks like:

62 https://github.com/swagger-api/swagger-codegen/releases



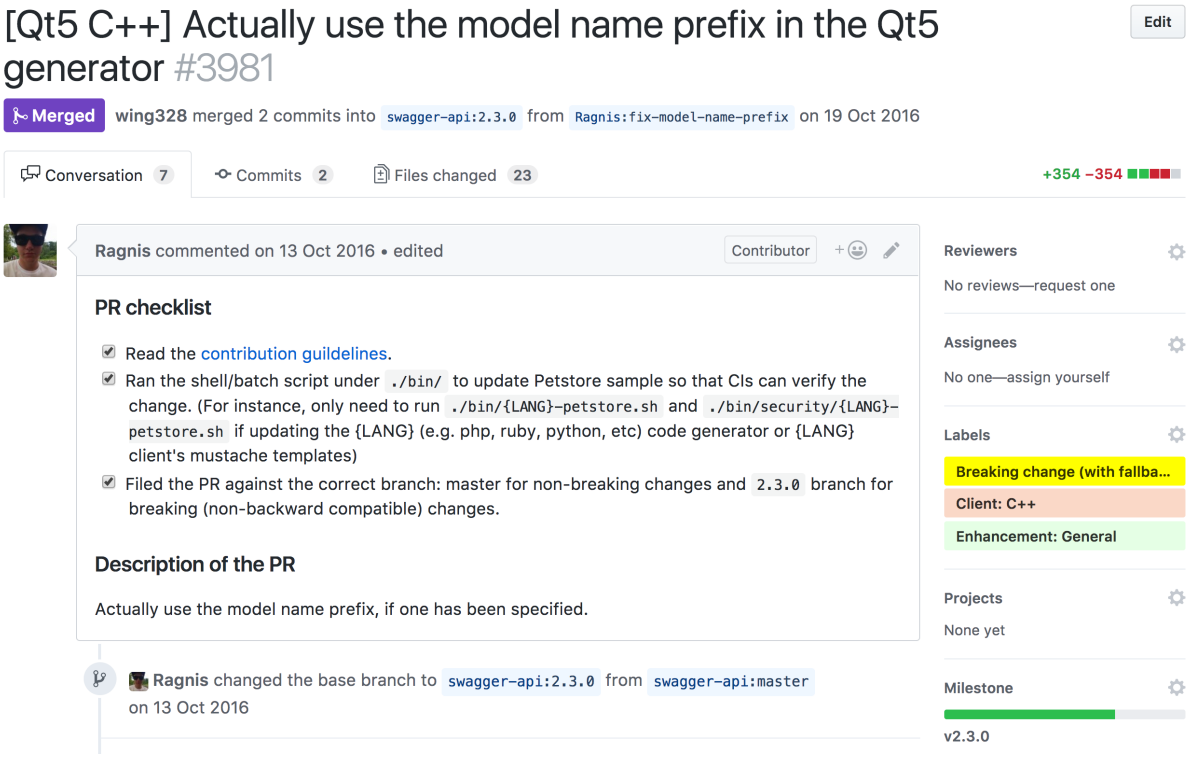

Some changes are labeled with "**Breaking changes (without fallback)**". Clicking on these changes will show more details about the breaking changes. Below shows the upgrade note for the second issue in the list above:




Ref: https://github.com/swagger-api/swagger-codegen/pull/5012#issuecomment-290646137

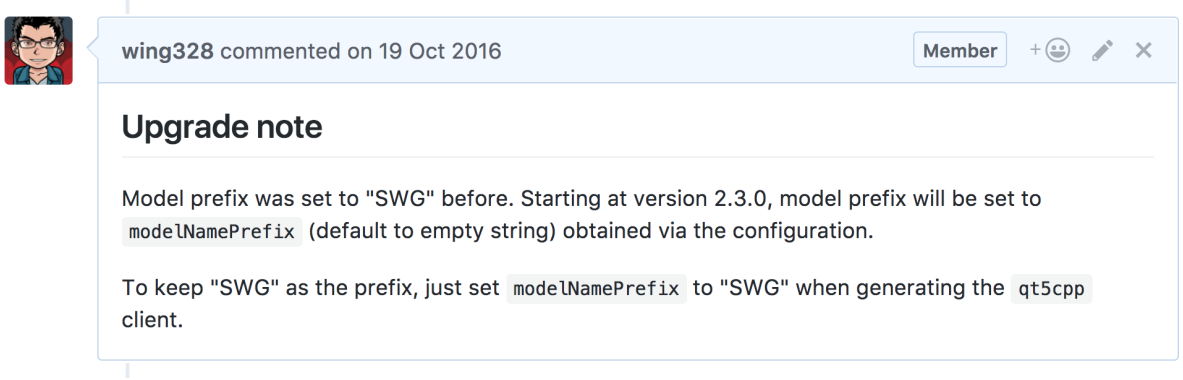
By reviewing the upgrade note, you will find it much easier to perform Swagger Codegen upgrade as you now have a much better understanding of the change. If you still have questions after reviewing the upgrade note or want to share some upgrade tips for this particular change, please feel free to do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

For changes labeled as "**Breaking changes (without fallback)**", they usually include 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/swagger-api/swagger-codegen/pull/3981

The change will apply the model prefix provided by the users when generating models. To fallback with “SWG” as the prefix, one can simply set “modelNamePrefix” to “SWG” as mentioned in the upgrade note shown below:




Ref: https://github.com/swagger-api/swagger-codegen/pull/3981#issuecomment-254843334

## 5.2 Release schedule

There’s no fixed schedule on how often we release a stable version. For patch release (e.g. 2.2.3), it usually takes 3 to 6 months. For minor release such as 2.3.0, it may take at least 6 months or a year. For major release (3.0.0), which targets OpenAPI 3.0 specification (released on July 21st, 2017), may need even more time as it relies on dependencies such as Swagger Parser to support OpenAPI specification 3.0.

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

## Compatibility

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

| Swagger Codegen Version | Release Date | OpenAPI Spec compatibility | Notes |
|---|---|---|---|
| 3.0.0 (upcoming major release) SNAPSHOT | TBD | 1.0, 1.1, 1.2, 2.0, 3.0 | Major release with breaking changes |
| 2.3.0 (current master, upcoming minor release) SNAPSHOT | Jul/Aug 2017 | 1.0, 1.1, 1.2, 2.0 | Minor release with breaking changes |
| 2.2.3 (**current stable**) | 2017-07-15 | 1.0, 1.1, 1.2, 2.0 | tag v2.2.3 |
| 2.2.2 | 2017-03-01 | 1.0, 1.1, 1.2, 2.0 | tag v2.2.2 |
| 2.2.1 | 2016-08-07 | 1.0, 1.1, 1.2, 2.0 | tag v2.2.1 |
| 2.1.6 | 2016-04-06 | 1.0, 1.1, 1.2, 2.0 | tag v2.1.6 |
| 2.0.17 | 2014-08-22 | 1.1, 1.2 | tag v2.0.17 |
| 1.0.4 | 2012-04-12 | 1.0, 1.1 | tag v1.0.4 |

63 https://github.com/swagger-api/swagger-codegen#compatibility

# 6. Common use cases and issues

## 6.1 Removing prefix from “operationId” (method name)

For some specifications we looked at, the ”operationId”, which is used by Swagger Codegen 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 Swagger Codegen. (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 swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l csharp --remove-operation-id-prefix \
   -o /var/tmp/petstore_api/csharp/
```

[Windows]

```
java -jar swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l 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.

## 6.2 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 Swagger Codegen using the “--model-name-prefix“ option. For example:

[Mac/Linux]

```
java -jar swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l java --model-name-prefix Bitcoin \
   -o /var/tmp/petstore_api/java/
```

[Windows]
```
java -jar swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l 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”

## 6.3 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 “Swagger-Codegen/#{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 swagger-codegen-cli.jar generate \
   -i http://petstore.swagger.io/v2/swagger.json \
   -l ruby --http-user-agent “Ruby/Debug/User123” \
   -o /var/tmp/petstore_api/ruby/
```

[Windows]
```
java -jar swagger-codegen-cli.jar generate ^
   -i http://petstore.swagger.io/v2/swagger.json ^
   -l 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”.

## 6.4 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 specification. Using the Petstore example, we can replace the following line:

```
host: petstore.swagger.io
```

with

```
host: staging.petstore.swagger.io
```

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 petstore.swagger.io 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 petstore.swagger.io
```

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://petstore.swagger.io/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.petstore.swagger.io: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.

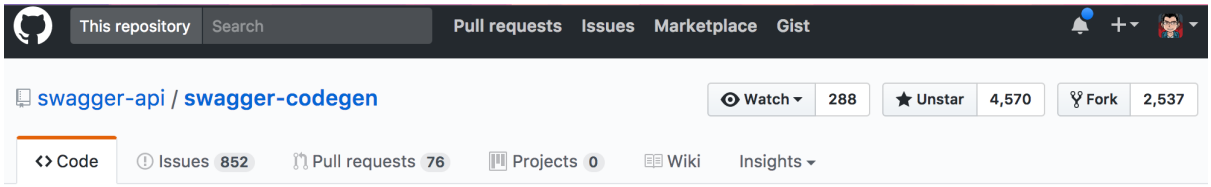
# 7. Contribute back

Swagger Codegen 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.

## 7.1 Report issues

Given that Swagger Codegen supports more than 25 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 Swagger Codegen version, OpenAPI/Swagger spec to reproduce the issue and more.

## 7.2 Help the others

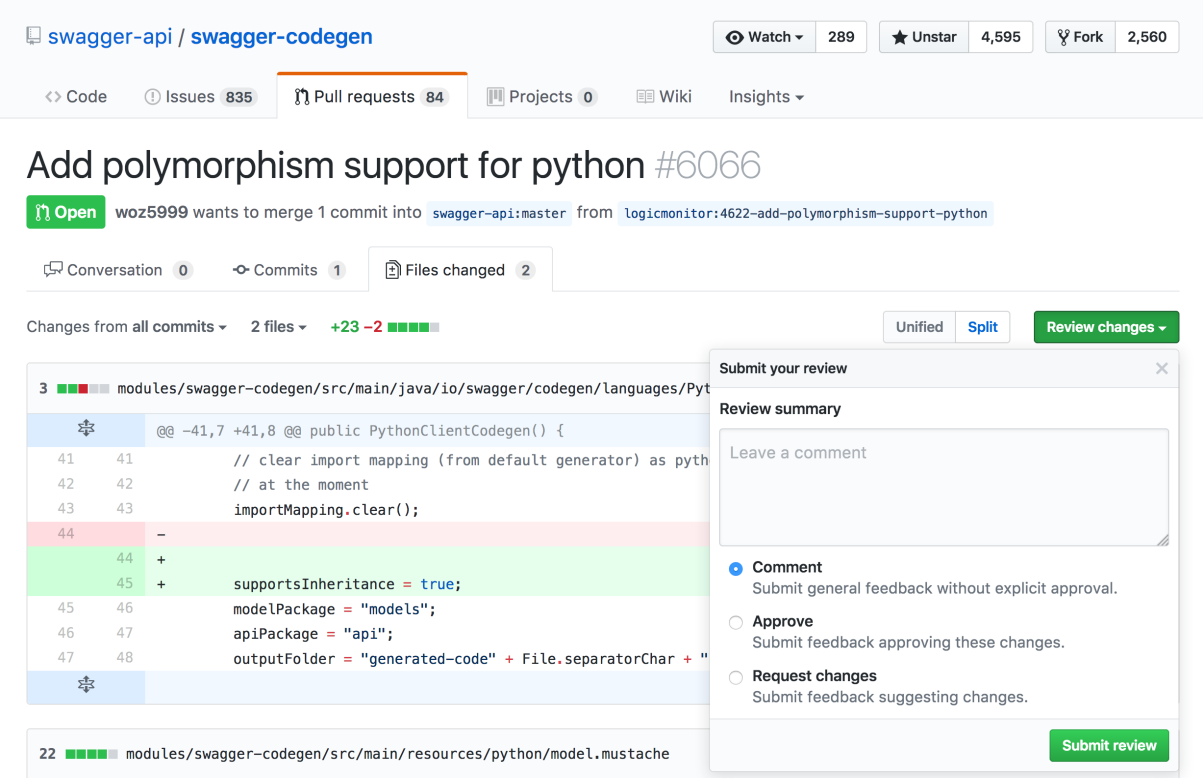
The issue tracker in the Github repository is the primary channel to report issues, provide feedbacks and start a discussion related to Swagger Codegen. There are other channels such as Google Groups[65], StackOverflow[66] to discuss Swagger Codegen but we prefer using the Github issue tracker to consolidate 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 Swagger Codegen. This will also help you to gain a better understanding of Swagger Codegen by sharing what you know and find out Swagger Codegen features that you have not yet discovered.

64 https://github.com/search/advanced?q=
65 https://groups.google.com/forum/#!forum/swagger-swaggersocket
66 https://stackoverflow.com/questions/tagged/swagger-codegen

## 7.3 Review pull requests

In the past year, there have been 1812 change requests submitted against Swagger Codegen[67], which means there are close to 5 change requests on average submitted per day. With so many change requests for enhancements and bug fixes, I want to take this opportunity to thank the Swagger Codegen 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/swagger-api/swagger-codegen/pull/6066 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 Swagger Codegen 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 provides details on what you want to change.

## 7.4 Enhancements or bug fixes

Swagger Codegen would not have achieved its success today without the vibrant community. More than 700 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

67 Github search string: "is:pr created:<2016-07-12"

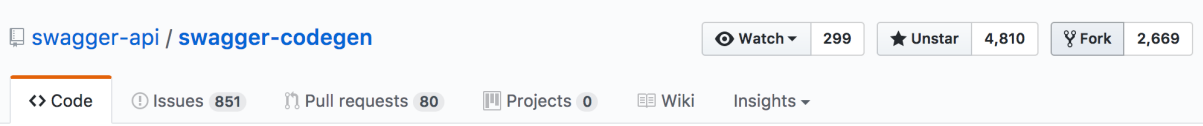
make Swagger Codegen better, you are welcome to start or join the discussion in the issue tracker. One easy way to filter out tasks that need community contribution is to use the search filter below:

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

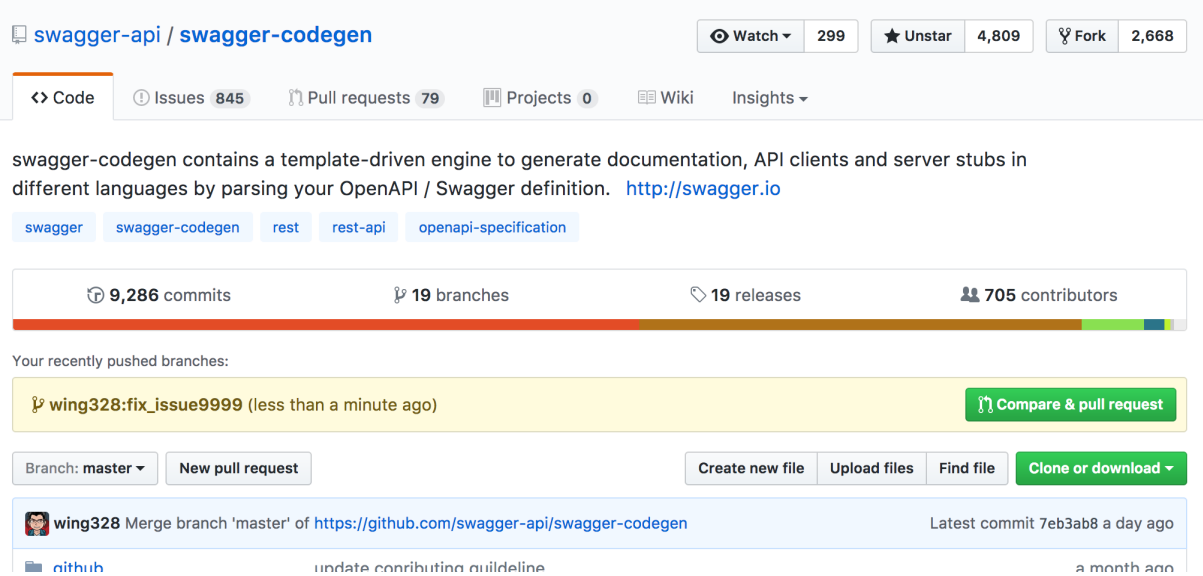
Then you can further filter tasks based on programming languages.

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

1. Visit https://github.com/swagger-api/swagger-codegen 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/swagger-codegen.git
3. cd swagger-codegen
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/swagger-api/swagger-codegen in the browser and click on the button to file a new PR based on fix_issue9999 as shown below




Ref: http://bit.ly/2uHveEu

Contributing to Swagger Codegen 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 experiences with the open-source software. Do a Google search and you will find job openings looking for candidates with experience in OpenAPI and Swagger Codegen so we strongly recommend you to add OpenAPI and Swagger Codegen to your resume.

Reviewing other PRs is also a good way to understand how changes are made to Swagger Codegen. 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 question on the merged PRs, please feel free to reply with your question and the community will try their best to help you out.

## 7.5 Technical committees

For those who want to participate in the future development of Swagger Codegen, 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[68].

68 https://github.com/swagger-api/swagger-codegen#swagger-codegen-technical-committee

# 8. What's next?

Congratulations! You have completed this book to gain a better understanding of OpenAPI (formerly known as Swagger) 2.0 specification and how to leverage Swagger Codegen to streamline your software development experience. This is just the beginning. Both OpenAPI and Swagger Codegen 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 Swagger Codegen, which has new generators contributed by the vibrant community almost every month. We also strongly encourage Swagger Codegen users to share their experience via blog posts, video and more. Some materials shared by the Swagger Codegen community can be found in the project's README[69]. Please feel free to add yours with a pull request.

We are also working hard on writing another book on Swagger Codegen to cover more advanced topics such as creating a generator, customizing the generator in the code level and more. Stay tuned by following my Twitter's account: https://twitter.com/wing328.

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

69 https://github.com/swagger-api/swagger-codegen#companiesprojects-using-swagger-codegen

# Appendix A

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

Swagger Codegen general options:

```
NAME
    swagger-codegen-cli generate - Generate code with chosen lang

SYNOPSIS
    swagger-codegen-cli generate
        [(-a <authorization> | --auth <authorization>)]
        [--additional-properties <additional properties>...]
        [--api-package <api package>] [--artifact-id <artifact id>]
        [--artifact-version <artifact version>]
        [(-c <configuration file> | --config <configuration file>)]
        [-D <system properties>...] [--git-repo-id <git repo id>]
        [--git-user-id <git user id>] [--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>]
        (-l <language> | --lang <language>)
        [--language-specific-primitives <language specific primitives>...]
        [--library <library>] [--model-name-prefix <model name prefix>]
        [--model-name-suffix <model name suffix>]
        [--model-package <model package>]
        [(-o <output directory> | --output <output directory>)]
        [--release-note <release note>] [--remove-operation-id-prefix]
        [--reserved-words-mappings <reserved word mappings>...]
        [(-s | --skip-overwrite)]
        [(-t <template directory> | --template-dir <template directory>)]
        [--type-mappings <type mappings>...] [(-v | --verbose)]

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

    --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.

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

--artifact-id <artifact id>
    artifactId in generated pom.xml

--artifact-version <artifact version>
    artifact version in generated pom.xml

-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 -l {lang} command for language specific config options.

-D <system properties>
    sets specified system properties in the format of
    name=value,name=value (or multiple options, each with name=value)

--git-repo-id <git repo id>
    Git repo ID, e.g. swagger-codegen.

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

--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
    'Swagger-Codegen/{packageVersion}}/{language}'

-i <spec file>, --input-spec <spec file>
    location of the swagger spec, as URL or file (required)

--ignore-file-override <ignore file override location>
    Specifies an override location for the .swagger-codegen-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

-l <language>, --lang <language>
    client language to generate (maybe class name in classpath,
    required)

--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.

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

--model-name-prefix <model name prefix>
    Prefix that will be prepended to all model names. Default is the
    empty string.

--model-name-suffix <model name suffix>
    Suffix that will be appended to all model names. Default is the
    empty string.

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

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

--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.

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

--type-mappings <type mappings>
    sets mappings between swagger spec types and generated code types in
    the format of swaggerType=generatedType,swaggerType=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 feedbacks 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