(/)

-14% -30% -11% -35% -28% -48% -7%

'freestar.com/?
ce=branding&utm_name=baeldung_adhesion)

(h

Genere un cliente REST de Spring Boot con Swagger

Última modificación: 23 de marzo de 2020

p

por baeldung (https://www.baeldung.com/author/baeldung/)
(https://www.baeldung.com/author/baeldung/)

DESCANSO (https://www.baeldung.com/category/rest/)

Bota de primavera (https://www.baeldung.com/category/spring/spring-boot/)

Pavonearse (https://www.baeldung.com/tag/swagger/)

/

Comience con Spring 5 y Spring Boot 2, a través del curso de referencia *Learn Spring* :

/ **>> VER CURSO (/ls-course-start)**

1. Introducción

En este artículo, usaremos los proyectos Swagger Codegen (https://github.com/swagger-api/swagger-codegen) y OpenAPI Generator (https://github.com/OpenAPITools/openapi-generator) para generar clientes REST a partir de un archivo de especificaciones de OpenAPI / Swagger (https://swagger.io/specification/) .

Además, crearemos un proyecto Spring Boot, donde usaremos clases generadas.

Usaremos el ejemplo de la API Swagger Petstore (http://petstore.swagger.io/) para todo.

e

2. Genere un cliente REST con Swagger Codegen

Swagger provides a utility jar that allows us to generate REST clients for various programming languages and multiple frameworks.

2.1. Download Jar File

'freestar.com/?
ce=branding&utm_name=baeldung_adhesion)

The *code-gen_cli.jar* can be downloaded from here (https://search.maven.org/classic/remotecontent?filepath=io/swagger/swagger-codegen-cli/2.2.3/swagger-codegen-cli-2.2.3.jar).

For the newest version, please check the swagger-codegen-cli (https://search.maven.org/classic/#search%7Cgav%7C1%7Cg%3A%22io.swagger%22%20AND%20a%3A%22swagger-codegen-cli%22) repository.

2.2. Generate Client

Let's generate our client by executing the command *java -jar swagger-code-gen-cli.jar generate:*

```
java -jar swagger-codegen-cli.jar generate \
  -i http://petstore.swagger.io/v2/swagger.json \
  --api-package com.baeldung.petstore.client.api \
  --model-package com.baeldung.petstore.client.model \
  --invoker-package com.baeldung.petstore.client.invoker \
  --group-id com.baeldung \
  --artifact-id spring-swagger-codegen-api-client \
  --artifact-version 0.0.1-SNAPSHOT \
  -l java \
  --library resttemplate \
  -o spring-swagger-codegen-api-client
```

The provided arguments consist of:

- A source swagger file URL or path – provided using the *-i* argument
- Names of packages for generated classes – provided using *–api-package*, *–model-package*, *–invoker-package*
- Generated Maven project properties *–group-id*, *–artifact-id*, *–artifact-version*
- The programming language of the generated client – provided using *-l*
- The implementation framework – provided using the *–library*
- The output directory – provided using *-o*

To list all Java-related options, type the following command:

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

Openbank Grupo Santander
Consulta condiciones en openbank.es/hipoteca-open
Hipoteca Open
Solicitud 100 % online
Preaprobación instantánea.
Financiación hasta el 80 %¹.
¹ Del menor valor entre tasación y compraventa.

Swagger Codegen supports the following Java libraries (pairs of HTTP clients and JSON processing libraries):

- *jersey1* – Jersey1 + Jackson
- *jersey2* – Jersey2 + Jackson
- *feign* – OpenFeign + Jackson

- *okhttp-gson* – OkHttp + Gson
- *retrofit* (Obsolete) – Retrofit1/OkHttp + Gson
- *retrofit2* – Retrofit2/OkHttp + Gson
- *rest-template* – Spring RestTemplate + Jackson
- *rest-easy* – Resteasy + Jackson

In this write-up, we chose *rest-template* as it's a part of the Spring ecosystem.

3. Generate REST Client With OpenAPI Generator

OpenAPI Generator is a fork of Swagger Codegen capable of generating 50+ clients from any OpenAPI Specification 2.0/3.x documents.

Whereas Swagger Codegen is maintained by SmartBear, OpenAPI Generator is maintained by a community that includes more than 40 of the top contributors and template creators of Swagger Codegen as founding team members.

3.1. Installation

Perhaps the easiest and most portable installation method is using the *npm* package (https://www.npmjs.com/package/@openapitools/openapi-generator-cli) wrapper, which works by providing a CLI wrapper atop the command-line options supported by the Java code. Installation is straightforward:

```
npm install @openapitools/openapi-generator-cli -g
```

For those wanting the JAR file, it can be found in Maven Central (https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli). Let's download it now:

```
wget https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli/4.2.3/openapi-generator-cli-4.2.3.jar \
  -O openapi-generator-cli.jar
```

3.2. Generate Client

First, the options for OpenAPI Generator are almost identical to those for Swagger Codegen. The most notable difference is the replacement of the *-l* language flag with the *-g* generator flag, which takes the language to generate the client as a parameter.

Next, let's generate a client equivalent to the one we generated with Swagger Codegen using the *jar* command:

```
java -jar openapi-generator-cli.jar generate \
  -i http://petstore.swagger.io/v2/swagger.json \
  --api-package com.baeldung.petstore.client.api \
  --model-package com.baeldung.petstore.client.model \
  --invoker-package com.baeldung.petstore.client.invoker \
  --group-id com.baeldung \
  --artifact-id spring-openapi-generator-api-client \
  --artifact-version 0.0.1-SNAPSHOT \
  -g java \
  -p java8=true \
  --library resttemplate \
  -o spring-openapi-generator-api-client
```

To list all Java-related options, type the command:

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

OpenAPI Generator supports all of the same Java libraries as Swagger CodeGen plus a few extra. The following Java libraries (pairs of HTTP clients and JSON processing libraries) are supported by OpenAPI Generator:

- *jersey1* – Jersey1 + Jackson
- *jersey2* – Jersey2 + Jackson
- *feign* – OpenFeign + Jackson
- *okhttp-gson* – OkHttp + Gson
- *retrofit* (Obsolete) – Retrofit1/OkHttp + Gson
- *retrofit2* – Retrofit2/OkHttp + Gson
- *resttemplate* – Spring RestTemplate + Jackson
- *webclient* – Spring 5 WebClient + Jackson (OpenAPI Generator only)
- *resteasy* – Resteasy + Jackson
- *vertx* – VertX + Jackson
- *google-api-client* – Google API Client + Jackson
- *rest-assured* – rest-assured + Jackson/Gson (Java 8 only)
- *native* – Java native HttpClient + Jackson (Java 11 only; OpenAPI Generator only)
- *microprofile* – Microprofile client + Jackson (OpenAPI Generator only)

4. Generate Spring Boot Project

Let's now create a new Spring Boot project.

4.1. Maven Dependency

We'll first add the dependency of the Generated API Client library – to our project *pom.xml* file:

```
<dependency>
    <groupId>com.baeldung</groupId>
    <artifactId>spring-swagger-codegen-api-client</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

4.2. Expose API Classes as Spring Beans

To access the generated classes, we need to configure them as beans:

```
@Configuration
public class PetStoreIntegrationConfig {

    @Bean
    public PetApi petApi() {
        return new PetApi(apiClient());
    }

    @Bean
    public ApiClient apiClient() {
        return new ApiClient();
    }
}
```

4.3. API Client Configuration

The *ApiClient* class is used for configuring authentication, the base path of the API, common headers, and it's responsible for executing all API requests.

For example, if you're working with OAuth:

```
@Bean
public ApiClient apiClient() {
    ApiClient apiClient = new ApiClient();

    OAuth petStoreAuth = (OAuth) apiClient.getAuthentication("petstore_auth");
    petStoreAuth.setAccessToken("special-key");

    return apiClient;
}
```

4.4. Spring Main Application

We need to import the newly created configuration:

```
@SpringBootApplication
@Import(PetStoreIntegrationConfig.class)
public class PetStoreApplication {
    public static void main(String[] args) throws Exception {
        SpringApplication.run(PetStoreApplication.class, args);
    }
}
```

4.5. API Usage

Since we configured our API classes as beans, we can freely inject them in our Spring-managed classes:

```
@Autowired
private PetApi petApi;

public List<Pet> findAvailablePets() {
    return petApi.findPetsByStatus(Arrays.asList("available"));
}
```

5. Alternative Solutions

There are other ways of generating a REST client other than executing Swagger Codegen or OpenAPI Generator CLI.

5.1. Maven Plugin

A swagger-codegen Maven plugin (https://github.com/swagger-api/swagger-codegen/blob/master/modules/swagger-codegen-maven-plugin/README.md) that can be configured easily in your *pom.xml* allows generating the client with the same options as Swagger Codegen CLI.

This is a basic code snippet that we can include in our project's *pom.xml* to generate client automatically:

```
<plugin>
    <groupId>io.swagger</groupId>
    <artifactId>swagger-codegen-maven-plugin</artifactId>
    <version>2.2.3</version>
    <executions>
        <execution>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <inputSpec>swagger.yaml</inputSpec>
                <language>java</language>
                <library>resttemplate</library>
            </configuration>
        </execution>
    </executions>
</plugin>
```

5.2. Swagger Codegen Online Generator API

An already published API that helps us with generating the client by sending a POST request to the URL *http://generator.swagger.io/api/gen/clients/java* passing the spec URL alongside with other options in the request body.

Let's do an example using a simple curl command:

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

La respuesta sería un formato JSON que contiene un enlace descargable que contiene el código de cliente generado en formato zip. Puede pasar las mismas opciones utilizadas en la CLI de Swaager Codegen para personalizar el cliente de salida.

https://generator.swagger.io (https://generator.swagger.io) contiene una documentación de Swagger para la API donde podemos consultar su documentación y probarla.

5.3. API de generador en línea OpenAPI Generator

Al igual que Swagger Godegen, OpenAPI Generator también tiene un generador en línea. Realicemos un ejemplo usando un comando curl simple:

```
curl -X POST -H "content-type:application/json" \
  -d '{"openAPIUrl":"http://petstore.swagger.io/v2/swagger.json"}' \
  http://api.openapi-generator.tech/api/gen/clients/java
```

La respuesta, en formato JSON, contendrá un enlace descargable al código de cliente generado en formato zip. Puede pasar las mismas opciones utilizadas en la CLI de Swagger Codegen para personalizar el cliente de salida.

https://github.com/OpenAPITools/openapi-generator/blob/master/docs/online.md (https://github.com/OpenAPITools/openapi-generator/blob/master/docs/online.md) contiene la documentación de la API.

6. Conclusión

Swagger Codegen y OpenAPI Generator le permiten generar clientes REST rápidamente para su API con muchos idiomas y con la biblioteca de su elección. Podemos generar la biblioteca cliente utilizando una herramienta CLI, un complemento de Maven o una API en línea.

Este es un proyecto basado en Maven que contiene tres módulos de Maven: el cliente de la API Swagger generado, el cliente OpenAPI generado y la aplicación Spring Boot.

Como siempre, puede encontrar el código disponible en GitHub (https://github.com/eugenp/tutorials/tree/master/spring-swagger-codegen) .

p e
s o
/ n
/ t
fr e
e n
e t_
st d
ar y
c n
o a
m m
/ ic
a

Comience con Spring 5 y Spring Boot 2, a través del curso *Learn Spring* :

>> VER CURSO (/ls-course-end)

n
g
_l
e
a
d
e
r
b
o

Cree su arquitectura de microservicio con Spring Boot y Spring Cloud

Ingrese su dirección de correo electrónico

Descargar ahora

di
n
g
&

4 COMENTARIOS Más antiguo

Ver comentarios

ar d _ at f)

utm_name=baeldung_incontent_dynamic_desktop)

¡Los comentarios están cerrados sobre este artículo!

CATEGORIAS

PRIMAVERA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
LADO DEL CLIENTE HTTP (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)

SERIE

TUTORIAL "VOLVER A LO BÁSICO" DE JAVA (/JAVA-TUTORIAL)
TUTORIAL DE JACKSON JSON (/JACKSON)
TUTORIAL DE HTTPCLIENT 4 (/HTTPCLIENT-GUIDE)
DESCANSO CON SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
TUTORIAL DE PERSISTENCIA DE PRIMAVERA (/PERSISTENCE-WITH-SPRING-SERIES)
SEGURIDAD CON SPRING (/SECURITY-SPRING)
TUTORIALES REACTIVOS DE PRIMAVERA (/SPRING-REACTIVE-GUIDE)

ACERCA DE

SOBRE BAELDUNG (/ABOUT)
LOS CURSOS (HTTPS://COURSES.BAELDUNG.COM)
TRABAJOS (/TAG/ACTIVE-JOB/)
EL ARCHIVO COMPLETO (/FULL_ARCHIVE)
ESCRIBE PARA BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORES (/EDITORS)
NUESTROS COMPAÑEROS (/PARTNERS)
ANÚNCIESE EN BAELDUNG (/ADVERTISE)

TÉRMINOS DE SERVICIO (/TERMS-OF-SERVICE)
POLÍTICA DE PRIVACIDAD (/PRIVACY-POLICY)
INFORMACIÓN DE LA COMPAÑÍA (/BAELDUNG-COMPANY-INFO)
CONTACTO (/CONTACT)