(/)

Generate Spring Boot REST Client with Swagger

Last modified: November 5, 2018

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

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

Spring Boot (https://www.baeldung.com/category/spring/spring-boot/)

1. Introduction

In this article, we'll use the Swagger CodeGen (https://github.com/swagger-api/swagger-codegen) project to generate a REST client from an OpenAPI/Swagger spec (https://swagger.io/specification/) file.

Also, we'll create a Spring Boot project, where we'll use generated classes.

We'll use the Swagger Petstore (http://petstore.swagger.io/) API example for everything.

2. Generate REST Client

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

2.1. Download Jar File

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
```

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 Spring Boot Project

Let's now create a new Spring Boot project.

3.1. 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>
```

3.2. 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();
    }
}
```

3.3. 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;
}
```

3.4. 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);
    }
}
```

3.5. 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"));
}
```

4. Alternative Solutions

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

4.1. 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>
```

4.2. 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://petstore.swagger.io/v2/swagger.
http://generator.swagger.io/api/gen/clients/java
```

The response would be JSON format that contains a downloadable link that contains the generated client code in zip format. You may pass the same options used in the Swaager Codegen CLI to customize the output client.

https://generator.swagger.io (https://generator.swagger.io) contains a Swagger documentation for the API where we can check its documentation and try it.

5. Conclusion

Swagger Codegen enables you to generate REST clients quickly for you API with many languages and with the library of your choice. We can generate the client library using CLI tool, Maven plugin or Online API.

The implementation of this example can be found in this GitHub project (https://github.com/eugenp/tutorials/tree/master/spring-swagger-codegen). This is a Maven based project that contains two Maven modules, the generated API client, and the Spring Boot application.

▲ newest ▲ oldest ▲ most voted

Guest

ApacheEnthu

Any reason why we have `PetStoreIntegrationConfig.apiClient` annotated with @Bean? All it can be seen is its been injected in PetApi?

+ 0 −

1 year ago

Grzegorz Piwowarek (http://4comprehension.com)(https://www.baeldung.com/author/grzegorz-author/)

Member

Yeah, but keep in mind that this is a super simple example – normally, it could be injected in multiple different classes

+ 0 −

1 year ago

Guest

Julio Villane

The versión '2.2.2-SNAPSHOT' doesn'tn work.

You have to use '2.2.2' or the new one '2.2.3'

+ 0 −

1 year ago

Grzegorz Piwowarek (http://4comprehension.com)(https://www.baeldung.com/author/grzegorz-author/)

Member

Good catch, thanks!

+ 0 −

1 year ago

Comments are closed on this article!

CATEGORIES

SPRING (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
REST (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SECURITY (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
HTTP CLIENT-SIDE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIES

JAVA "BACK TO BASICS" TUTORIAL (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)

HTTPCLIENT 4 TUTORIAL (/HTTPCLIENT-GUIDE)
REST WITH SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
SPRING PERSISTENCE TUTORIAL (/PERSISTENCE-WITH-SPRING-SERIES)
SECURITY WITH SPRING (/SECURITY-SPRING)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE COURSES (HTTPS://COURSES.BAELDUNG.COM)
CONSULTING WORK (/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
THE FULL ARCHIVE (/FULL_ARCHIVE)
WRITE FOR BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORS (/EDITORS)
OUR PARTNERS (/PARTNERS)
ADVERTISE ON BAELDUNG (/ADVERTISE)

TERMS OF SERVICE (/TERMS-OF-SERVICE)
PRIVACY POLICY (/PRIVACY-POLICY)
COMPANY INFO (/BAELDUNG-COMPANY-INFO)
CONTACT (/CONTACT)