Automated Code Generation for Inter-parameter Dependencies in REST APIs

Saman Barakat[1][0000−0002−7714−3742], Enrique Barba Roque[0000−0002−7018−498X], Ana Belén Sánchez[1][0000−0003−1473−0955], and Sergio Segura[1][0000−0001−8816−6213]

SCORE Lab, I3US Institute,
Universidad de Sevilla,
Seville, Spain
`saman.barakat@gmail.com`

Abstract. The generation of code templates from REST API specifications is a common practice in industry. However, existing tools neglects the dependencies among input parameters (so called inter-parameter dependencies), extremelly common in practice and usually described in natural language. As a result, developers are responsible for implementing the corresponding validation logic manually, a tedious and error-prone process. In this paper, we present an approach for the automated generation of code for inter-parameter dependencies specified using the IDL4OAS extension. As a proof of concept, we present an extension of the popular openapi-generator tool ecosystem, automating the generation of Java and Python code for the management of inter-parameter dependencies in both, servers and clients. Preliminary results show the effectiveness of the approach in accelerating the development of APIs while making them potentially more reliable.

Keywords: Web APIs · Scaffolding · Code generation · OpenAPI Generator

1 Introduction

REST Application Programming Interfaces (APIs) are widely described using the OpenAPI Specification (OAS) language [5], which provides a structural description of the APIs. OAS document enables developers and machines to use and understand the functionality of services without knowledge of the actual code implementation.

One of the applications of OAS is scaffolding: OAS documents can be used by code generation tools to produce client and server code in various programming languages. This saves time and effort for both API developers and API consumers. OpenAPI Generator is a popular code generation tool for OAS [4]. It is developed in Java and has over 50 generators for clients and servers in different languages.

Inter-parameter dependencies (or simply dependencies) are constraints that restrict the way in which two or more input parameters can be combined to form a valid call to the service. For example in the Yelp API [6] (endpoint *businesses/search*), the parameter `location` is required if either the `latitude` or `longitude` parameters are not provided, and both parameters are required if the `location` is not provided. A recent study revealed that dependencies are extremelly common and pervasive in industrial REST APIs: they occur in 4 out of every 5 APIs across all application domains and types of operations [9]. However, OAS provides no support for the formal description of these types of dependencies, despite being a highly demanded feature by practitioners[1]. Instead, users are encouraged to describe them informally using natural language[2].

Current scaffolding tools do not support inter-parameter dependencies because they are not supported in OAS. Therefore, the validation code associated to these dependencies must be manually implemented. In the previous example, for instance, developers should write the required assertions to make sure that (`latitude` and `longitude`) must be used together when the (`location`) parameter is not provided, both in clients and servers. This is not only tedious, but also error-prone making validation failures very common in practice [10].

The aim of this paper is to automate the generation of validation code for inter-parameter dependencies in REST APIs. Specifically, we propose a specification-driven approach supported by an extension of the popular tool OpenAPI Generator.

2 Approach

Our approach relies on the IDL4OAS extension for OAS [8]. Specifically, IDL4OAS allows to formally describe the inter-parameter dependencies of API operations using the domain specific language IDL (Inter-parameter Dependency Language) [8].

Our work follows a model-based approach where the specification of the API—including the description of its dependencies using IDL4OAS—is used as a starting point for the generation of code. Our approach has been implemented in an extension of OpenAPI Generator as a base, extending its functionalities to generate code assertions for the dependencies. This way, we can just focus on the process of parsing IDL and generating code, while also providing the OpenAPI community with the possibility of using IDL for their own projects.

OpenAPI Generator uses a set of templates to generate code for a specific programming language and/or framework. The templates contain common code, independent of the specific API, and have variables that are replaced with the parsed and processed information from the OAS file. For example, a Java template file could be a class skeleton that implements the operations of an API, and the method names would be substituted from the parsed OAS file.

The project extends the OpenAPI Generator logic that processes the OAS file and generates the variable map for the template. If an operation uses the IDL4OAS extension [8], this is processed into a code assertion in the target language and added to the variable map. The templates are modified to include a conditional block that will add as many assertions as there are in the map. Figure 1 illustrates the process of automatically generating code with the OpenAPI

[1] https://github.com/OAI/OpenAPI-Specification/issues/256
[2] https://swagger.io/docs/specification/describing-parameters/#dependencies

Generator using the extension for the management of inter-parameter dependencies. An example of OAS file using IDL with dependencies is shown in Listing 1. The generated code from the before IDL example is presented in Listing 2.

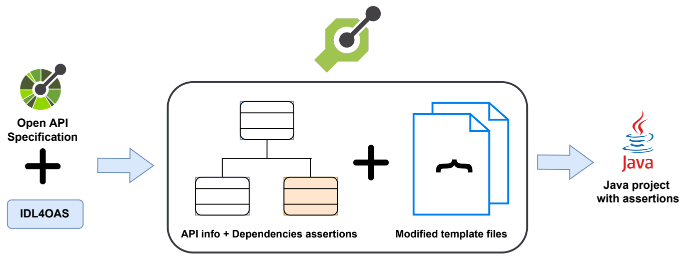
Open API Specification
IDL4OAS
API info + Dependencies assertions
Modified template files
Java
Java project with assertions

Fig. 1: Process of generating code with OpenAPI Generator

```
paths:
  /businesses/search:
  get:
      parameters:
        - name: location [...]
        - name: latitude [...]
        - name: longitude [...]
        - name: open_now [...]
        - name: open_at [...]
        - [...]
      [...]
      x-dependencies:
        - Or(location, latitude AND longitude);
        - ZeroOrOne(open_now, open_at);
```

Listing 1: OAS document of the businesses/search operation from the Yelp API extended with IDL4OAS

```
// Check dependency: Or(location, latitude AND longitude);
if(DependencyUtil.doNotSatisfyOrDependency((location != null),(latitude != null) && (longitude != null))){
    return new ResponseEntity("Dependency not satisfied: Or(location, latitude AND longitude);",
                HttpStatus.BAD_REQUEST);
        }
// Check dependency: ZeroOrOne(open_now, open_at);
 if(DependencyUtil.doNotSatisfyZeroOrOneDependency((openNow != null),(openAt != null))){
    return new ResponseEntity("Dependency not satisfied: ZeroOrOne(open_now, open_at);",
                HttpStatus.BAD_REQUEST);
        }
```

Listing 2: Code with inter-parameter dependencies automatically generated

3 Preliminary evaluation

As a preliminary evaluation, we compared the code generated by the original version of OAS Generator (without suppport for dependencies) and our extension of the tool (with support for dependencies) in terms of lines of code. Table 1 shows the result of our experiment on 5 real-world API operations. As illustrated, the original tool generated 70 lines of code for the videos resource in the Youtube API [7], whereas our extension generated 104 lines of code after including the IDL specification, i.e., 34 new lines of code that represent an increase of 32% over the original code. While, in the Github API [2], it can be seen that the original tool generated 108 lines of code and our tool generated 123 lines, which makes 15 lines more related to IDL specification (about 12% of increment regarding the original code). Similarly, after using our tool with Yelp [6], OMDb [3], and DHL [1] APIs, it generated 12, 13, and 12 lines more for each API, respectively, comparing with the original tool. It can be observed that the Youtube API has more numbers of code lines generated compared to the other APIs. This is because Youtube API has more IDL dependencies than the other APIs used in this experiment.

API	**Resource**	**#LOC Original tool**	**#LOC IDL extension**
Youtube	GET /youtube/v3/videos	70	**104**
Github	GET /user/repos	108	**123**
Yelp	GET /transactions/search	52	**64**
OMDb	GET /search	56	**69**
DHL	GET /find-by-address	76	**88**

Table 1: #LOC in OpenAPI Generator tool and in our extension

4 Conclusions

In this paper, we aimed to develop an extension for the OpenAPI Generator tool to support code generation of inter-parameter dependencies in the REST APIs. As far as we know, this is the first work aimed to generate code of inter-parameter dependencies in the REST APIs. Preliminary evaluations show that our work accelerates the development and makes APIs potentially more reliable since some of the validation logic is added automatically. In the future, we plan to improve the tool and enable the generation of code for more programming languages.

Acknowledgements

This work has been partially supported by the European Commission (ERDF) and Junta de Andalucía under projects APOLO (US-1264651) and EKIPMENT-PLUS (P18-FR-2895), and by the Spanish Government under project HORATIO (RTI2018-101204-B-C21).

References

1. Dhl location finder api. https://developer.dhl.com, online; accessed May 2022
2. Github api. https://docs.github.com/en/rest/repos, online; accessed May 2022
3. Omdb api. http://www.omdbapi.com/, online; accessed May 2022
4. Openapi generator. https://openapi-generator.tech/, online; accessed April 2022
5. OpenAPI Specification. https://swagger.io/specification/, online; accessed March 2022
6. Yelp api. https://www.yelp.com/developers/documentation/v3, online; accessed May 2022
7. Youtube data api. https://developers.google.com/youtube/v3/docs/videos/list, online; accessed May 2022
8. Martin-Lopez, A., Segura, S., Muller, C., Ruiz-Cortes, A.: Specification and automated analysis of inter-parameter dependencies in web apis. IEEE Transactions on Services Computing pp. 1–14 (2021). https://doi.org/10.1109/TSC.2021.3050610
9. Martin-Lopez, A., Segura, S., Ruiz-Cortés, A.: A catalogue of inter-parameter dependencies in restful web apis. In: Yangui, S., Bouassida Rodriguez, I., Drira, K., Tari, Z. (eds.) Service-Oriented Computing. pp. 399–414. Springer International Publishing, Cham (2019)
10. Martin-Lopez, A., Segura, S., Ruiz-Cortés, A.: Restest: Black-box constraint-based testing of restful web apis. In: Kafeza, E., Benatallah, B., Martinelli, F., Hacid, H., Bouguettaya, A., Motahari, H. (eds.) Service-Oriented Computing. pp. 459–475. Springer International Publishing, Cham (2020)