23-Swagger & OpenAPI Specification 3

Author: Vincent Lau

Learning Objectives

> Understand how to use Swagger annotations to construct Swagger UI & Open API Spec.
>
> What is API Documenetaton (in json format)
>
> What is the Swagger UI

Introduction

Swagger and OpenAPI are related tools and specifications that are commonly used to document and test RESTful APIs.

Swagger

Swagger is a framework for describing, producing, consuming, and visualizing RESTful web services. It provides a range of tools for API development, including tools for API

documentation, client generation, and API testing. Swagger 3 is also known as the "OpenAPI Specification" version 3.

OpenAPI Spec

OpenAPI is a specification for building APIs. It defines a standard, language-agnostic interface for RESTful APIs, allowing both humans and computers to understand API capabilities without access to source code or documentation. The OpenAPI Specification (OAS) provides a format for API documentation and can be used to generate interactive API documentation.

Key Features and Components

Formal Specification vs. Ecosystem

- OpenAPI 3: OpenAPI 3 is a formal specification for describing RESTful APIs. It provides a standard format in JSON or YAML for documenting APIs.
- Swagger 3: Swagger 3 is the ecosystem of tools, libraries, and implementations built around the OpenAPI Specification. It includes tools for generating API documentation, client SDKs, server code, and more.

OpenAPI as the Standard

- OpenAPI 3: OpenAPI is the official and standardized term for the API specification. It is governed by the OpenAPI Initiative, which is part of the Linux Foundation.
- Swagger 3: "Swagger" is often used informally to refer to API documentation and the Swagger UI. However, Swagger is now aligned with the OpenAPI Specification.

API Documentation and Visualization

- OpenAPI 3: OpenAPI 3 provides the format for documenting API contracts but does not include tools for rendering interactive documentation.
- Swagger 3: The Swagger ecosystem includes tools like Swagger UI for generating interactive API documentation.

Tooling and Libraries

- OpenAPI 3: As a specification, OpenAPI defines the structure of API documentation, but it doesn't provide tooling directly.
- Swagger 3: The Swagger ecosystem includes a wide range of tools, including code generators, testing libraries, and UI components.

Extensibility and Customization

- OpenAPI 3: The OpenAPI Specification is highly extensible, allowing you to include custom metadata, vendor-specific extensions, and other details in your API documentation.
- Swagger 3: The Swagger ecosystem allows you to customize API documentation using Swagger extensions and tools.

Versioning and Evolution

- OpenAPI 3: OpenAPI encourages versioning and evolving APIs while maintaining compatibility. It provides features for documenting different API versions.
- Swagger 3: The Swagger ecosystem supports versioning and evolving APIs, and tools like Swagger Codegen can generate client code for specific API versions.

Adoption and Community

- OpenAPI 3: OpenAPI has a broad and growing community of users and contributors, including major organizations, making it a de facto standard for API documentation.
- Swagger 3: Swagger is the most widely recognized name in the API documentation space, with a strong community and ecosystem.

Implementations

1. Maven Dependency

This starter includes most of the necessary dependencies for Swagger integration.

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.2.0</version>
</dependency>
```

Swagger provides a variety of annotations to help you document and describe your API comprehensively. Here are some of the most commonly used Swagger annotations, along with a brief description of each:

2. Code Example

```
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
```

```java
import io.swagger.v3.oas.annotations.media.Content;
import io.swagger.v3.oas.annotations.media.Schema;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.responses.ApiResponses;

public interface DataOperation {

    @Operation(summary = "Get Finnhub Stock Data",
      description = "This endpoint retrieves a stock data from Finnhub Endpoints (/stock).",
      tags = "Get a Stock",
      parameters = {@Parameter(name = "SymbolReqDTO",
          description = "Stock Symbol", required = true)})
    @ApiResponses({
      @ApiResponse(responseCode = "200",
          content = {@Content(schema = @Schema(implementation = Stock.class),
              mediaType = "application/json")}),
      @ApiResponse(responseCode = "404",
          content = {@Content(schema = @Schema())}),
      @ApiResponse(responseCode = "500",
          content = {@Content(schema = @Schema())})})
    @GetMapping(value = "/stock")
    @ResponseStatus(value = HttpStatus.OK)
    ApiResp<StockDTO> stockInfo(
      @RequestParam("symbol") SymbolReqDTO symbol);

}
```

3. @Operation

- **Description**: Annotates a method to describe an API operation.
- **Attributes**:
  - `summary` : A brief description of the operation.
  - `description` : Additional details or notes about the operation.
  - `tags` : The HTTP method used for the operation (e.g., "GET" or "POST").
  - `parameters` : An array of `@Parameter` annotations describing possible responses.
    - `name` - parameter name
    - `description` - brief description of the parameter
    - `required` - mandatory

4. @ApiResponses & @ApiResponse

- **Description**: Annotates a method to specify a possible response from an API operation.
- **Attributes**:
  - `responseCode` : The HTTP status code for the response.
  - `content` : A description of the response.

```
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;


public ResponseEntity<String> createResource(@RequestBody ResourceDTO
resourceDTO) {
    // Controller logic
}
```

5. @Schema

- **Description**: Annotates fields in a model to specify additional information about the property.

- **Attributes**:

  - `Schema` : A description of the property.

```
public class ApiResp<T> {
  // attribute name by default same as JSON field name after serialziation
  @Schema(description = "Code For System Response Cat")
  private int code;

  @Schema(description = "Message to indicate error")
  private String message;

  @Schema(description = "Response Data Body")
  private T data;
}
```

6. @Hidden

- **Description**: Annotates a class, method, or property to indicate that it should be ignored by the Swagger documentation.

```
@Hidden
@GetMapping("/ignored")
```

```
3 public ResponseEntity<String> ignoredEndpoint() {
4     // Controller logic
5 }
```

These annotations allow you to provide detailed information about your API, including its operations, parameters, responses, and models. You can use them to create comprehensive and interactive documentation using Swagger (OpenAPI 3). Additionally, you can customize the annotations to suit your API's specific requirements and use cases.

Access Swagger UI

Start your Spring Boot application. You can access the Swagger UI by navigating to `http://localhost:8085/swagger-ui/index.html` in your web browser.

The Swagger UI provides an interactive interface for exploring and testing your API endpoints.

Access API Documentation

The generated OpenAPI documentation is also available as JSON. You can access it at `http://localhost:8085/v3/api-docs`

Summary

Overall, Swagger 3 / OpenAPI 3 is a powerful tool for API documentation and development, and it is widely adopted in the API development community. You can use it to create well-documented APIs that are easy to understand and consume.