# Documenting a Spring REST API Using OpenAPI 3.0

## What is openapi/swagger :

Swagger is an open-source tool that helps developers create, document, and test RESTful APIs for Spring Boot projects

* **Documentation**: Swagger generates API documentation in a visual format that can be viewed in a web browser. It can automatically convert the API structure defined in code annotations into an API specification.
* **Testing**: Swagger provides a user interface that lets users test API calls in the browser.
* **Design**: Swagger can be used to design APIs.

Swagger's main components include:

* **Swagger Editor**: For writing and editing API specifications
* **Swagger UI**: For creating interactive API documentation
* **Swagger Codegen**: For generating server stubs or client libraries for an API

## Integrating Swagger into a Spring Boot project is relatively easy. You can do this by:

1. Updating your Maven POM file or editing your Gradle build file
2. Creating a new Spring Boot project from scratch using Spring Initializr

After Swagger is included in the project and the project is run, it will start serving at http://localhost:8080/swagger-ui/index.html.

## Swagger documentation has many advantages, including:

* **Automated documentation**

Swagger automatically generates and updates documentation as the API evolves.

* **Improved collaboration**

Swagger provides a unified API specification for developers, testers, and other stakeholders to work from.

* **Interactive API exploration**

Developers can explore and test APIs directly from the Swagger documentation.

* **API versioning**

Swagger supports API versioning, allowing developers to manage changes and updates.

* **Enhanced security**

Swagger allows developers to define and validate API request and response formats, which reduces the likelihood of errors.

* **Clear documentation**

Swagger provides clear documentation to help testers understand the API functionality, requirements, and expectations.

* **Ease of use**

Swagger is easy to use, and testers can view the API details, try out requests and responses, and see errors and validations without having to write complex scripts.

* **Uniformity**

Swagger enables a level of uniformity that makes it easier for developers to learn and consume APIs.

Swagger is a framework for designing APIs with a common language. It was created in 2011 by Tony Tam and was later renamed to the OpenAPI Specification.

Documentation is an essential part of building REST APIs. In this tutorial, we’ll look at SpringDoc, which simplifies the generation and maintenance of API docs based on the OpenAPI 3 specification for Spring Boot 3.x applications.

## Setting up springdoc-openapi

Spring Boot 3.x requires to use [version 2](https://github.com/springdoc/springdoc-openapi/releases/tag/v2.1.0) of [springdoc-openapi](https://central.sonatype.com/search?q=springdoc-openapi-starter-webmvc-ui):

<**dependency**>

<**groupId**>org.springdoc</**groupId**>

<**artifactId**>springdoc-openapi-starter-webmvc-ui</**artifactId**>

<**version**>2.5.0</**version**>

</**dependency**>Copy

## OpenAPI Description Path

After setting up the dependency correctly, we can run our application and find the OpenAPI descriptions at /v3/api-docs, which is the default path:

<http://localhost:8080/v3/api-docs>

Further, we can customize the path in application.properties using the springdoc.api-docs property. For example, we can set the path to /api-docs:

springdoc.api-docs.path=/api-docs

Then, we’ll be able to access the docs at:

<http://localhost:8080/api-docs>

The OpenAPI definitions are in JSONformat by default. For yaml format, we can obtain the definitions at:

<http://localhost:8080/api-docs.yaml>

## Integration With Swagger UI

Besides generating the OpenAPI 3 specification, we can integrate springdoc-openapi with Swagger UI to interact with our API specification and exercise the endpoints.

**The springdoc-openapi dependency already includes Swagger UI**, so we’re all set to access the API documentation at:

http://localhost:8080/swagger-ui/index.html