# SWAGGER

Swagger is an open-source framework that is used for designing, building, documenting, and testing RESTful APIs. It provides a set of tools for creating API documentation that is easy to read and understand and can be used by both humans and machines.

The Swagger framework includes several components, including:

1. Swagger Editor: A web-based editor that allows you to design and edit OpenAPI (formerly known as Swagger) specifications.
2. Swagger UI: A collection of HTML, CSS, and JavaScript files that allow you to view and interact with the OpenAPI specification in a web browser. Swagger UI provides a user-friendly interface for exploring and testing RESTful APIs.
3. Swagger Codegen: A tool that generates server and client code from an OpenAPI specification. This allows developers to quickly create boilerplate code for their API implementations.
4. SwaggerHub: A cloud-based platform for hosting, designing, and collaborating on OpenAPI specifications. SwaggerHub provides a centralized location for managing your API documentation and sharing it with others.

Overall, Swagger helps to streamline the API development process by providing tools and standards for designing and documenting RESTful APIs.

## Difference between spring doc OpenAPI and Swagger

Springdoc-openapi and Swagger are both tools for generating API documentation based on the OpenAPI specification (formerly known as Swagger specification). However, there are some differences between them.

1. Framework: Springdoc-openapi is a library specifically designed for generating OpenAPI documentation for Spring Boot applications, while Swagger is a general-purpose tool that can be used with any framework.
2. Configuration: Springdoc-openapi uses annotations to document the API endpoints and models, whereas Swagger uses a YAML or JSON file to define the API specification.
3. Integration: Springdoc-openapi is tightly integrated with the Spring framework, making it easy to use with Spring Boot applications. Swagger, on the other hand, can be used with any framework and requires additional configuration to integrate with Spring.
4. Features: Both tools offer similar features for documenting APIs, including support for defining paths, parameters, responses, and authentication. However, Springdoc-openapi offers some additional features, such as automatic generation of the OpenAPI documentation based on the Spring REST API and support for Spring Data REST.

In summary, while both Springdoc-openapi and Swagger are useful tools for generating API documentation based on the OpenAPI specification, Springdoc-openapi is more tightly integrated with the Spring framework and offers some additional features specifically designed for Spring Boot applications.

## Is it possible to use yaml or json file to define the api specification in Swagger 2

|  |
| --- |
| import static springfox.documentation.spi.DocumentationType.SWAGGER\_2;  import java.util.List;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;  import org.springframework.context.annotation.Primary;  import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  import springfox.documentation.builders.PathSelectors;  import springfox.documentation.builders.RequestHandlerSelectors;  import springfox.documentation.spring.web.plugins.Docket;  import springfox.documentation.swagger.web.SwaggerResource;  import springfox.documentation.swagger.web.SwaggerResourcesProvider;  import springfox.documentation.swagger2.annotations.EnableSwagger2;  @Configuration  @EnableSwagger2  public class SwaggerConfiguration implements WebMvcConfigurer {  @Bean  public Docket swagger() {  return new Docket(SWAGGER\_2)  .select()  .apis(RequestHandlerSelectors.any())  .paths(PathSelectors.any())  .build();  }  @Override  public void addResourceHandlers(ResourceHandlerRegistry registry) {  registry.addResourceHandler("/swagger-apis/\*\*")  .addResourceLocations("classpath:/static/swagger-apis/");  }  @Primary  @Bean  public SwaggerResourcesProvider swaggerResourcesProvider() {  return () -> List.of(loadResource("combat"));  }  private SwaggerResource loadResource(String resource) {  SwaggerResource wsResource = new SwaggerResource();  wsResource.setName(resource);  wsResource.setSwaggerVersion("2.0");  wsResource.setLocation("/swagger-apis/swagger.yaml");  //location path: resource -> static -> swagger-apis -> swagger.yaml  return wsResource;  }  } |