Swagger - Um Guia Prático Integrado para Java e Spring Boot

A documentação de uma API é essencial para garantir que outros desenvolvedores possam utilizá-la corretamente e sem dificuldades.

No entanto, mantê-la atualizada e organizada nem sempre é fácil. O **Swagger** surge como uma solução eficaz para esse problema, fornecendo documentação interativa diretamente a partir do código, permitindo que os usuários testem os endpoints da API e visualizem as respostas de maneira prática e dinâmica.

Neste artigo, vamos abordar, em detalhes, a integração completa do Swagger com **Java LTS** (versões 8, 11, 17 e a última versão) utilizando **Spring Boot**, com exemplos reais de código. Ao longo do texto, você aprenderá a configurar, documentar e visualizar sua API em uma interface amigável. Além disso, exploraremos as versões recomendadas do Spring Boot para cada versão do Java, garantindo que você tire o máximo proveito dessa integração.

O que é Swagger e Por Que Usá-lo?

Swagger é uma poderosa ferramenta open-source que permite a geração automática de documentação para APIs RESTful. Ele cria uma interface interativa que permite aos desenvolvedores visualizar e testar as APIs diretamente no navegador. Além disso, é amplamente compatível com diversas linguagens de programação e frameworks.

Vantagens do Swagger:

- **Documentação Automática**: Baseada diretamente no código da API, mantendo sempre atualizado conforme o desenvolvimento avança.
- **Interface Interativa**: Permite que desenvolvedores testem as requisições e respostas diretamente pelo navegador.
- **Padrão Global**: Amplamente adotado por grandes empresas, o que facilita a integração entre equipes e sistemas diferentes.
- **Agilidade no Desenvolvimento**: Reduz o esforço manual de criar e manter documentações separadas, evitando desatualizações.

Estrutura Completa de Configuração e Documentação com Swagger

Agora, vamos explorar como configurar e documentar APIs em **Java LTS** e **Spring Boot** de maneira detalhada, mostrando o passo a passo para cada versão do Java e Spring Boot.

1. Configuração do Swagger com Java 8 e Spring Boot 2.1.x

Apesar de já existir há bastante tempo, o **Java 8** ainda é amplamente utilizado em projetos mais antigos, sendo uma das versões LTS mais estáveis. Para essa configuração, vamos utilizar a biblioteca **Springfox**.

1. **Adicionar dependências ao `pom.xml`**:

   No projeto Maven, adicione as seguintes dependências para habilitar o Swagger:

```
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger2</artifactId>
   <version>2.9.2</version>
</dependency>
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger-ui</artifactId>
   <version>2.9.2</version>
</dependency>
```

2. **Criar uma classe de configuração**:

   Para habilitar o Swagger no projeto, crie uma classe de configuração anotada com @EnableSwagger2.

```
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableSwagger2
public class ConfiguracaoSwagger {

   @Bean
   public Docket api() {
      return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.basePackage("com.seuprojeto.api"))
            .paths(PathSelectors.any())
            .build();
   }
```

```
}
```

3. **Criar o controlador para expor endpoints**:

   Agora, crie um controlador básico com um endpoint de exemplo.

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ControladorMensagem {

    @GetMapping("/api/mensagem")
    public String obterMensagem() {
        return "API rodando com Java 8 e Swagger!";
    }
}
```

4. **Acessar a interface do Swagger**:

   Após iniciar a aplicação, a interface do Swagger estará disponível no navegador em: http://localhost:8080/swagger-ui.html

2. Configuração do Swagger com Java 11 e Spring Boot 2.3.x

Com o lançamento do **Java 11**, foram introduzidos novos recursos e melhorias de performance, e é considerado um marco importante no ciclo LTS do Java. Aqui, vamos utilizar a biblioteca **springdoc-openapi** que traz integração nativa com o Swagger.

1. **Adicionar dependências no pom.xml**:

   Para integrar o Swagger com o Java 11, basta adicionar a seguinte dependência:

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.4.3</version>
</dependency>
```

2. **Criar um controlador de exemplo com anotação @Operation**:

   A anotação @Operation permite detalhar cada endpoint diretamente no código.

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import io.swagger.v3.oas.annotations.Operation;

@RestController
public class ControladorMensagem {
```

```
    @Operation(summary = "Retorna uma mensagem", description = "Este endpoint
devolve uma mensagem simples.")
    @GetMapping("/api/mensagem")
    public String obterMensagem() {
        return "API rodando com Java 11 e Swagger!";
    }
}
```

3. **Acessar a interface do Swagger**:

   Após a execução da aplicação, a documentação estará disponível em: http://localhost:8080/swagger-ui.html

3. Configuração do Swagger com Java 17 e Spring Boot 2.6.x

O **Java 17**, sendo a versão LTS mais recente, traz uma série de novos recursos como **Pattern Matching** e **Records**, além de melhorias na performance. Para essa versão, vamos manter o uso do **springdoc-openapi**.

1. **Adicionar dependências ao pom.xml**:

   Para Java 17 e Spring Boot 2.6.x, adicione a dependência abaixo:

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.13</version>
</dependency>
```

2. **Criar o controlador com recursos do Java 17**:

   O Java 17 permite a utilização de **Records** para simplificar a criação de objetos imutáveis. Podemos utilizá-lo em nosso controlador.

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import io.swagger.v3.oas.annotations.Operation;

@RestController
public class ControladorMensagem {

    @Operation(summary = "Exemplo com Java 17", description = "Demonstra o uso de
Records no Java 17.")
    @GetMapping("/api/mensagem")
    public Mensagem obterMensagem() {
        return new Mensagem("API rodando com Java 17 e Swagger!");
    }

    public record Mensagem(String conteudo) {}
}
```

3. **Acessar a interface do Swagger**:

   Após iniciar a aplicação, a interface Swagger pode ser acessada em: http://localhost:8080/swagger-ui.html

4. Configuração do Swagger com a Última Versão do Java (JDK 23) e Spring Boot 3.x

Com o **Java 23**, a comunidade de desenvolvimento experimenta avanços significativos em **concorrência** com o **Project Loom** e em **Pattern Matching**. O **Spring Boot 3.x** foi otimizado para aproveitar ao máximo essas funcionalidades.

1. **Adicionar dependências no pom.xml**:

   Para usar o Swagger com o Java 23 e Spring Boot 3.x, utilize a seguinte dependência:

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.7.0</version>
</dependency>
```

2. **Criar o controlador utilizando recursos do Project Loom**:

   O **Project Loom** permite criar aplicações concorrentes de forma simplificada. Aqui está um exemplo de como usar isso em um endpoint de API:

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import io.swagger.v3.oas.annotations.Operation;

@RestController
public class ControladorMensagem {

    @Operation(summary = "Exemplo com Project Loom", description = "Utiliza Project
Loom para operações concorrentes.")
    @GetMapping("/api/mensagem")
    public String obterMensagem() {
        return "API rodando com Project Loom e Swagger no Java 23!";
    }
}
```

3. **Acessar a interface do Swagger**:

   A documentação gerada estará disponível em:
   http://localhost:8080/swagger-ui.html

A integração do **Swagger** com as diferentes versões do **Java LTS** e **Spring Boot** permite que você otimize o desenvolvimento de APIs, oferecendo uma documentação atualizada e interativa de forma automática. Cada versão do Java traz novas funcionalidades que podem ser aproveitadas junto com o Swagger para tornar suas APIs ainda mais eficientes e fáceis de usar.

EducaCiência FastCode para a comunidade