



Desenvolvimento para Servidores-II

Documentação com o Swagger

Neste tópico abordaremos o uso da SpringDoc e Swagger para documentar uma API REST.

Prof. Ciro Cirne Trindade

SpringDoc

- SpringDoc é uma biblioteca que auxilia a geração automática da documentação de uma API
- Baseado na especificação OpenAPI 3
- Oferece suporte ao Swagger



- Para que o springdoc-openapi gere automaticamente a documentação da especificação OpenAPI 3 para sua API, simplesmente adiciona a dependência do springdoc-openapi-ui no seu pom.xml:

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-ui</artifactId>
  <version>1.6.8</version>
</dependency>
```

- Então quando você executar sua aplicação, as descrições do OpenAPI estarão disponíveis no caminho `/v3/api-docs` por default:
 - <http://localhost:8080/v3/api-docs/>
- Para customizar esse caminho, indique-o no arquivo `application.properties`:
 - `springdoc.api-docs.path=/api-docs`

Assim a documentação estará disponível em: <http://localhost:8080/api-docs/>



- swagger.io
- Ferramenta *open source* para a geração automática da documentação de APIs
- Automatiza o processo de geração e atualização da documentação
- Amplamente utilizado
- Integração com várias plataformas
- Incluído no SpringDoc

Swagger

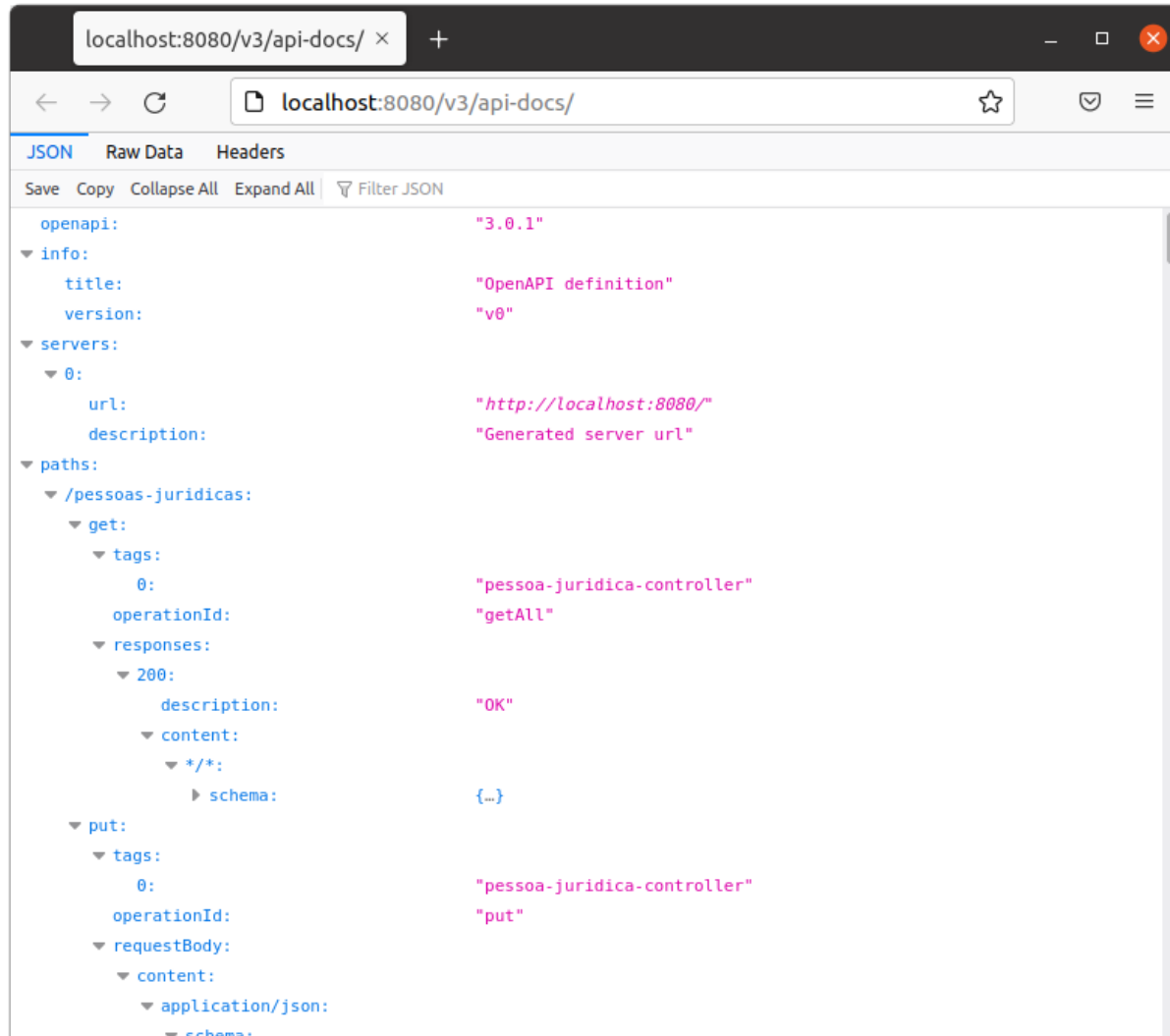
- O acesso a documentação gerada pelo Swagger pode ser feita através URL:
 - <http://localhost:8080/swagger-ui.html>
- Para customizar essa esse caminho, indique-o no arquivo application.properties
 - `springdoc.swagger-ui.path=/swagger-ui-custom.html`

Liberando acesso ao Swagger no SpringSecurity

- Em SecurityConfig.java libere acesso aos caminhos do Swagger

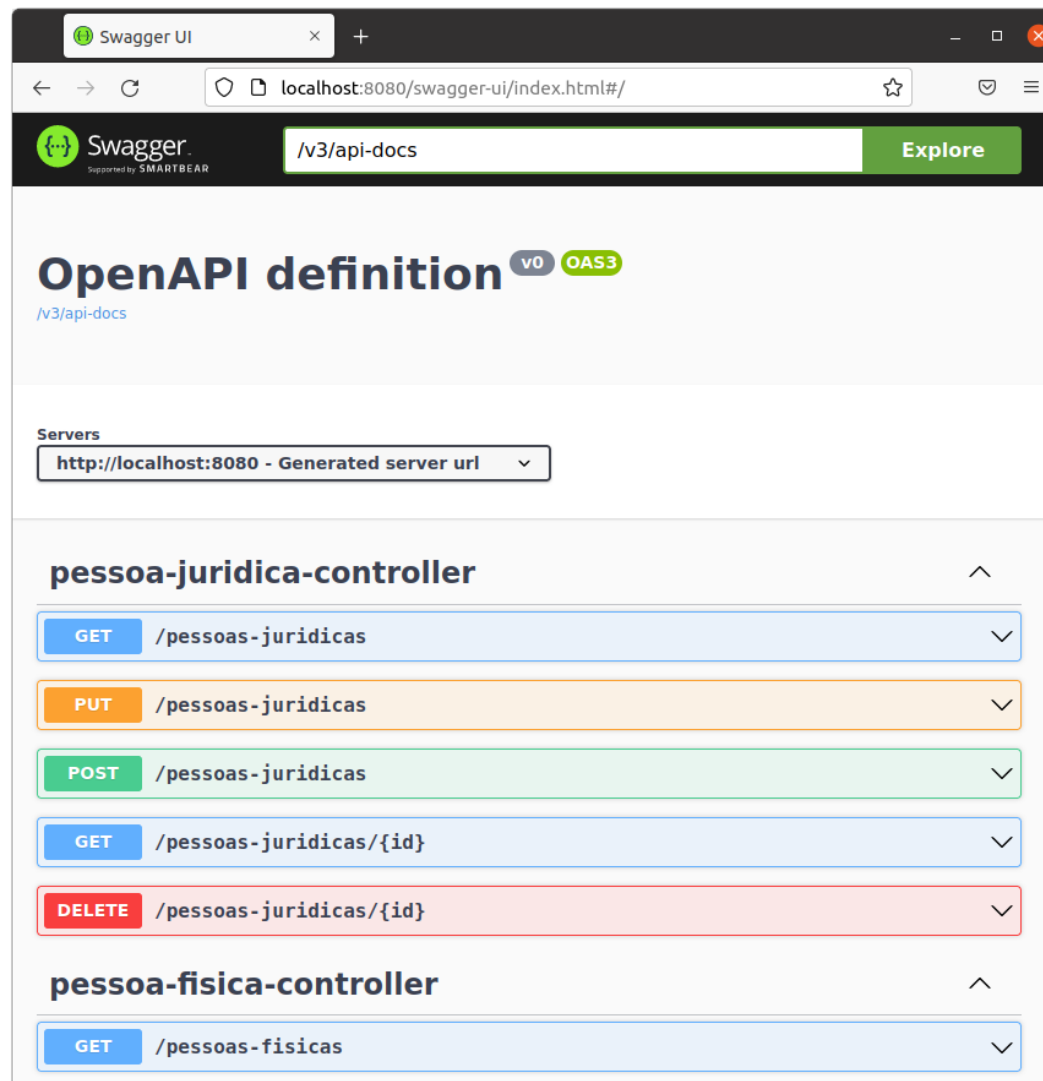
```
@Override
protected void configure(HttpSecurity http) throws Exception
{
    ...
    http.authorizeRequests()
        .antMatchers(HttpMethod.GET, PUBLIC_MATCHERS).permitAll()
        .antMatchers("/v3/api-docs/**", "/swagger-ui/**",
                     "/swagger-ui.html").permitAll()
        .antMatchers(HttpMethod.POST, PUBLIC_MATCHERS_POST)
        .permitAll().anyRequest().authenticated();
    ...
}
```

Documentação Padrão do OpenAPI



```
openapi: "3.0.1"
info:
  title: "OpenAPI definition"
  version: "v0"
servers:
  0:
    url: "http://localhost:8080/"
    description: "Generated server url"
paths:
  /pessoas-juridicas:
    get:
      tags:
        0: "pessoa-juridica-controller"
      operationId: "getAll"
      responses:
        200:
          description: "OK"
          content:
            */*:
              schema: {...}
    put:
      tags:
        0: "pessoa-juridica-controller"
      operationId: "put"
      requestBody:
        content:
          application/json:
            schema:
```


Documentação Padrão do Swagger



Personalizando as Informações da API

- Crie uma classe de configuração do Swagger com um método que devolve um objeto

OpenAPI

@Configuration

```
public class SwaggerConfig {
```

```
    @Bean
```

```
    public OpenAPI filmesOpenAPI() {
```

```
        return new OpenAPI().info(new Info())
```

```
            .title("API do Projeto Finanças da FATEC")
```

```
            .description("Esta API é utilizada na disciplina  
Desenvolvimento para Servidores-II")
```

```
            .version("v0.0.1")
```

```
            .contact(new Contact()
```

```
                .name("Ciro Cirne Trindade").email("ciroct@gmail.com"))
```

```
            .license(new License()
```

```
                .name("Apache 2.0").url("http://springdoc.org")));
```

```
    }
```

```
}
```

API do Projeto Finanças da FATEC v0.0.1 OAS3

[/v3/api-docs](#)

Esta API é utilizada na disciplina Desenvolvimento para Servidores-II

[Contact](#) [Ciro Cirne Trindade](#)

[Apache 2.0](#)

Definindo Códigos de Retorno Personalizados

- Isso também pode ser feito através da anotação `@ApiResponse` nos *end points* da API

```
@ApiResponse(value = {  
    @ApiResponse(responseCode = "200",  
        description = "Retorna a lista de categorias"),  
    @ApiResponse(responseCode = "403",  
        description = "Você não tem permissão para  
acessar este recurso"),  
    @ApiResponse(responseCode = "500",  
        description = "Foi gerada uma exceção"),  
})
```

Definindo o content-type das respostas

- O Swagger por padrão define que o "Response content type" é o valor `"*/*"`
- Para alterar este ponto, basta indicar no seu *end point* o tipo do conteúdo que ele produz com o atributo `produces`

```
@GetMapping(value =("/{id}"),  
            produces = "application/json")  
public ResponseEntity<?> get(@PathVariable("id")  
                             Long id) {  
    ...  
}
```

Descrivendo o *end point*

- Também é possível utilizar a anotação `@Operation` para descrever o end point
- Por exemplo:

```
@Operation(summary = "Retorna a lista de categorias")
```

Escondendo um controlador da documentação

- Se você não quiser exibir a documentação de um controlador REST, basta anotá-lo com `@Hidden`

```
import io.swagger.v3.oas.annotations.Hidden;

@RestController
@RequestMapping("/categorias")
@Hidden
public class CategoriaController implements
    ControllerInterface<CategoriaDTO>{
```

Referência

- SWAGGER. Disponível em:
<https://swagger.io/specification/>