Curso: Spring Boot com Ionic - Estudo de Caso Completo

https://www.udemy.com/user/nelio-alves

Prof. Dr. Nelio Alves

Capítulo: Ferramentas

Objetivo geral:

- Introduzir algumas ferramentas úteis para desenvolvedores Spring Boot

Preparação para a o capítulo

Checklist

- Clonar o projeto do Github
- Testar o projeto
  - Se der problema na porta, tente mudar a porta
  - Se der problema no H2, tente fazer a configuração alternativa

Documentação com Swagger - Parte 1

ATENÇÃO: vamos criar um branch "swagger"

Considerações sobre a ferramenta:

- Objetivo: gerar documentação da API automaticamente a partir do projeto. Automatiza o processo de geração e atualização da documentação.
- Amplamente utilizado.
- Possui integração com várias plataformas.
- Site: https://swagger.io
- Swagger: responsável por processar e extrair informações
- Swagger UI: responsável por gerar a UI da documentação

Checklist:

- Incluir as dependências. Vamos usar a implementação da Springfox:

```
<dependency>
	<groupId>io.springfox</groupId>
	<artifactId>springfox-swagger2</artifactId>
	<version>2.7.0</version>
</dependency>

<dependency>
	<groupId>io.springfox</groupId>
	<artifactId>springfox-swagger-ui</artifactId>
	<version>2.7.0</version>
</dependency>
```

- Criar classe de configuração

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

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

- Em SecurityConfig, incluir a configuração para liberar o acesso aos caminhos do Swagger:

```
@Override
public void configure(WebSecurity web) throws Exception {
    web.ignoring().antMatchers("/v2/api-docs", "/configuration/ui", "/swagger-resources/**", "/configuration/**",
"/swagger-ui.html", "/webjars/**");
}
```

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

Documentação com Swagger - Parte 2

1) Selecionar pacote do projeto:

```
.apis(RequestHandlerSelectors.basePackage("com.nelioalves.cursomc.resources"))
```

2) Informações personalizadas

- Criar método para retornar um objeto ApiInfo:

```
private ApiInfo apiInfo() {
     return new ApiInfo(
               "API do curso Spring Boot",
               "Esta API é utilizada no curso de Spring Boot do prof. Nelio Alves",
               "Versão 1.0",
               "https://www.udemy.com/terms",
               new Contact("Nelio Alves", "udemy.com/user/nelio-alves", "nelio.cursos@gmail.com"),
               "Permitido uso para estudantes",
               "https://www.udemy.com/terms",
               Collections.emptyList() // Vendor Extensions
       );
}
```

ATENÇÃO AOS IMPORTS:

```
import springfox.documentation.service.Contact;
import springfox.documentation.service.ApiInfo;
```

- Incluir a chamada no bean Docket:

```
.apiInfo(apiInfo());
```

3) Personalizando mensagens globais

```
private final ResponseMessage m201 = simpleMessage(201, "Recurso criado");
private final ResponseMessage m204put = simpleMessage(204, "Atualização ok");
private final ResponseMessage m204del = simpleMessage(204, "Deleção ok");
private final ResponseMessage m403 = simpleMessage(403, "Não autorizado");
private final ResponseMessage m404 = simpleMessage(404, "Não encontrado");
private final ResponseMessage m422 = simpleMessage(422, "Erro de validação");
private final ResponseMessage m500 = simpleMessage(500,  "Erro inesperado");


.useDefaultResponseMessages(false)
.globalResponseMessage(RequestMethod.GET, Arrays.asList(m403, m404, m500))
.globalResponseMessage(RequestMethod.POST, Arrays.asList(m201, m403, m422, m500))
.globalResponseMessage(RequestMethod.PUT, Arrays.asList(m204put, m403, m404, m422, m500))
.globalResponseMessage(RequestMethod.DELETE, Arrays.asList(m204del, m403, m404, m500))


private ResponseMessage simpleMessage(int code, String msg) {
	return new ResponseMessageBuilder().code(code).message(msg).build();
}
```

4) Response headers personalizados

```
private ResponseMessage customMessage1() {

	Map<String, Header> map = new HashMap<>();
	map.put("location", new Header("location", "URI do novo recurso", new ModelRef("string")));

	return new ResponseMessageBuilder()
		.code(201)
		.message("Recurso criado")
		.headersWithDescription(map)
		.build();
}
```

5) Descrições personalizadas para os endpoints

```
@ApiOperation(value="Busca por id")
```

6) Mensagens de resposta específicas

```
@ApiResponses(value = {
	@ApiResponse(code = 400, message = "Não é possível excluir uma categoria que possui produtos"),
	@ApiResponse(code = 404, message = "Código inexistente") })
```