Introdução ao Uso de Serviços Web Usando REST

- Parte 4 -

Prof. Julio Cesar Nardi.

Esse tutorial é a Parte 4 da “Introdução ao Uso de Serviços Web Usando REST”. Ao seguir este tutorial, você já deve ter feito “Introdução ao Uso de Serviços Web Usando REST” partes 1, 2 e 3. Vamos usar o projeto Spring Tool Suite já criado quando da realização da Parte 3.

Neste tutorial, vamos abordar a geração de documentação de APIs (*Application Programming Interface*) de serviços. Esse é um assunto relevante, pois ao gerar documentação das APIs de serviço de um sistema: (i) membros da equipe de desenvolvimento/manutenção podem se beneficiar; e (ii) usuários externos podem usar tal documentação disponível para entender como acessar a API e, assim, estabelecer integração entre sistemas de informação distintos.

Uma vez gerada a documentação, é imprescindível mantê-la atualizada a fim de que os usuários dessa documentação possam, de fato, utilizá-la de maneira efetiva. Documentação desatualizada induzirá usuários ao erro.

Para abordarmos essa temática, vamos utilizar o *framework* Swagger/OpenAPI (https://springdoc.org/). Este é um dos *frameworks* mais populares para documentação de APIs.

Usando o mesmo projeto da Parte 3

Como mencionado usaremos o projeto já criado quando da realização da Parte 3 do tutorial “Introdução ao Uso de Serviços Web Usando REST”. Logo, abra este projeto para iniciarmos nosso trabalho. A estrutura geral de pacotes pode ser observada na Figura 1.

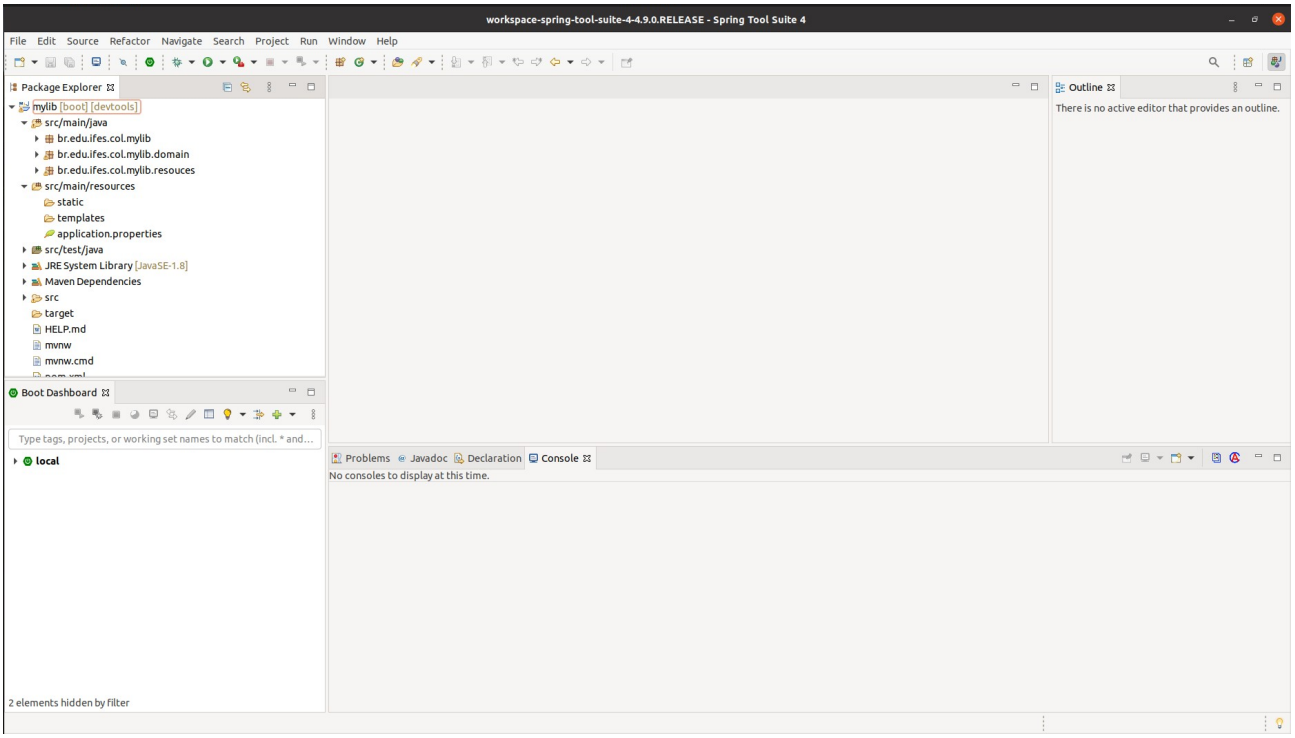

Figura 1: Estrutura geral de pacotes do projeto implementado no Tutotial Parte 3.

Configurando nosso projeto

Para incorporar o *framework* Swagger/OpenAPI em nosso projeto, vamos incluí-lo no arquivo pom.xml. Assim, inclua as seguintes linhas no referido arquivo:

```
<dependency>
	<groupId>org.springdoc</groupId>
	<artifactId>springdoc-openapi-ui</artifactId>
	<version>1.2.32</version>
</dependency>
```

Após isso, vamos testar se o framework está funcionando. Inicialize sua aplicação e acesse a URL http://localhost:8080/swagger-ui.html. A Figura 2 apresenta a tela inicial da documentação considerando o que o *framework* já gerou automaticamente.

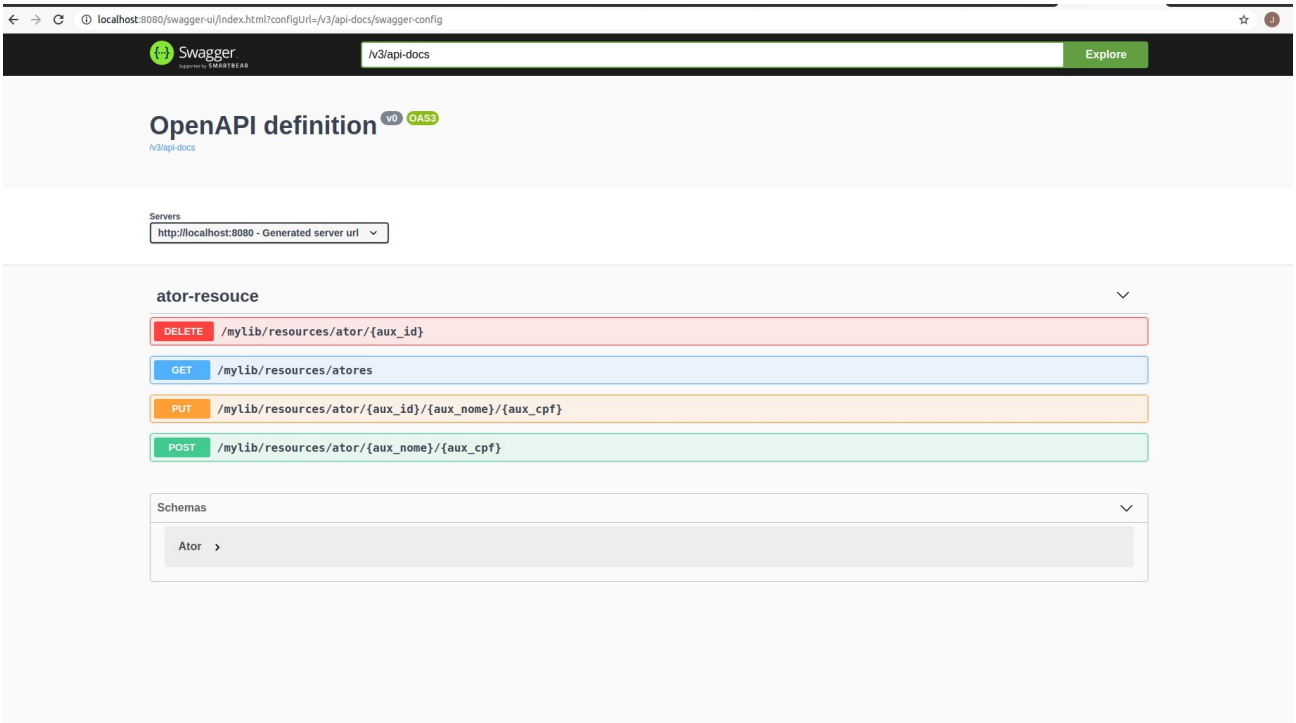

Figura 2: Tela inicial da documentação gerada automaticamente.

Observe que todas as APIs do projeto foram, automaticamente, detectadas (por meio das anotações utilizadas durante a implementações das partes 1, 2 e 3 do tutorial) e apresentadas na documentação. Assim, já temos uma documentação básica da APIs. Já podemos navegar por entre os links da página da documentação e verificar os detalhes de cada API.

Mesmo já possuindo uma documentação básica, vamos buscar fornecer mais informações à documentação de modo a torná-la mais ampla e útil. Assim, vamos fazer uso de anotações fornecidas pelo Swagger/OpenAPI e incorporá-las aos nosso código, conforme mostrado abaixo.

```
@Service
@RestController
@RequestMapping (produces = "application/json")
@Tag(name = "AtorResource", description = "Fornece serviços web REST para acesso
e manipulação de dados de atores.")
public class AtorResouce {

	@Autowired
	private AtorRepository repositorioAtores;
...
```

Você pode ainda detalhar cada operação da sua classe. Vamos usar a *tag* @Operation, conforme a seguir, para melhor explicar o funcionamento do método *obterAtores*.

```
...
	@GetMapping (value="/mylib/resources/atores")
	@Operation (description="Retorna todos os atores cadastrados.")
	public List<Ator> obterAtores(){

		return repositorioAtores.findAll();

	}
...
```

Vamos agora detalhar um pouco mais o método incluirAtor, conforme a seguir.

```
...

	@PostMapping (value="/mylib/resources/ator/{aux_nome}/{aux_cpf}")
	@Operation (
			description="Dados o nome e o cpf, cadastra um novo ator.",
			responses = {
					@ApiResponse(responseCode = "200", description =
"Caso o ator seja incluído com sucesso."),
					@ApiResponse(responseCode = "400", description =
"Caso não tenha sido possível realizar a operação.")
			}
			)
	public ResponseEntity incluirAtor(@PathVariable(value="aux_nome") String
nome, @PathVariable(value="aux_cpf") String cpf) {

...
```

Inclua, agora, anotações semelhantes nos demais métodos da classe a fim de oferecer uma documentação mais ampla.

Observe que o endereço *swagger-ui.html* é o endereço padrão oferecido pelo *framework* Swagger/OpenAPI. Caso queira alterar esse endereço, você pode acessar o arquivo *application.properties* e coloque, por exemplo `springdoc.swagger-ui.path=/mylibdoc.html`. Nesse caso, o novo endereço da página de documentação será http://localhost:8080/mylibdoc.html .

DICAS:

1. Uma lista das annotations pode ser econtrada em https://github.com/swagger-api/swagger-core/wiki/Swagger-2.X---Annotations.
2. Caso deseje, o conteúdo da documentação pode se basear na sua especificação de caso de uso / requisitos construída para o sistema. De todo modo, o ideal é que seja padronizado na sua empresa e dentro do projeto uma maneira de construír/descrever a documentação.

Agora você tem a documentação da sua API. Mais do que isso, você tem um ponto central onde todos os seus serviços web estão listados e documentados. Isso facilita muito a busca por serviços a serem (re)utilizados tanto por colaboradores internos à sua organização, quanto por usuários/parceiros externos que queiram integrar suas aplicações às de sua empresa.

Há outros *frameworks* e ferramentas para gerar documentações de API. Como dito, Swagger/OpenAPI é uma das opções e é uma opção muito utilizada na prática em empresas que desenvolvem orientado a serviço usando REST.

Bom trabalho.