03

Documentação da API com SpringFox Swagger

Transcrição

[00:00] Vimos no último vídeo que é importante documentar nossa API para que os clientes consigam entender, saber como funciona, fazer testes. O Swagger vai ser a ferramenta que vamos utilizar. Mas no caso, como vamos utilizar no nosso projeto? Tem um site chamado SpringFox, que é uma biblioteca que consegue documentar uma API REST utilizando o Swagger, sendo que esse projeto é desenvolvido em Java com Spring. Como nossa API REST foi desenvolvida em Java web com Spring, o SpringFox vai nos ajudar a documentar.

[00:54] Um detalhe importante. O SpringFox tem Spring no nome, mas não é uma biblioteca do Spring. É parecido com aquela outra biblioteca que nós usamos para monitoramento. Foi um pessoal que desenvolveu essa biblioteca baseada no Swagger para documentar aplicações utilizando API REST.

[01:20] Vamos para o Eclipse. O primeiro passo é baixar o SpringFox para o nosso projeto, adicionar como uma dependência no Maven. Temos que trocar o groupId para io.springfox. O artifactId vai ser springfox-swagger2. Como esse não vem do Spring Boot, tenho que dizer qual a versão. A versão que vamos utilizar no curso é a 2.9.2. Além disso, precisamos de outra dependência, porque o Swagger vai gerar a documentação da nossa API, vai ler as classes, os endpoints, e disponibilizar esses endpoints, essa documentação através de um site. Nós vamos ter um endpoint, um endereço onde conseguimos navegar pela documentação. Quem faz essa geração da interface é outra dependência.

[02:40] Essa dependência também é do SpringFox, mas o artifactId é springfox-swagger-ui. Cuidado que esse não tem o 2. A versão é a mesma. Vou salvar, o Maven vai baixar as dependências.

[03:06] Ele não vem habilitado por padrão no projeto. Como de costume, para habilitar coisas no projeto, abrimos nossa classe main e colocamos a anotação @EnableSwagger2.

[03:30] O Swagger precisa que criemos uma classe com algumas configurações, precisamos dizer algumas coisas da nossa API. Por exemplo, qual o pacote raiz onde ele vai começar a ler as classes, porque às vezes não quero que ele leia todas as classes do meu projeto. Qual a url base, porque também posso ter algumas URLs restritas. Dentre outras coisas.

[03:53] Como essas configurações são feitas via código Java, eu vou criar uma nova classe no projeto, e vou jogar essa classe no pacote configure, mas vou criar um subpacote chamado Swagger e uma classe com o nome Swagger configurations.

[04:16] Vou colocar o @Configuration em cima da classe, que é aquela anotação do Spring para carregar essa classe de configuração. Dentro dessa classe, preciso definir um bean e devolver um objeto do tipo docket. Vou chamar o método de forumApi. Esse método tem que estar anotado com o @Bean do Spring, para o Spring saber que estou exportando esse bean, que é o objeto do tipo docket.

[04:51] Dentro do método tenho que instanciar esse objeto docket e setar todas as informações que o SpringFox Swagger precisa para configurar nosso projeto. Vou colar o código, que é um pouco chato, vocês vão pegar ele pronto no exercício.

[05:14] Eu estou retornando um new docket. Na hora de dar new nessa classe você tem que dizer qual o tipo de documentação, que no caso é usando o Swagger2. Na sequência, .select, .apis, e aí tenho que dizer para ele qual o

basePackage. A partir de qual pacote ele vai começar a ler as classes do nosso projeto. No nosso caso não tenho restrições, quero que ele leia todas as classes do projeto.

[05:47] Na sequência tenho o .paths, e aí passamos quais endpoints o SpringFox Swagger precisa analisar. No caso, também não tenho nenhuma url restrita, então coloquei /**.

[06:07] Depois temos o .build. No final, só coloquei o detalhe do .ignoredParameterTypes(Usuario.class). Quero que ele ignore todas as URLs que trabalham com a classe usuário, porque senão, na tela do Swagger vai começar a aparecer os dados da senha do usuário. Não quero isso.

[06:42] Já está configurado o Swagger no projeto. Vou rodar nosso fórumApplication, ele vai inicializar. Quando eu rodar o projeto, o Swagger já vai rodar a classe, escanear os pacotes, entrar nos endereços da nossa aplicação, e ele gera essa documentação em um site. Para acessar esse site, entramos no localhost:8080/Swagger-ui.html. Esse é o endereço para abrir a documentação do Swagger.

[07:20] Deu o erro 403, porque como estamos com o Spring security configurado, preciso liberar o endereço do Swagger. Vou abrir a classe do securityConfigurations. Dentro dessa classe temos que liberar quais URLs queremos liberar e quais bloquear. Eu não liberei, ele caiu automaticamente em bloquear. Basta liberar.

[07:49] Tem um problema. Não é só esse endereço que precisamos liberar. Esse é o endereço raiz, mas quando entramos nele, o Swagger dispara algumas requisições para carregar alguns arquivos estáticos, como CSS, Java script e imagens. Vamos precisar liberar vários endereços. Mas não vamos fazer isso aqui.

[08:12] Lembra que tinha um terceiro método configure na classe SecurityConfiguration? Era o que recebe o webSecurity como parâmetro. Nes…

método configuro as coisas que não quero que o Spring security intercepte.

[08:28] Quero que o Spring security ignore as requisições para o Swagger. Não é para ele interceptar, ter login, autenticação para acessar a documentação da API. Tenho essa configuração copiada, só vou colar. A ideia é que dentro desse método coloco ignoring().antMatchers, e passo uma lista de string com todas as URLs que queremos ignorar. Precisamos liberar algumas URLs do Swagger. Por exemplo, /**html. São endereços que o Swagger chama quando entramos no navegador, naquela url para abrir a documentação. Depois no exercício vocês vão pegar isso certinho.

[09:28] Feito isso, salvei. Ele vai reiniciar. Acho que já consigo entrar no Swagger.ui.html. Aqui tem a documentação da nossa API. Ele encontrou três controllers. O AutenticacaoController, o HelloController e o TopicosController. Você pode navegar por essa documentação. Posso clicar em um controller e ele vai mostrar todos os métodos que têm mapeamento. Clicando no método, ele mostra todos os parâmetros, quais as respostas possíveis.

[10:50] Em cima, perto da url, tem o botão try it out, posso clicar nele e fazer o teste diretamente do Swagger. Além de ter essa documentação, esse site com um visual bacana, também consigo testar a API. Até se eu tiver uma equipe de testes na minha equipe de desenvolvimento, eles podem fazer o teste da API aqui.

[11:24] Descendo, tem o botão execute. Clicando, ele dispara e mostra o que foi devolvido. É bem interessante, dá para navegar pela documentação.

[11:48] Lembram do DELETE? Se eu testar, ele vai pedir para passar o id. A url para excluir um tópico é restrita, precisa ter autenticação. Ele também detectou a AutenticacaoController, que é nosso controller para se autenticar. Posso pegar o token e tentar novamente. Porém, onde eu mando os cabeçalhos? Não tem um lugar para isso, porque é um detalhe de segurança da nossa API e não ensinamos isso para o Swagger. Ele não sabe como se autenticar. Esse vai ser o assunto do próximo vídeo.