02

Introdução ao Interceptador

Transcrição

[00:00] Nesse vídeo nós vamos entender um pouco sobre Interceptor, que é um recurso do Spring. O objetivo dele é interceptar requisições do usuário. Nós vamos implementar uma funcionalidade que se encaixa bem nesse cenário e é bem interessante.

[00:15] O usuário faz requisições e as requisições caem nos nossos "Controllers", então requisição para "/home" tem "Controller", para "/ofertas", "/login" etc.

[00:26] E depois que você colocar o sistema em produção, pode ser que uma das necessidades seja otimizar a aplicação para ela consumir menos recursos, ou até para ela responder melhor às requisições, às interações do usuário, caso ela fica lenta ou caso tenhamos problema de performance.

[00:47] Para conseguirmos atuar em cima da performance da aplicação e trazermos uma implementação melhor, quebrarmos em um microsserviço, implementar um *cache*, alterar banco de dados, otimizar a implementação etc. - você precisa levantar informações sobre o uso da sua aplicação.

[01:05] Quantas requisições estão sendo feitas para "/home", para ofertas? Qual o tempo dessas requisições? E coisas desse tipo, que vão ser necessárias para decidirmos como otimizar ou priorizar a otimização da nossa aplicação. Para isso, precisamos guardar informações sobre os acessos dos usuários.

[01:29] Imagine que temos uma base de dados de acesso - seja memória, seja em banco mesmo - e cada funcionalidade nós conversamos, nós invocamos

essa funcionalidade de acesso, que ela vai registrando os acessos do usuário. É o que eu falei, o tempo de acesso, quantidade de acessos que tem essa funcionalidade, no decorrer do dia etc.

[01:52] E para cada funcionalidade que tivermos na nossa aplicação, vamos ter que plugar essa funcionalidade específica com o registro de acessos.

[02:03] Esse cenário que eu estou mostrando no slide nos induz a imaginar que todas as funcionalidades. Temos que implementar essa integração com acessos, para guardarmos os acessos do usuário.

[02:20] E claramente isso é ruim por manutenção e a própria implementação é ruim. Porque se você tiver várias funcionalidades na aplicação, você vai acabar mexendo em todas elas para poder plugar com acessos, pode gerar *merges*. E depois, para fazer a manutenção e alterar também essa funcionalidade, você vai ter que alterar em todos os lugares.

[02:45] Essa não é uma implementação muito boa, até porque acesso não é uma funcionalidade que faz parte da funcionalidade de "/home", que abre a página principal - assim como ela também não é a funcionalidade que faz parte da funcionalidade de ofertas ou mesmo de login, de autenticação. Ela é outra funcionalidade.

[03:04] Temos a funcionalidade "/home", temos a funcionalidade "/ofertas", temos a funcionalidade "/login" e temos a funcionalidade acesso. Essa é uma funcionalidade diferente, ela não deveria estar espalhada na nossa aplicação em todas as outras funcionalidades, o ideal é que consigamos isolar isso.

[03:22] Só que a funcionalidade de acesso precisa de uma informação, que normalmente nós conseguimos dentro das outras funcionalidades, que é o tempo de processamento, por exemplo. É informação de requisição e resposta, para conseguir saber quanto durou, quanto tempo, quantas vezes está sendo invocado. Só que existe um recurso no Spring, que é exatamente o assunto desse vídeo, que é o Interceptor.

[03:51] O Interceptor nós implementamos uma classe em Java mesmo, que implementamos integrado com o Spring e podemos mapeá-lo para ele poder interceptar as requisições. O legal dele é que você implementa uma única vez, em um único ponto. Fala para o Spring que ele vai interceptar as requisições todas - e ele intercepta tanto a requisição quanto a resposta.

[04:14] Fica até mais fácil e mais assertivo você conseguir pegar o tempo total de processamento, desde o momento em que você recebe a requisição quanto o momento em que você tem que responder a requisição. Então você consegue esse intervalo melhor.

[04:29] E quando você tem esse Interceptor em um ponto único da nossa aplicação, esse Interceptor é quem vai acessar a base de dados de acessos. Então a nossa funcionalidade de acesso vai estar aqui sendo invocada pelo Interceptor.

[04:42] Então não precisamos dessa base de dados aqui, do uso da funcionalidade de acessos pelas outras funcionalidades. Nós conseguimos a informação de quantas vezes a "/home" é chamada, qual o tempo que ela demora para processar.

[05:01] Por exemplo: "/ofertas" é invocado muito mais vezes do que as outras páginas, só que a "/home" está demorando mais para processar. Então agora com essa funcionalidade nós conseguimos levantar esse tipo de informação e decidir onde vamos atuar para otimizar, para implementar o *cache* etc., para melhorar a performance da nossa aplicação.

[05:23] E é isso que vamos aprender na prática no próximo vídeo. Até lá!