04

Invalidando o cache

Transcrição

[00:00] Continuando então. No último vídeo aprendemos como configurar e utilizar o cache no nosso projeto e colocamos o cache na funcionalidade de listagem de tópico. Inclusive, vimos que o Spring consegue diferenciar se o método já foi chamado de acordo com os parâmetros e valores desses parâmetros.

[00:18] Porém tínhamos aquele problema. Se alguém cadastrar, excluir ou alterar um tópico no projeto, o cache não vai ser atualizado automaticamente. Na próxima consulta para listagem eu teria uma informação desatualizada. Só viria os resultados daqueles três caches que tenho cadastrados no banco de dados, mas o novo tópico que foi cadastrado, excluído ou alterado eu ainda não teria a informação, porque o cache estaria lá intacto.

[00:48] Precisamos avisar para o Spring que quando um novo tópico for cadastrado, excluído ou alterado, ele precisa atualizar o cache, porque senão o usuário vai continuar recebendo uma informação desatualizada. Isso pode ser um problema.

[01:02] Para fazer isso também é bem tranquilo. Tenho o tópico controller aberto. Vou procurar meu método salvar. Aqui, faço uma alteração nos tópicos. Quando esse método for chamado, quero que o Spring limpe aquele cache que chamei de lista de tópicos. Para dizer isso para o Spring é bem fácil. Em cima do método você coloca a anotação @CacheEvict, para dizer que quero que o Spring limpe determinado cache.

[01:34] Mas como posso ter vários caches no projeto, preciso dizer para ele qual dos caches ele precisa limpar. Por isso, quando colocamos o @Cacheable, passamos no value um lista de tópicos. Na hora de fazer o @CacheEvict, preciso dizer no value qual dos caches quero limpar. Quero limpar o que eu chamei de lista de tópicos.

[02:02] Tem outro parâmetro para limpar todos os registros. No nosso caso sim, quero limpar todos os registros, para ele atualizar tudo e deixar o cache zerado de novo. Vou salvar. E aí vamos fazer um teste no Postman.

[02:26] Vou fazer uma requisição para o nosso endpoint/tópicos, sem paginação nem nada. Vou disparar a requisição. Trouxe os três registros. No console do Eclipse ele disparou o select. Vou limpar e fazer uma nova chamada. Veio os três registros, e ele não fez o select. Agora, vou fazer o teste do cadastro. Vou cadastrar um novo tópico e ver se o Spring vai invalidar aquele cache. Vou trocar a requisição para POST, a url é /tópicos. Mas lembre-se que nos cabeçalhos tenho que passar o content type. Na aba body, tenho que passar no raw o JSON com o tópico que quero cadastrar.

[03:55] Vou disparar a requisição. Voltou 201. O registro foi criado. Vamos dar uma olhada no Eclipse. Ele disparou um select para carregar o curso, já que estou mandando só o nome do curso. Fez o insert. Vou limpar o console. E agora vou disparar a requisição GET, para ler os tópicos. Na teoria, o Spring deveria disparar o select, porque ele invalidou o cache, então na próxima chamada ele tem que recarregar essa lista do banco de dados em memória. Vou disparar a requisição. Ele realmente disparou um select para carregar os tópicos do banco de dados.

[04:48] Deu certo a invalidação do cache. Toda vez que eu chamei o salvar, se um cliente fizer a chamada, o Spring vai limpar o cache e na próxima chamada para o lista ele vai recarregar a lista do banco de dados em memória.

[05:03] Eu preciso copiar essa linha, porque não só na hora de salvar, mas na hora que atualizo um tópico e quando excluo um tópico do banco de dados

preciso invalidar o cache, porque mudou as informações que estavam guardadas naquele cache. Em cima do método remover e atualizar, coloquei aquele @CacheEvict.

[05:27] Vou testar no Postman, disparando uma requisição DELETE para /tópicos/3. Vou disparar uma requisição GET para trazer a lista de tópicos. Ele fez o select certinho. Ele não trouxe o quatro porque quando coloquei o @CacheEvict e salvei, ele reiniciou o projeto. Toda vez que ele reinicia, ele limpa o banco e cria os registros que estão no data SQL, que são só os três registros. E disparou o select quando fiz a consulta. Ou seja, ele realmente invalidou na hora de excluir, e na listagem fez a chamada ao método para carregar os registros.

[06:26] Se eu fizer uma nova chamada para o lista, veio só o um e o dois, e não fez mais a consulta no banco de dados. O cache funcionou na hora de excluir. Com isso, consegui utilizar cache na listagem, mas no cadastro, alteração e exclusão consegui invalidar o cache para não afetar os registros e para o cliente não ver uma informação desatualizada. Esse era o conteúdo do vídeo de hoje. No próximo vídeo vamos ter uma discussão sobre onde utilizar cache e onde não utilizar, e os problemas que isso pode me trazer.