06

Removendo tópicos

Transcrição

[00:00] Continuando nossa API. Fizemos a atualização no último vídeo, já estamos conseguindo atualizar um tópico. Para fechar o "*CRUD*", as quatro operações, falta a parte de exclusão (excluir um determinado tópico). Vai ser o mesmo procedimento, vou ter um novo método, copiar o `ResponseEntity`, e vou chamar esse método de `remover()`.

```
public ResponseEntity<TopicoDto> remover() {

}
```

COPIAR CÓDIGO

Quero remover um tópico específico, então na URL vai ter o parâmetro do `id` que vai chegar no método. Neste caso, como é exclusão, vamos mapear esse método com o `@DeleteMapping` (em lógica de exclusão, usamos o método `delete()` no REST). Na URL tem que ter o `"/{id}"`, para assim dizer, com a URL, qual o tópico que quero excluir de fato.

```
@DeleteMapping("/{id}")
public ResponseEntity<TopicoDto> remover(@PathVariable Long id) {

}
```

COPIAR CÓDIGO

[01:05] Qual vai ser a lógica? Vai chegar o `id`, tenho que pegar e excluir esse tópico (que tem esse `id`) do banco de dados. Não vão chegar outros parâmetros, não vou ter um novo *Form*. Vai ser tranquilo de implementar. Eu vou pegar o `topicoRepository`. Nele, tem um método chamado `deleteById()`. Você passa o `id` e ele dispara o DELETE no banco de dados.

```
@DeleteMapping("/{id}")
public ResponseEntity<TopicoDto> remover(@PathVariable Long id) {
    topicoRepository.deleteById(id);

}
```

COPIAR CÓDIGO

[01:38] Neste caso, qual vai ser meu retorno? Tenho que devolver um código "200 OK" (excluí um registro e a requisição foi processada com sucesso). Mas como eu o excluí do banco de dados, o recurso não existe mais. Eu não vou devolver nenhum corpo na resposta. Vou copiar o `return ResponseEntity.ok(new TopicoDto(topico))`, do método `atualizar()`, mas vou deixar o `ok()` vazio, já que só quero devolver o "200 OK".

```
@DeleteMapping("/{id}")
public ResponseEntity<TopicoDto> remover(@PathVariable Long id) {
    topicoRepository.deleteById(id);
        return ResponseEntity.ok();
}
```

COPIAR CÓDIGO

[02:05] Ele vai reclamar, porque o retorno do meu método é `ResponseEntity` e tem um `<TopicoDto>` nos *generics*, então vou apagá-lo. Não quero devolver nada no *generics*. Ele continua reclamando, porque falta o método `build()`. Ele seguirá reclamando porque o `ResponseEntity` tem um *generics*. Posso até colocar uma

interrogação ("?"), só para dizer que é um *generics*, mas não sei qual é o tipo, e não especifiquei nada.

```
@DeleteMapping("/{id}")
public ResponseEntity<?> remover(@PathVariable Long id) {
    topicoRepository.deleteById(id);
        return ResponseEntity.ok().build();
}
```

COPIAR CÓDIGO

[02:41] Teminado, vamos testar no Postman. Só para garantir, vou disparar primeiro "GET" para a nossa URL http://localhost:8080/topicos/3 (http://localhost:8080/topicos/3) e ele devolverá todos os tópicos que estão cadastrados no banco de dados. Logo, ele apresentará os tópicos "1", "2" e "3".


```
[
   {
      "id": 1,
          "titulo": "Dúvida",
          "mensagem": "Erro ao criar projeto",
          "dataCriacao": "2019-05-05T18:00:00"
   },
     }
        "id": 2,
            "titulo": "Dúvida 2",
            "mensagem": "Projeto não compila",
            "dataCriacao": "2019-05-05T19:00:00"
     },
     }
        "id": 3,
            "titulo": "Dúvida 3",
            "mensagem": "Tag HTML",
            "dataCriacao": "2019-05-05T20:00:00"
```

```
    }
]
```

COPIAR CÓDIGO

Agora, para testar o exlcuir, vou trocar o método para "DELETE" e vou excluir o tópico de id "2", então http://localhost:8080/topicos/2 (http://localhost:8080/topicos/2). Vou disparar a requisição. Voltou o status "200 OK", mas vazio. No corpo da resposta não tem nada. A princípio funcionou e excluiu.

[03:30] Vou testar de novo o GET para http://localhost:8080/topicos (http://localhost:8080/topicos) e verificar se voltará apenas o tópico de id "1" e o "3".

```
[
   {
      "id": 1,
          "titulo": "Dúvida",
          "mensagem": "Erro ao criar projeto",
          "dataCriacao": "2019-05-05T18:00:00"
     },
     }
        "id": 3,
           "titulo": "Dúvida 3",
           "mensagem": "Tag HTML",
           "dataCriacao": "2019-05-05T20:00:00"
     }
]
```

COPIAR CÓDIGO

Tudo certo. Nossa exclusão funcionou sem nenhum mistério. Ela é tranquila, não tem muita dificuldade.

[03:47] Só vamos arrumar um detalhe. No método `atualizar()`, que está no `TopicosController.java`, nós colocamos o `@Transactional`. Segundo o Spring Data,

a ideia é que todo método que tiver uma operação de escrita, ou seja, "salvar", "alterar" e "excluir", deveríamos colocar o `@Transactional` .

No "salvar" e "excluir" não precisamos. Mas por via das dúvidas é bom colocar no "salvar" (`cadastrar()`)' e no "excluir" (`remover()`) o `@Transactional` . Vai que troco de provedor ou banco de dados e para de funcionar. É bom garantir que esses três métodos rodarão dentro de uma transação.

[04:32] Com isso já tenho meus cinco métodos: `lista()` , para listar todos os tópicos; o `cadastrar()` , para cadastrar um novo tópico; o `detalhar()` , para trazer os detalhes de um único tópico; o `atualizar()` , para atualizar as informações de um tópico; e o `remover()` , para excluir um determinado tópico do meu banco de dados. E assim fechamos nossa API REST com as operações CRUD da nossa entidade tópico.

[05:04] Na próxima aula vamos ver como fazer um tratamento, tanto no método `detalhar()` , quanto no `atualizar()` e `remover()` , no caso de chegar um `id` inválido. Se ele passar um `id` de um tópico que não existe no banco, vai dar um erro. Na próxima aula vamos ver como corrigir isso e como fazer o tratamento adequado.