06

Testando o cadastro com Postman

Transcrição

[00:00] Continuando então. Já implementamos a lógica para cadastrar o tópico e, na última aula, fizemos uma mudança para não devolver o código 200. Também utilizamos as boas práticas do REST, porque a ideia é devolver o 201, com cabeçalho *Location* e corpo da resposta sendo uma representação do recurso que acabou de ser criado. Agora só falta fazer o teste.

[00:27] O problema é que, como nossa requisição é do tipo POST, `@PostMapping`, não consigo testar diretamente pela barra de endereços do navegador. Vamos ter que usar alguma ferramenta que consiga disparar requisições para testar a API REST. Tem várias ferramentas que fazem isso (no próprio browser temos alguns plugins para fazer teste em API REST), mas nós vamos usar o **Postman**. Se trata de uma ferramenta gratuita para multiplataforma, você instala e consegue fazer testes da API.

[01:08] Quando você abrir o site, você vai criar um projeto. Existe uma tela ("My Workspace") para criar sua requisição. Você vai dizer para ele como vai ser a requisição. Localizando "Untiled Request" na parte superior esqueda da tela, logo abaixo, no primeiro campo você diz qual o método (no caso, "GET"), e no segundo campo, o endereço da sua API. No nosso caso, por exemplo, o endereço é http://localhost:8080/topicos (http://localhost:8080/topicos). Posso até testar, pressionando o botão "Send" e ele chamará nosso primeiro *endpoint*.

[01:40] Ainda nessa tela, mais abaixo na aba "Body", ele mostra o corpo da resposta. Então, aparece o JSON corretamente com os três tópicos. Mas eu não quero testar o GET e, sim, o POST. Quero fazer o cadastro de um novo tópico.

Para simular no POST, lembre-se que a URL é a mesma, "/topicos". Também precisamos trocar o método de "GET" para "POST". Mas aí, antes de disparar a requisição, preciso configurar duas coisas. A primeira é que na aba "Body", preciso levar o corpo da requisição, os parâmetros, as informações que preciso enviar para fazer o cadastro.

[02:23] Na aba "Body", vamos selecionar a opção "raw", porque vamos mandar o JSON diretamente. Seguindo, na caixa de texto, vou abrir chaves e, no JSON, preciso passar, entre aspas, o nome do campo. Por exemplo, título `"titulo":"Dúvida Postman",` . Também precisamos mandar a mensagem, então, `"mensagem":"Texto da mensagem",` . A terceira informação é o nome do curso, isto é, `"nomeCurso":"Spring Boot"` (precisamos indicar um valor que existe no nosso banco de dados e o que temos lá é o "Spring Boot").

```
{
    "titulo":"Dúvida Postman",
        "mensagem":"Texto da mensagem",
        "nomeCurso":"Spring Boot"
}
```

COPIAR CÓDIGO

[03:19] Esse é o corpo da requisição (quero testar uma requisição do tipo POST levando esse JSON como parâmetro). Estamos trabalhando com JSON, só que, no nosso código, `TopicosController.java` , recebemos o `Form` , que é um DTO, e aí o Spring pega o JSON que está vindo na requisição - no `@RequestBody` da requisição - e converte para um objeto `TopicoForm` .

[03:48] Além disso, ainda preciso definir um cabeçalho. Ppara isso, existe a aba chamada "Headers". Como estou mandando um JSON, preciso avisar para o servidor que o corpo da requisição está no formato JSON. Preciso indicar isso por um cabeçalho chamado "Content-Type", que eu seleciono abrindo a aba "Headers". Desta forma, na chave, "*Key*", do "Headers" eu selecionarei "Content-Type" e no "Value", selecionarei "application/json".

[04:20] O cabeçalho "Content- Type" é para o cliente dizer qual o conteúdo que está sendo enviado. E aí, no nosso caso, passamos "JSON".

[04:30] Com tudo feito, posso disparar a requisição pressionando o botão "Send". Dando tudo certo, na aba "Body", que está no centro esquerdo da tela, ele mostra o que foi devolvido pelo servidor. Estamos devolvendo o 201, que devolve no corpo da resposta uma representação do recurso que acabou de ser criado.

```
{
    "id": 4
        "titulo": "Dúvida Postman",
        "mensagem": "Texto da mensagem",
        "dataCriacao": "2019-05-14T14:29:01.6623906"
}
```

COPIAR CÓDIGO

Lembre-se que o nosso banco tinha três registros. Nesse caso, temos também `"id": 4` , portanto, realmente funcionou, ele criou um quarto registro. O título é `"Dúvida Postman"` , a mensagem é o `"Texto da mensagem"` e a data de criação é a atual, `"2019-05-14T14:29:01.6623906"` .

[05:14] Se olharmos um pouco acima do código, na parte centro direita da tela, encontraremos "Status" marcando "201 Created". Ou seja, ele realmente devolveu 201. Na mesma altura, lado esquerdo, temos a aba "Headers" onde encontraremos os cabeçalhos que foram devolvidos do servidor.

"**Location** -> http://localhost:8080/topicos/4 (http://localhost:8080/topicos/4)

Content-Type -> application/json:charset=UFT-8

Transfer-Encoding -> chunked

Date -> Tue, 14 May 2019 17:29:01 GMT"

O primeiro que aparece é o cabeçalho *Location* que havíamos comentado e com o endereço correto: http://localhost:8080/topicos/4 (http://localhost:8080/topicos/4), sendo que o "4" se refere ao "id" do recurso que acabamos de criar. Só não conseguimos chamar esse endereço, porque ainda não mapeamos ele.

[05:42] Funcionou. O Postman é uma ferramenta que consigo utilizar para fazer testes da API REST. Posso testar qualquer tipo de requisição (GET, POST, PUT e DELETE). A partir daqui, vamos utilizar sempre o Postman para fazer os testes da nossa API.

[05:58] Na próxima aula vamos fazer o seguinte teste. Quando mandamos o JSON, digitei as informações, mas e se eu não digitar nada? Se eu mandar só um JSON vazio? Se eu mandar só o título ou só a mensagem? Não fizemos nenhum tipo de validação. Na próxima aula vamos aprender como fazer isso, para garantir que os campos sejam informados.