04

Primeiro Endpoint da API

Transcrição

[00:00] Continuando, então, nosso treinamento. Já temos as classes de domínio da nossa aplicação. Com isso, podemos de fato construir nosso primeiro endpoint, o endereço que vai devolver a lista com todos os tópicos que estão cadastrados no sistema.

[00:17] Para fazer isso, vou precisar de um controller mapeando o endereço, que, quando for chamado, devolve a lista com os tópicos. Vou acessar o pacote "controller". Em seguida, vou criar uma nova classe, "Class", selecionando com o comando "Ctrl + N". O nome da classe vai ser "TopicosController".

[00:37] Para dizer que é um controller do Spring, escrevemos `@controller`. Agora é só importar a anotação do Spring `org.springframework.stereotype.Controller;`. Dentro do código, temos que ter um método mapeando algum endereço. Por isso, vou criar o método `public ? lista() {` ("public", seguido do sinal de interrogação para indicar o que vamos devolver e ainda não foi definido. Depois, o nome do método, `lista()`).

```
package br.com.alura.forum.controller;

import org.springframework.stereotype.Controller;

@Controller
public class TopicosController {


    public ? lista() {
```

```
        }

    }
```

COPIAR CÓDIGO

[01:06] Em cima desse método, preciso colocar o endereço, qual é o mapeamento desse método: `@RequestMapping("/topicos")` . Quando eu digitar no navegador a URL http://localhost:8080/topicos (http://localhost:8080/topicos) (que é o endereço do servidor mais a barra ("/") e "topicos") o Spring sabe que é para chamar esse método.

```
package br.com.alura.forum.controller;

import org.springframework.stereotype.Controller;

@Controller
public class TopicosController {

    @RequestMapping("/topicos")
    public ? lista() {

        }

}
```

COPIAR CÓDIGO

[01:22] Nesse método, a lógica vai ser: tenho que carregar a lista com todos os tópicos e devolver para quem tiver feito a chamada. No outro exemplo, do "Hello World!", tínhamos colocado uma string, `public String hello()` , e estávamos devolvendo só a epxressão "Hello Word!", `return "Hello World!";` . Mas, nesse caso, quero devolver não uma string, mas, uma lista com os tópicos. É justamente esse o retorno. Vou usar o `List` do `java.util` , e vai ser um lis

de tópico, `List<Topico>` , que é nossa classe de domínio.Agora, vou só importar o `java.util.List` (o list vem do `java.util` e o tópico é do nosso pacote de modelo).

```
package br.com.alura.forum.controller;

import java.util.List;

 @Controller
 public class TopicosController {

       @RequestMapping("/topicos")
       public List<TopicoDto> lista() {

         }

    }
```

COPIAR CÓDIGO

[02:03] Ele dá um erro de compilação porque preciso devolver uma lista com os tópicos. Nessa aula, ainda não vamos usar um banco de dados. Não vamos fazer a consulta no banco de dados, vamos só criar tópicos em memória. Instanciar objetos tópicos, guardar numa lista e devolvê-los como resposta. Depois, vamos ter acesso ao banco de dados utilizando JPA.

[02:28] Para esta aula, o que eu preciso é criar alguns objetos tópicos. Então, eu vou instanciar alguns objetos.

```
Topico topico = new Topico(titulo, mensagem, curso);
```

COPIAR CÓDIGO

[02:43] Para não ter que setar os atributos do tópico - por exemplo, título e mensagem, linha por linha - vou criar um construtor que já recebe os atribut

como parâmetro. Na hora de instanciar um tópico, preciso passar para ele, como parâmetro: um título,por exemplo, `"Duvida"` ; uma mensagem, que será o conteúdo da dúvida, `"Duvida com Spring"` ; e um curso, que é nossa classe de domínio, `new Curso()` . Então, preciso instanciar um objeto `Curso` . Vou importar a classe `Curso` e ele vai reclamar, porque, na hora de dar `new` no `Curso` , eu também criei um construtor onde já passo o nome do curso e a categoria do curso.Então, o nome do curso será "Spring" e a categoria desse curso será "Programação".

```
Topico topico = new Topico("Duvida", "Duvida com Spring",
new Curso("Spring", "Programação"));
```

COPIAR CÓDIGO

[03:44] Agora está compilando a hora de instanciar o objeto. Só está dando erro porque preciso devolver uma lista de tópicos. Para simplificar, tem uma classe do Java chamada `Arrays` . Essa classe tem um método `asList()` , `Arrays.asList()` , um método estático, em que você pode passar vários objetos separados por vírgula. Ele transforma isso em uma lista e te devolve ela com todos os objetos.

```
return Arrays.aslist(topico, topico, topico);
```

COPIAR CÓDIGO

[04:16] No caso, eu passei três vezes o próprio objeto "topico", não tem problema. Ele vai criar uma lista com três elementos. Agora, ele já está compilando tudo certinho. Já estou devolvendo a lista com esses três tópicos, que na verdade é o mesmo tópico.

[04:34] Vou testar abrindo o navegador e acessando a URL http://localhost:8080/topicos (http://localhost:8080/topicos), que é endereço que eu mapeei. Você vai ver um "Error 404 Not Found". Ele está dizendo que

não entrou o endereço. Voltando ao nosso projeto, vemos que faltou o detalhe, a anotação `@ResponseBody`, que é para dizer para o Spring que não vou navegar para uma página (não é uma aplicação web tradicional), vou só devolver (pego o retorno desse método e devolvo direto para o navegador). Agora, retorno ao navegador e aperto "F5".

```
package br.com.alura.forum.controller;

import java.util.Arrays;

 @Controller
 public class TopicosController {

       @RequestMapping("/topicos")
         @ResponseBody
       public List<Topico> lista() {
            Topico topico = new Topico("Duvida", "Duvida
com Spring", new Curso("Spring", "Programação"));

                return Arrays.asList(topico, topico,
topico);
                }

        }

    }
```

COPIAR CÓDIGO

[05:15] Mesmo assim, o "Error 404 Not Found" continua. Está dando problema porque eu criei esse código - essa classe, esse método - mas não lembrei de parar o servidor. Tenho que ir no "console" (pressionando o quarto ícone, localizado na barra vertical do lado direito da tela) e parar o servidor (após ter pressionado o botão "console", uma nova tela aparecerá. Nela selecionarei o ícone "pause", localizado na barra horizontal, parte superior. Depois, ao lado de "pause", pressionarei o ícone "X" para fechar). Na barra horizontal da parte

superior da tela (da esquerda para a direita, segunda linha) vou clicar no ícone "play", selecionar "1 ForumAplication", e, com isso, rodar de novo nosso "ForumApplication". É algo até chato, toda vez que faço uma alteração no código, tenho que parar o servidor e subir de novo. Depois, vamos ver como resolver isso, para não ter a necessidade de repetir esse processo. Continuando, vamos reiniciar o projeto e, no navegador, pressionar de novo "F5".

[05:47] Agora funcionou. Você percebe que, por padrão, o Spring pega o retorno (a lista) e devolve ela em um formato JSON. Na verdade, não é o Spring que faz isso. O Spring usa uma biblioteca chamada "Jackson". É o Jackson que faz a conversão de Java para JSON. O Spring usa o Jackson disfarçadamente, "por debaixo dos panos". Ele pegou a lista, `List<Topico>`, que foi devolvida, passou ela para o Jackson. O Jackson converteu para JSON, e ele pegou esse JSON e devolveu como uma string.

[06:23] Com isso, já temos nosso primeiro endpoint. Quando acessamos a URL http://localhost:8080/topicos (http://localhost:8080/topicos), ele chama o `TopicosController`, pega a lista em memória (a partir de `public`) com os três tópicos que foram instanciados em memória (`"Duvida", "Duvida com Spring", new Curso("Spring", "Programação")`), passa para o Jackson e devolve um JSON para o browser. Nosso primeiro endpoint está montado.

[06:44] Na próxima aula, vamos só simplificar o código. Para não ter que ficar repetindo o `@ResponseBody` em todos os métodos. Vamos ver que não é uma boa prática devolver a classe `topico` de domínio, até porque, na aula seguinte, vamos utilizar o JPA e essa classe vai virar uma entidade da JPA, por isso, não é uma boa prática devolver uma entidade da JPA no meu controller. E vamos simplificar algumas coisas para deixar o código mais simples. Espero vocês na próxima aula.