03

Usando Vue.js

Transcrição

Segue o JSON utilizado durante o vídeo:

```
[
    {
        "id" : 4,
        "nomeProduto" : "Wireless Earbuds TaoTronics",
        "valorNegociado" : null,
        "dataDaEntrega" : null,
        "urlProduto" : "https://www.amazon.com/TaoTronics-
SoundLiberty-79-Technology-
Waterproof/dp/B08397W1F7/ref=gbps_img_m-9_475e_ff9c5f1b?
smid=AAJVIARPZY8SB&pf_rd_p=5d86def2-ec10-4364-9008-
8fbccf30475e&pf_rd_s=merchandised-search-
9&pf_rd_t=101&pf_rd_i=15529609011&pf_rd_m=ATVPDKIKX0DER&pf_rd_

        "urlImagem" : "https://images-na.ssl-images-
amazon.com/images/I/51SgO0fSIuL._AC_SL1300_.jpg",
        "descricao" : "descrição Wireless Earbuds
TaoTronics",
        "status" : "AGUARDANDO"
    },
    {
        "id" : 3,
        "nomeProduto" : "ASICS Men's GEL Venture 5 Running",
        "valorNegociado" : null,
        "dataDaEntrega" : null,
        "urlProduto" : "https://www.amazon.com/ASICS-Gel-
Venture-5-M-Silver-Light/dp/B00NUZD2PW?
```

```
ref_=Oct_DLandingS_D_6340036f_60&smid=A2NEM58BFPMEIL",
        "urlImagem" : "https://images-na.ssl-images-
amazon.com/images/I/718OQVH7AnL._AC_UX625_.jpg",
        "descricao" : "descrição para ASICS ",
        "status" : "AGUARDANDO"
    }
]
```

COPIAR CÓDIGO

[00:00] Agora vamos montar a tela de oferta. Vou fechar aqui os outros e vou abrir a *home* de oferta, o arquvio `home.html`. Nessa *home* não vamos usar Thymeleaf para listar pedidos, vamos fazer isso em Vue.js.

[00:20] Aqui no finalzinho, antes de fechar a *tag* "body", eu vou criar uma área de "script", onde vamos implementar os nossos *scripts* com o Vue.js. Eu vou pesquisar aqui o Vue, na documentação, mostrar para vocês a documentação do Vue, que é muito boa e tem em português também.

[00:40] E aqui no início ele vai falando da proposta do Vue, ele explica até a pronúncia. Aqui tem um *script* de importação do Vue e o ruim dessa pronúncia de "Vue" é que às vezes você escreve assim, "*view*", sem querer.

[01:02] Vamos copiar isso daqui: `<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>`. Aqui está o *vue* e aqui está o *script*. Já importamos para a página.

[01:13] É importante importarmos para a página. E, o que acontece? Se eu começar a escrever código JavaScript utilizando as funções e objetos do Vue, pode ser que eu não consiga acessar; porque eu dependo que esse carregamento aconteça antes, ele precisa primeiro baixar o *script* para eu conseguir.

[01:36] Eu vou criar aqui uma função chamada de `onLoad()`, que vai ser invocada depois que toda a página for carregada. Como eu faço isso? Eu venh

no `<body`, coloco `function onload() {`.

[01:52] Recebe a invocação dessa função, ou seja, quando o *body* todo for carregado e vejo que os *scripts* estão dentro do *body*, ele vai chamar essa função para mim, ele vai executar esse código `<body onload="onLoad()">`, executar esse código, `function onLoad() {` e invocar essa função.

[02:06] Aqui dentro é que eu vou começar a implementar o meu código com o Vue.js. Aqui tem a primeira parte mais simples, que ele mostra como se fosse um *hello world*, `Hello Vue!`, e ele mostra dois pedaços de código: um é o HTML, quando trabalhamos com Java, e tem o Thymeleaf. Então é mais ou menos isso. Aqui é o JavaScript e o HTML.

[02:35] No back-end é mais ou menos a mesma coisa: tem o Java, onde você vai buscar os dados, inclusive; e você tem um HTML com Thymeleaf, que aqui não é Thymeleaf, a sintaxe dos elementos aqui é um pouco diferente.

[02:56] E o que ele faz? Ele diz que esse HTML está associado ou é gerenciado por essa nova aplicação Vue, e ele faz esse *bind* usando esse atributo chamado de `el` do Vue.

[03:14] O que eu vou fazer é colocar um ID para a nossa `div` *container*, que tem todo o conteúdo que queremos. Vou chamar de `ofertas`, `id="ofertas">`. E vou usar esse `id="ofertas">` aqui, porque eu vou copiar esse `var app = new Vue({` para cá. Vou apertar as teclas "Ctrl + Shift + F" para poder indentar isso aqui, jogar para cá.

[03:40] Essa é nossa função, `onLoad()` é nossa primeira aplicação Vue. Eu posso até colocar um ponto e vírgula (`;`) aqui no final. E o que eu vou fazer? Aqui na parte de "data" temos todo o conteúdo que vai poder ser acessado pelo HTML.

[03:56] Assim como temos no back-end, se eu abrir o "HomeController.java", por exemplo, nós buscamos informação no banco de dados e adicionamos

nesse `model.addAtribute("pedidos", pedidos);` . Tudo o que você adiciona no *model* é acessível pelo Thymeleaf. Tudo o que você adiciona no *data* é acessado pelo Vue.js nessa sintaxe, como se fosse o Thymeleaf. Então é legal, funciona de modo bem interessante.

[04:27] Vamos voltar para a `home.html` . O que eu vou fazer aqui? Eu vou dizer que eu não vou ter `message` , mas essa página, a fonte de informação principal dela é `pedidos` também - e esse `pedidos` é um *array* que ele vai buscar, que pode até 10 itens. Nós vamos percorrer agora esse *array* de `pedidos` e começar a montar os nossos formulários.

[04:52] Ou seja, vamos fazer esse mesmo `for each` , só que na sintaxe do Vue.js. Para cada pedido que eu encontrar nesse *array*, eu vou montar um formulário onde o usuário vai poder setar valor, data de entrega e gerar uma oferta. Ele vai clicar no botão lá e vai fazer uma requisição de oferta.

[05:09] Para conseguirmos já montar essa tela, antes de fazermos a integração com back-end, eu vou copiar esse conteúdo JSON e vou jogar aqui em `pedidos` . Em vez de ter esses colchetes fechados, eu vou jogá-lo para lá, e vou apertar o atalho "Ctrl + Shift + F" de novo.

[05:25] Ficou bem ruim, porque ele jogou lá para frente. Vou trazê-lo mais para trás, esses dados também vou trazer mais para trás. Assim está bom!

[05:39] O que eu vou fazer? Eu vou percorrer esse `pedidos` . Ele não está dentro de `data` ? Está, então eu posso acessá-lo a partir do HTML! Só que em vez de usar o `th:each` , que é do Thymeleaf, eu vou usar a sintaxe do Vue.js e vou olhar isso na documentação - para mostrar para você que a documentação é muito boa e para não fazer nada errado também.

[06:01] Aqui embaixo no site do Vue.js ele tem "Conditionals and Loops", onde ele mostra como usar o `if` e como usar o `for each` , que é o `v-for` . Vamos copiar esse `v-for=` e eu vou substituir por esse `th:each` .

[06:17] Só que a sintaxe aqui dentro, `"pedido : ${pedidos}"`, apesar de ser parecida, não é a mesma. Em vez de `:` é o `in`, e para acessar uma variável que está dentro do *data*, ele simplesmente acessa, sem precisar indicar isso como um Thymeleaf. Pronto, `v-for="pedido in pedidos">`! Isso já é o `for each` usando o Vue.js.

[06:38] Em cima ele usa o `th:text`. No Thymeleaf nós colocamos o `th:text` aqui e colocamos `pedido.nomeProduto` para aparecer no lugar desse conteúdo, `Nome do produto`. No Vue.js é diferente, toda vez que você quer mostrar o conteúdo que entra dentro de uma `li`, dentro de uma `div`, etc. você usa essa sintaxe de abre e fecha chaves.

[07:02] Vou abrir e vou fechar chaves e vou acessar, por exemplo, `pedido`, que é a variável que eu criei a partir do *loop* em cima de pedidos - que é uma variável que está dentro do *data*. Só para vocês não ficarem perdidos.

[07:18] Então `pedido.` - o que eu quero acessar? - Nome do produto, então `{{pedido.nomeProduto}}`. A primeira coisa que eu vou mostrar é o nome do produto no *card header* do Bootstrap, aqui do nosso formulário que estamos criando.

[07:33] O valor e data de entrega ainda não existem, então ao invés de ter um `span` mostrando esse valor, eu vou apagar isso e transformar em um `input`, que vai ser preenchido pelo usuário - tanto o valor quanto data de entrega, porque é exatamente isso a oferta, é quando o usuário pretende ou pode trazer o produto para o comprador e qual o valor que ele vai cobrar. Então está aqui!

[08:03] No produto não vamos colocar o `input` também, porque o usuário aqui não vai dizer qual produto, ele vai mostrar qual produto. Vamos usar aqui um `href` ao invés de um `input`, e o `href` tem depois no final. Então estou transformando esse `input` em um link.

[08:21] Em vez de usar o `th:value`, eu vou usar a sintaxe do Vue.js, que é `v-bind:`. O que é isso, url do produto? Eu tenho que colocar onde? No link. Eu tenho que colocar no atributo `href`, então: `v-bind:href`. Eu não preciso desse abre e fecha chaves, porque isso é do Thymeleaf. Então o `href` vai ser `"pedido.urlProduto"`.

[08:48] Esse `value` não existe aqui no link, não é aqui que colocamos. Eu vou colocar o texto do link, que fica aparecendo para o usuário, vai ser o nome do produto. Então através do `v-bind:href` eu estou acessando a url do produto para colocar no `href` do link. Estou usando essa sintaxe também, `{{pedido.nomeProduto}}`, do Vue.js, para mostrar para o usuário e para imprimir o nome do produto.

[09:16] Aqui na descrição, que é um `textarea`, eu também vou usar essa mesma sintaxe de cima, `{{pedido.nomeProduto}}`. Entre o abre `textarea` e o fecha `textarea`, eu vou colocar `{{pedido.descricao}}`.

[09:32] Veja que quando usamos `v-bind`, não precisamos abrir e fechar chaves, é só quando estamos colocando entre abre *tag* e fecha *tag*, tipo `div`, o `<a>` no link.

[09:45] E na url do produto aqui está `th:src` - e tem o `v-bind:src`, que é do Vue.js, mais uma vez. Eu vou apagar isso aqui também, `src=""`.

[09:59] Vou salvar e nós vamos ver. Será que vai funcionar? Vamos tentar acessar essa página *home*. O `/oferta` já vai abrir essa página aqui. Então, vamos ver se isso está funcionando? Vamos lá! `localhost:8080/oferta`. Se estivermos logados, acessaremos. Funcionou!

[10:23] Engraçado, esse aqui está como se fosse um *input*. É esse `form-control`. Deixe-me tirar isso, ficou esquisito. Ele já mostrou, ele mostrou esse aqui, "Wireless Earbuds TaoTronics", que são dois, e mostrou o "ASICS" aqui embaixo.

[10:38] O valor e o data de entrega são realmente *inputs* que preenchemos. O produto é um link, se eu clicar aqui em "Wireless Earbuds TaoTronics", ele vai abrir o link. Na descrição está a descrição que está no banco de dados, que eu coloquei. Isso eu não tinha cadastrado, mas eu cadastrei depois, aqui, no banco também, que estão aqui no JSON.

[11:06] No próximo vídeo nós vamos ver como fazemos para, ao invés de termos o conteúdo fixo aqui, estático, nós consumirmos esse *endpoint*.

[11:16] Já estamos montando a tela com o Vue.js, em vez de ser Thymeleaf, vimos que de certa forma tem uma boa semelhança. Agora vamos aprender a consumir o *endpoint* com o conteúdo mesmo para essa página, usando o Axios - que é uma biblioteca para fazer requisições Ajax.

[11:38] Vamos trocar isso aqui, que não são últimos pedidos, para finalizar esse vídeo. Não vai ser "Últimos Pedidos", mas "Faça sua Oferta". Atualizou, pronto!

[11:53] E outra coisa que podemos fazer é adicionarmos o link. Agora sim vamos poder finalizar a aula. Vamos no "base.html". Já temos o link para a *home*, o link para "Meus Pedidos" e eu vou criar um link para `/oferta`. É só isso! E vai exibir "Faça sua Oferta".

[12:25] Voltamos para o navegador. Esse é o link para oferta, link para os meus pedidos, usando o Thymeleaf, a *home* e aqui usando o Vue.js. Duas tecnologias aqui trabalhando e nós entendendo como funcionam essas duas.

[12:39] Até o próximo vídeo!