03

Percorrendo a lista na view

Transcrição

[00:00] O que eu vou fazer agora é - como nós não vamos implementar integração com banco de dados agora, nesse momento - é criar um pedido, “new Pedido()” e colocar informações nele para ele ser renderizado na “view”, ser processado na “view”.

[00:13] Então vou criar um pedido. Vou criar essa classe não dentro de “controller”, mas dentro de “model” e vou colocar os atributos dela. Então o que o pedido tem? Nós colocamos “Nome”, “Valor”, “Data de entrega”. Esse valor é o valor negociado daquele pedido.

[00:32] Então, “nomeProduto” nós sabemos que é um “private String nome”, vou colocar “nomeProduto”. Depois, “Valor”, “private BigDecimal valorNegociado;” e “private LocalDate dataDaEntrega;”.

[01:05] Pronto! O que mais? Nós temos a URL do produto, uma descrição e uma URL da imagem. Eu vou colocar “private String urlProduto;”, “private String urlImagem;” e por fim, uma descrição que vai estar associada ao pedido individualmente.

[01:30] Então se você fizer [CONTROL 3 GGAS], ele abre esse plugin do Eclipse para nós criarmos os “getters” e os “setters” automaticamente. Salvei o pedido. Agora vou preencher esse pedido. Vou adicionar em uma nova variável local com as informações. Por exemplo: “pedido.setNomeProduto()”.

[01:49] Qual é o nome do produto? Eu vou pegar na internet. Vou procurar ur produto na Amazon. Vamos abrir a Amazon. Tem o Kindle, vamos clicar nele.

Isso não é o produto em si. Vamos pegar um celular. Qual é nome desse produto? É esse aqui, “Note 8”.

[02:22] Então “nomeProduto()”, “private pedido.setUrlImagem()”, então vamos pegar a URL da imagem. Vamos copiar o endereço da imagem. E “pedido.setUrlProduto” em si que é essa página. Então apertamos as teclas “Ctrl + C” e colocamos aqui. Pronto! Então, “nomeProduto”, “Imagem”, “URLProduto”, a “Descrição, o “Preço” e a “Data”, posso colocar também. “.setDescricao(“uma descrição qualquer para esse pedido”)”. Beleza!

[03:05] Essas informações do “pedido” têm que ser enviadas para a tela, mas dentro de uma lista, por mais que seja só uma, aquela tela “home” que criamos sempre vai apresentar uma lista de pedidos. Se passarmos uma lista com um pedido, ela vai só o pedido mesmo.

[03:19] Então eu vou transformar esse pedido em uma lista com um pedido só usando essa classe para me ajudar, a “Arrays.asList” passando o pedido. Então esse aqui me retorna, se colocarmos em uma variável, uma lista de pedidos. Eu vou colocar “pedidos” e vou colocar isso no “model”.

[03:36] Lembra como passamos valores, variáveis para a “view” utilizando o “model”, aquela interface do Spring? Ou seja, nós não precisamos colocar como atributo do “HttpServeltRequest”. Nós colocamos como “addAttribute(“pedidos”, pedidos”)”.

[03:58] Salvou, então pronto! Isso já faz com que estejamos mandando informações para a “view”. Só que não estamos acessando essa informação. Então se eu abrir “localhost:8080/home”, vai continuar como estava com aquelas informações fake que colocamos no HTML. Só que agora eu vou substituir usando o “th:text”.

[04:19] Como eu vou acessar um determinado pedido de uma lista de pedidos? Eu preciso percorrer essa lista porque eu não tenho como fazer “pedidos” do

tipo 0. Não faz sentido eu percorrer desse jeito porque eu não sei quantos pedidos vão vir dentro dessa lista.

[04:36] Então eu vou colocar “th:each”, que é do “Thymeleaf”, “pedido:”. Como eu acesso os pedidos? Utilizando esse símbolo. Então “pedido.nomeProduto”. Só que ele não vai acessar isso diretamente, ele vai achar que isso é só um texto que ele tem que apresentar.

[05:03] Para ele extrair o valor da variável, nós temos que colocar aqui dessa forma. A mesma coisa vamos fazer para o “valor negociado”. Então “th:text” nesse “span” e colocar de novo o Expression Language, “pedido.valorNegociado”.

[05:21] Será que já mudou alguma coisa? Se quebrou ou se está funcionando? Beleza, está funcionando! O valor é que não setamos mesmo e nem data. Então só vamos ver alguma coisa aqui quando passarmos o “Produto”, a “URLProduto” e a “Descrição”. “Data da entrega”, “th:text”, de novo, “pedido.dataDaEntrega”.

[05:45] Só para confirmarmos aqui, é só entrarmos no “Pedido” e o nome é “Data da entrega” mesmo. Beleza! A “UrlProduto”, vamos só conferir uma coisa porque eu estou colocando a URL direto dentro da “div”. Mas a URL do produto fica dentro de um “input”. Você deve ter visto isso. Então, “

<input th:value=“url do produto”
”.

[06:10] Vamos ver como ele muda. Vou apertar a “F5” aqui. Pronto! Agora vou dar um “input” mesmo. Só que nós temos que substituir esse “value” e usamos o “th:value” no caso, já que estamos querendo substituir esse aqui, o “value” do “input”.

[06:25] Então colocamos “pedido.urlProduto”. E no “textarea”, “th:text” mesmo, que é o texto que fica dentro do elemento, “pedido.Descricao” e na URL,

“th:src” também tem e colocamos “pedido.urlImagem”.

[06:52] Pronto! Já substituímos todos os valores. “Data da entrega”, “Produto”, “URLProduto”, a “Descrição” e a “URLImagem”. Como fica agora? Veja que na imagem ele já está apresentando a “URLProduto”, uma “Descrição” qualquer para esse pedido, que colocamos no “Controller” na hora que ele preencheu.

[07:11] A data e o valor não existem, mas o produto já existe. Então isso já está sendo dinamicamente. Ele está percorrendo lista para gerar essa tela. É legal porque se nós entrarmos no “HomeController” e nessa lista, eu só pegar esse pedido e adicionar algumas vezes, ele vai percorrer. Nós só temos um pedido. Se você atualizar, vai ter esse mesmo pedido replicado.

[07:38] Então já está realmente feito dinamicamente. Deixe-me voltar para o que estava mesmo. Então o que precisamos fazer agora é aplicar o Bootstrap para deixarmos o layout de acordo com esse layout que recebemos. Até o próximo vídeo!