03

Rotas do status

Transcrição

[00:00] O que nós temos é essa página principal listando todos os pedidos, mas sem a capacidade de listar por “status”, que é esse “Aguardando”, “Aprovado” e “Entregue”. Então esses links ainda não estão funcionando e nós vamos fazer isso agora. Quando fazemos uma requisição para “/home”, ele bate nesse “homeController” e nesse método que está mapeado com “GetMapping(“/home”)”.

[00:23] O que nós vamos fazer agora? Vamos criar mais um método que em vez de ser para “/home”, vai ser para “/home/aguardando” que é o primeiro “status” - que é o “status” de quando o pedido é criado. Por exemplo: esse pedido aqui.

[00:39] Então vamos mudar o nome do método para “Aguardando” e vamos fazer uma melhoria aqui. Esses dois métodos - o primeiro, que é “/home” que lista todos, já está mapeado para esse “/home”. O novo que criamos está “/home/aguardando” e os outros também vão ter esse mesmo prefixo “/home”.

[00:59] O que eu vou fazer? Eu vou pegar esse prefixo e vou associar ele ao “@Controller” usando “@RequestMapping”, uma outra anotação. Então todas as requisições “/home” vão bater aqui. Então na anotação, só “/home” vai chegar nesse método. Uma anotação “/home/aguardando” vai bater nesse método aqui de baixo. Só que esse método “aguardando” ainda está apresentando todos os pedidos. Então se olharmos, “/home/aguardando” já funciona.

[01:32] Só que ele continua listando todo mundo, não só esse primeiro pedido que é o único que está com o “status” aguardando. Como resolvemos isso? Nós

criamos um “findByStatus” e passamos o “status” do pedido. Só que esse “status” não existe.

[01:53] Nós precisamos receber o “status” do usuário. Como fazemos para receber o “status” do usuário? Vamos fazer o seguinte: vamos colocar “StatusPedido.AGUARDANDO” e criar a declaração desse método da interface “Repository”.

[02:09] Então vou apertar as teclass “Ctrl + 1”, “Create method 'findByStatus'“ e ele já foi criado no nosso repositório, que é uma interface. De novo estamos utilizando, se você olhar a anotação... Esse “JpaRepository”, estamos vendo que é o Spring Data JPA. A implementação do Spring Data JPA olha para os nomes dos métodos que nós estamos declarando na interface. Então se declaramos um método que retorna uma lista de pedidos, ele sabe que vai fazer lá um “select *” na tabela “pedido”.

[02:43] E se o nome do método é “findByStatus” ele sabe que tem que adicionar um filtro, um “[WHERE]” onde “status” é igual ao “status” que estamos passando aqui. Não vou colocar “aguardando”, vou colocar “status”. Então o próprio Spring Data JPA vai conseguir implementar esse “select” para nós. Essa é uma das grandes vantagens de utilizar o Spring Data, porque ele simplifica bastante essa consulta.

[03:07] Então se fizermos uma requisição de novo para “/aguardando”, agora só aparece esse “Xiaomi Redmi Note 10” aqui porque agora o Spring Data já implementou esse método e ele está me retornando só os pedidos com “status” e “aguardando”. Só que existe uma maneira melhor de implementarmos esse método. Olhe o que acontece. Se eu utilizar o método desse jeito sempre, eu vou ter que criar mais um método, por exemplo, para “aprovado” e aí colocar aqui “aprovado”. E o “status” do pedido vai ser “aprovado”.

[03:43] Ou seja, se eu fizer uma requisição agora, em vez de “aguardando”, mas “aprovado”, ele vai trazer o Kindle porque é o único que está com “status” aprovado. Só que eu estou me repetindo muito e a única coisa que está sendo

alterada, na verdade, é o valor dessa requisição. Porque a implementação do método é praticamente a mesma.

[04:04] Uma outra forma mais simples de fazer isso é nós recebermos esse valor que veio no "path" da URL como uma variável. Darmos o nome de "status". Aí pedimos para o Spring nos retornar qual o valor que foi preenchido pelo usuário. Então, por exemplo: esse "status" que estamos esperando na URL, é esse valor que estamos passando aqui. No caso, "aprovado".

[04:34] Se eu fizer uma requisição para "/home/aprovado", o valor desse aqui é "/aprovado" e eu vou pedir para o Spring injetar para mim o "status". "Injete esse valor de 'status' aqui para mim". E aí eu vou indicar que esse parâmetro "status" tem que vir de uma variável que vem do "path" chamada de "status".

[05:01] Então o Spring, pega uma variável agora que eu declarei no "path" da URL e me dá dentro dessa variável "status". E agora, o que eu vou fazer? Eu vou passar agora esse "findByStatus". Em vez de passar o "StatusPedido.AGUARDANDO", eu vou colocar o "status" aqui.

[05:23] Isso não vai funcionar porque eu estou passando agora uma String e não aquele valor do "enum" do "StatusPedido". Só que conseguimos converter a String para um "enum". Porque o valor da String é o mesmo valor do "enum" e aí ele consegue converter String para "enum".

[05:42] O próprio "enum" tem um método chamado de "valueOf" que recebe uma String e retorna um "StatusPedido". Então eu vou passar "Status.valueOf(status)" e ele vai transformar essa String em um "StatusPedido" e tudo vai funcionar. Será? Vamos lá! Vamos fazer uma requisição de novo. Vamos ver o que acontece.

[06:12] Não conseguiu resolver "Status" nesse aqui. Então tem alguma coisa errada. Vamos tirar esse aqui e ver se é isso. Continua sem conseguir subir. Isso não é Thymeleaf. No caso de uma variável de "path" é só abrir e fechar chaves. Agora vai funcionar.

[06:38] Funcionou, mas deu erro. Deu erro por quê? Porque esse “aprovado” é o mesmo valor do “enum” que existe. Se olharmos dentro do “StatusPedido”, existe o valor do “enum” de “aprovado”. Só que aqui ele é caixa alto.

[06:50] Quando ele vai converter String para “enum”, a String tem que ser exatamente o mesmo valor do “enum”. Tem que ser em caixa alta. Então ele está reclamando exatamente isso. Eu não consegui converter esse valor para um “StatusPedido”. “StatusPedido.aprovado” não existe, “.aprovado” em minúsculo.

[07:09] E existe “StatusPedido.APROVADO”, em maiúsculo. Pronto! Se eu fizer uma requisição agora, vai funcionar. Então se eu colocar, por exemplo: “aprovado”, “entregue” que são “status” que existem, ele vai nos colocar para nós conseguirmos fazer uma busca.

[07:28] Eu só vou mudar o nome desse “aprovado” para “porStatus”. Eu vou colocar o nome desse método “porStatus”. Só que ainda temos um problema aqui, que é o seguinte: se o usuário vier aqui e colocar um valor que não existe, ele vai tomar um erro e nós estamos apresentando esse erro esquisito para o usuário.

[07:48] O que podemos fazer aqui é um método que, quando tiver acontecido algum “error”, nós fazemos aquele “redirect” que já vimos antes. “Redirect:/home”. Então se algum erro acontecer no “List<Pedido”, nós jogamos lá para “/home”. Só que temos que indicar qual é o erro.

[08:18] Existe uma anotação chamada de “@ExceptionHandler”, onde nós passamos qual é a exceção que queremos mapear. É a “IllegalArgumentException”. Vamos ver o que acontece agora. Pronto! Agora ele redireciona para “/home”.

[08:37] Então se eu fizer “/home/aguardando”, ele mostra todos os pedidos aguardando. Se for “aprovado”, ele mostra o pedido “aprovado”. E se for, por exemplo, “rejeitado” - que é “status” que não mapeamos - ele volta para

"/home" porque esse "status" não existe. Ele não encontra e ele lança um "IllegalArgumentException". Ou seja, essa conversão lança um "IllegalArgumentException" porque agora nós estamos fazendo um tratamento para essa exceção. Beleza? Bem simples!

[09:07] E agora já estamos conseguindo fazer as requisições de acordo com o "status". Só tem um problema: esses menus ainda não estão funcionando. Então vamos fazer esses menus funcionarem! Eu vou entrar na página "home" e procurar o menu. Está aqui o menu, "Todos", "Aguardando" e "Entregue".

[09:29] Eles já têm o "href", mas que não está sendo utilizado. Então nesse caso, no "Todos", vamos colocar "/home". Nesse de baixo, "/home/aguardando". Nesse aqui, "/aguardando/aprovado" e nesse "/home/entregue". Está certo? Tudo certo.

[10:00] Então vou fazer uma requisição. Vamos clicar em "Entregue". "Aprovado" está funcionando também e "Aguardando" também. Só tem mais uma questão que nós não estávamos resolvendo aqui. que é a seguinte: o único item desse menu que está ativo é o "Todos".

[10:15] Por exemplo: se eu clico em "Todos", ele tem que ficar escuro mesmo, mas se eu clico em "Entregue", quem tem que ficar escuro é o "Entregue". Essa classe "active" só pode ser adicionada em "Todos", quando eu estiver feito uma requisição para "/Todos".

[10:34] Só tem que aparecer o "active" para "Aguardando" se eu estiver fazendo uma requisição para "Aguardando". Então a primeira coisa que eu preciso fazer para criar essa funcionalidade, para adicionar uma classe quando eu estiver clicado em um item de menu, é adicionar um atributo que indica qual é o "status" que eu estou apresentando dos pedidos.

[10:56] Então em "homeController", eu vou adicionar no "Model" mais um atributo chamado de "status", onde eu vou passar esse valor do "status" que o usuário fez a requisição. Então agora tem um atributo novo chamado de

"status" em que eu posso acessar o valor, por exemplo, "Aprovado", "Entregue" e tal.

[11:17] Por que eu fiz isso? Porque em "home", eu vou fazer o seguinte: quando o "status" for "aguardando", eu adiciono uma classe chamada de "active". Então, como eu faço isso? "th:classapprend" e aí vou fazer o seguinte: se "status" for igual a "aguardando" eu adiciono a classe "active". Então eu uso esse "classappend". Então vamos clicar em "Aguardando". Pronto!

[12:07] "Aguardando" já está selecionado e realmente funcionou. Quando eu clico em "Todos", ele já não tem mais, já não está mais com essa classe e a seleção desse "active" só está funcionando com "Aguardando".

[12:22] Eu vou fazer ele funcionar para os outros também. Então vou copiar esse "classappend" para cá. Só que esse link é o "Aprovado". Então eu coloco "Aprovado" e aqui embaixo é o "Entregue". Vou colocar "Entregue". No caso do "Todos", é quando o "status" for o quê? Quando o "status" for igual a nulo. Quando não tiver sido preenchido.

[12:48] Vamos ver se funciona. Então vamos clicar em "Todos". Beleza! "Todos" está funcionando. "Aguardando" funcionando, "Aprovado" e "Entregue" também. Agora o menu já está funcionando. Se bem que não é difícil, é bem simples. Nós só tivemos que adicionar esse "classappend" que adiciona a classe "active" aqui dentro desse atributo do link.