04

Contexto e métodos nas views

Transcrição

[00:00] Agora que já fizemos a lógica de administração com autenticação e as views que listam o usuário e produto, já temos nosso sistema pronto. Essa última aula vai ser para melhorar um pouquinho a habilidade dele, vendo uma funcionalidade das views que ainda não vimos antes, que é criar uma função dentro de uma view, é semelhante a criar uma tag mas o escopo é local.

[00:24] Para começar, aquela lista de usuários de produto elas estão com alguns códigos feios então vamos dar uma olhada nelas, eu vou abrir aqui a nossa lista de usuários e vamos ver que tem aqui esse código horrível, que é um código meio confuso, podíamos simplificar ele um pouco. Vamos simplificar esse porque estamos usando ele aqui dentro, como abrimos uma tag? Como criamos uma função?

[00:55] Vamos vir aqui em cima ou em qualquer lugar da sua view, mas eu gosto de declarar tudo em cima e eu vou fazer um @numeroDeAcessos, isso cria uma função chamada numeroDeAcessos, o que eu vou passar? Eu vou passar um usuário para dentro dessa função, então usuario: Usuario, então agora eu tenho uma função que recebe um usuário.

[01:20] Eu vou declarar que essa função é uma função em scala, ela vai retornar código em scala, no caso ela vai retornar um número, o que eu vou fazer aqui? Eu vou usar a variável usuario.getAcessos().size(), a última linha de uma função escala dentro da view representa o retorno dela, como essa é a única linha, ela vai ser o retorno que é o tamanho da lista de acesso de um usuário, então agora eu já posso substituir aqui em baixo, em vez de @ usuario.getAcessos().size() eu vou substituir por @numeroDeAcessos.

[02:08] Passando o usuário como parâmetro. Agora vamos melhorar um pouco essa lógica aqui, vamos criar um método que pega o último acesso de um usuário, então vamos lá, @ultimoAcesso e vai receber também um usuário, usuario: Usuario, e declaramos aqui que isso é uma função scala, é para isso que serve esse @.

[02:40] Vamos lá, vamos fazer o que? Primeiro vamos pegar o último acesso desse usuário, então eu vou declarar aqui uma variável último, que vai ser o número de acessos, estamos usando a função acima, passando o usuário, só que o último acesso é o número de acessos -1. Já temos nosso último acesso, agora se o último acesso for maior ou igual a zero, se o usuário tiver tido algum acesso retornamos a data desse último acesso.

[03:26] Se o ultimo for maior ou igual a zero, vamos retornar o último acesso do usuário, usuario.getAcessos().get(ultimo).getData(). É só isso, agora substituímos lá embaixo e fica um código um pouco mais claro, vamos lá, usuário, removemos tudo isso aqui e vamos pegar ultimoAcesso do nosso usuário atual.

[04:05] Vamos fazer o teste, vamos dar uma olhada como fica isso aqui lá, se atualizarmos não muda nada, não era pra mudar, porque estamos exatamente com os mesmos dados no banco, agora se eu for lá e apagar todos acessos do meu usuário, então vou fazer delete from registro_de_acesso, agora meu usuário não tem mais acesso nenhum, vamos ver o que acontece aqui, eu vou atualizar, ele tem 0 acessos e o último acesso não existe.

[04:43] O código ficou mais legível, mais bonitinho e ainda aprendemos uma nova função, vamos usar a mesma coisa na tela de produtos para fazer a formatação daquele preço que estava meio zoado, produtos tem esse preço que tem infinitas casas decimais, vamos fazer uma função para fazer a formatação desse preço.

[05:07] Eu vou abrir aqui uma função chamada formataEmReais e vamos fazer uma lógica baseada em um preço, o preço é um double, então fazemos a função receber um double, abre e fecha a função, o que vamos fazer? Já temos o

número do preço, só precisamos formatar ele, então vamos usar String.fomart Do Java mesmo, String.format recebemos aqui uma string e um parâmetro, o parâmetro vai ser o preço.

[05:45] Essa string vai ser o que? Vai ser R$ e colocamos o preço aqui %f e já formatamos o preço com R$ coladinho no número, só que ainda não resolvemos o problema das casas decimais, como fazemos isso em uma string? É só digitar .2 você tá indicando que tem 2 caracteres depois da casa decimal, agora é só usarmos essa função lá em baixo no produto.getPreco.

[06:20] Então vamos remover isso aqui, @formataEmReais(produto.getPreco()), vamos ver como que fica, vou dar uma atualizada aqui e vamos ver que o nosso preço fica bem mais da hora, ele ficou arredondando para cima porque estava muito próximo do 13 e na verdade esse String.format é de scala e não de Java e ele tem essa função de arredondamento.

[06:50] Então é isso para essas duas views, ainda temos uma melhoria que podemos fazer que é alterar o cabeçalho, fazer ele se adequar caso você seja um admin e caso você seja um usuário, então vamos dar uma olhada nisso. Vamos abrir a página onde está o cabeçalho que é a página main.scala.html, precisamos dar um jeito de recuperar o usuário que está logado aqui na main.

[07:20] E para isso podemos passar o usuário em todas as views, só que isso seria um problema, porque você tem que adicionar em cada uma das views, em cada vez que você renderiza uma view você tem que passar o usuário, e nem sempre o usuário está logado, então isso pode ser um problema. Temos uma alternativa para isso, temos um mapa de argumentos dentro do contexto atual, lembra daquele contexto que vimos durante a autenticação?

[07:45] UsuarioAutenticado.java temos aqui o contexto, esse contexto tem uma lista de argumentos, podemos inserir o nosso usuário dentro desse contexto, então é isso que vamos fazer, vamos adicionar o usuário no contexto, mas para isso eu vou extrair a variável do usuário para uma variável local, então vamos lá, context.args.put, vou chamar essa variável aqui de usuario para pegarmos na view e passar o usuario pare dentro dela.

[08:18] Para usuário admin, temos que fazer a mesma coisa, então vamos lá no admin autenticado e fazer a mesma coisa, caso o usuário seja um admin colocamos ele dentro do contexto. Então agora que já temos o usuário dentro do contexto, vamos vir aqui e recuperar ele, como fazemos isso? Vamos declarar um método que faz essa lógica para gente, porque é uma lógica que não vamos querer ficar recuperando sempre, você vai descobrir o porquê agora.

[08:47] Eu vou declarar o método usuário que vai ser igual esse código bizarro aqui, temos que chamar o Http.Context.current, aqui temos nosso contexto, então podemos acessar aquela variável args, então .args.get e vamos pegar a variável de usuário, só que o contexto não sabe que isso é um usuário ele acha que isso é um objeto, então temos que dizer que isso é um usuário .asInstanceOf e passamos aqui a classe de usuário, agora temos nosso usuário sendo acessado por esse comando aqui e não essa coisa enorme, que não gostariamos de repetir.

[09:37] Então vamos lá, vou mudar a nossa lógica, ao em vez de perguntar para request se ele tem um username inválido, podemos perguntar se o método de usuário está retornando um objeto nulo ou não, caso ele não esteja, criamos a rota de painel e log out, isso para usuário comum, para clientes nossos está ok, mas e para os admins?

[10:00] Vamos deixar o painel em primeiro e o log out em último, agora perguntamos aqui @if(usuario().isAdmin()), podemos mostrar as rotas das listas de produto e usuário, então vou copiar aqui e alterar essas rotas para nós, AdminController.usuarios, aqui é a lista de usuários e mesma coisa para lista de produtos, vou alterar aqui para produtos e mostrar aqui a lista de produtos.

[10:37] Então é isso, vamos fazer o teste? Eu vou vir aqui no meu painel, clicar em painel e olha só, novo produto painel usuários produtos e log out, quer dizer que eu esqueci de apagar o link de novo produto, vamos apagar ele então, porque ele não é mais necessário, então vamos lá, agora sim F5, temos o painel, a lista de usuários porque o meu usuário é admin e a lista de produtos, porque eu posso criar um produto aqui dentro.

[10:07] Vamos alterar o meu usuário para ele não ser mais admin, update usuario set admin = false, então agora se eu der um F5 aqui, esses dois links tem que sumir, vamos ver? Sumiram é isso, agora temos um menu dinâmico para o nosso sistema, nosso código está bonitinho e por fim acabamos nosso projeto. No último vídeo eu vou falar rapidamente sobre o que vimos no curso inteiro, obrigado por assistir e até a próxima.