01

Listagens e autenticação

Transcrição

[00:00] Vimos na aula passada como criar ações, actions, que são lógicas que interceptam as requisições e executam um código antes do código do controller ser executado, mas caso a requisição seja autenticada, após a autenticação acontecer. Agora vamos alterar um pouco o modelo de usuário, para podermos ter administradores e clientes separadamente, mas antes vamos criar algumas telas de gerenciamento para os administradores não autenticadas.

[00:32] Por hora vamos só visualizar uma lista dos usuários e dos produtos, eu vou criar uma view que é uma lista de usuários, vou criar aqui um novo arquivo chamado usuarios.sacala.html e vou usar uma colinha aqui, porque a parte de designer da tela não importa tanto, o que importa é como vamos fazer a lógica de recuperar os dados.

[01:00] Então vamos ver, recebemos aqui uma lista de usuários, tem um título e tem uma tabelinha com ID, nome, email token e data do último acesso. Vamos preencher essa tabela, precisamos percorrer os usuários então vamos usar um laço um for, e vamos pegar nesse for um usuário dentro da lista de usuários, essa é a sintaxe que usamos para o for em escala.

[01:30] Então agora já temos nosso usuário, precisamos abrir uma linha para inserir os dados na tabela e eu vou copiar aqui essas entradas para irmos substituindo enquanto preenche os dados, deixa eu tirar a indentação desse tr aqui e vamos em frente. @usuario.getID, vamos fazer isso para todos que precisam aqui @usuario.getNome @usuario.getEmail @usuario.getToken, só que no caso vamos pegar o .getCodigo também

[02:17] Acessos quase a mesma coisa, vamos pegar os acessos e mostrar o tamanho deles, então acessos.size, um acesso é um pouquinho mais complicado, precisamos pegar o último acesso e pegar a data dele. Então acessos, temos a lista de acessos, vamos pegar o último acesso, o último acesso é o acesso que tem o tamanho da lista de acessos menos um, e pegamos a data dele.

[02:50] Nossa view de usuário está pronta, vamos fazer a de produtos, eu vou criar um novo arquivo produtos.scala.html vai ser quase a mesma coisa, como na tela de usuários eu vou pegar a minha colinha, para não termos que fazer a estilização e sim se preocupar com a lógica, temos aqui uma lista de produtos e de cada produto temos, ID, título, código, tipo, descrição e o preço.

[03:26] Vamos lá fazer a mesma coisa, @for(produto <- produtos), vamos adicionar uma linha com as informações do produto e fechamos esse laço aqui, agora eu vou copiar essas entradas substituindo aqui na nossa tabela, então @produto.getId, vamos fazer a mesma coisa com o título, código, tipo, descrição e preço, o preço vai ser um pouco diferente porque temos uma formatação, então aqui é tipo, temos a descrição.

[04:18] E o preço precisamos colocar a unidade monetária, então aqui colocamos um R$ e temos nosso preço. Agora precisamos fazer as rotas e as lógicas nos controllers, vamos só ver se compilou tudo bonitinho, está tudo certo, então não tem nenhum erro de lógica aqui nas views, então vamos partir paras rotas, eu vou copiar isso aqui e abrimos uma nova seção para os admins, eu vou fazer aqui /admin/usuarios e /admin/produtos para listarmos esses dois modelos.

[05:00] Então aqui em vez de ProdutoController é AdminController a mesma coisa aqui em baixo AdminController e vamos fazer aqui uma lista de usuários e uma lista de produtos, temos as nossas rotas, agora vamos partir para o controller, eu vou criar aqui uma nova classe, que é o AdminController, e aqui estendemos controle do play mvc, vou importar aqui * e já importar as views html que vamos usar.

[05:45] Vamos lá então, public Result, temos um método usuários que vai retornar uma lista de usuários, então return ok(usuarios.render()) e ele vai receber uma lista de usuários, ainda não temos essa lista, então por enquanto deixa assim, e a mesma coisa aqui, public Result produtos() que vai receber uma lista de produtos, return ok(produtos.render()) e passamos para ele uma lista de produtos.

[06:30] Como que pegamos a lista de produtos que já temos? Era lá do produtoDAO, então vamos injetar o produtoDAO e pegamos a lista para retornar para ele, então produtoDAO.todos, guarda isso em uma variável produtos e joga aqui para o render produtos, a mesma coisa vamos fazer para os usuários, só que ainda não temos o método todos no usuarioDAO, inject private UsuarioDAO usuarioDAO;

[07:15] Vamos vir aqui UsuarioDAO.todos e listamos todos usuários, vamos falar que isso é uma lista de usuários chamada usuários, agora podemos criar esse método e já fazer o retorno dele, então create método todos, retornamos aqui, controllers.Usuario não, classe errada aqui, import model. Então vamos lá, vamos vir aqui usuarios.all e retornamos todos usuários e aqui deve resolver os nossos erros de compilação.

[07:57] Eu cometi um erro aqui, porque a nossa view chama usuários e a nossa lista chama usuários também, então eu vou mudar para listaDeUsuarios e a mesma coisa aqui, listaDeUsuarios e a mesma coisa aqui no produtos vai ter que chamar listaDeProdutos, eu vou copiar isso aqui e jogar aqui no render e agora sim ele consegue renderizar tudo do jeito correto.

[08:25] Então agora podemos dar uma olhada em como que isso fica aqui admim/usuarios, vamos ver o que aparece para gente, olha só, uma lista de usuário com todos campos que fizemos e os dados de um usuário porque é o único usuário que tem no banco, no caso de produtos vamos conseguir ver aqueles 3 livros que cadastramos com preços adequados.

[08:55] O que eu adicionei aqui foi um botão de criar novos produtos, para facilitar quando estamos na lista, criamos um novo produto. Então, vamos criar mais um, um livro de sql, então livro-sql, vai ser do tipo livro, com a descrição livro de sql e o preço vai ser R$12,99 e vou colocar alguns 9 a mais para vermos um probleminha que temos aqui.

[09:23] Olha só, vamos lá na lista de produtos, admin/produtos e aqui está sem formatação, vamos resolver isso um pouco mais para frente, por hora o nosso foco é autenticar o admin, agora que temos as telas é só alterar o modelo de usuário para virar um admin, criar um sql e criar a autenticação, então vamos começar pelo usuário, criamos aqui uma chave admin eu vou criar uma boolean mesmo mas se você quiser fazer diversos papéis você gera um enumerador, private boolean admin com getter e setter.

[10:08] E eu vou pegar a cola do meu sql para fazer evolução, então vamos lá, eu vou abrir um novo arquivo aqui set.sql, vamos só copiar e colar aqui mesmo, ele já gera o número 7, eu vou vir aqui e pegar minha colinha da evolução, que é basicamente alter table, que é adicionar coluna, então alter table usuario, adicionar coluna admin e dropar coluna admin, caso precise desfazer a ação.

[10:39] Temos o nosso usuário com admin, agora vamos vir aqui atualizar o banco e vamos fazer a autenticação de um administrador, como fazemos isso? Vamos criar um usuário autenticado, no caso um admin autenticado, eu vou dar um “Ctrl + N” aqui, criar uma classe e chamar admin autenticado que vai estender Authenticator do play.mvc.Security.

[11:13] Vamos fazer quase a mesma coisa aqui, então eu vou copiar essa lógica aqui de dentro, eu vou copiar os dois métodos que subscrevemos, só que aqui vai ter uma coisa diferente, após descobrir se o usuário está presente, se ele existe, temos que conferir se ele é um admin, então possivelUsuario.get, guarda ele em uma variável usuario.isAdmin, se ele for admin retornamos o nome do usuário, se não ele vai cair aqui em baixo até retornar nulo.

[11:55] Agora aqui, eu não vou redirecionar ele para página de login, e podemos dizer que ele não está autorizado, mas eu vou redirecionar ele aqui para o painel de usuário e se o cara não estiver sequer logado, o painel do usuário tem

autenticação de usuário, ele vai redirecionar para o admim, eu vou apagar essa notificação porque ela não importa.

[12:20] E quais são as rotas que precisamos autenticar como admin? Claramente o admin controller precisa ser autenticado como admin, @Authenticated(AdminAutenticado.class) e a outra rota que precisa ser autenticada direitinho, é o produto controller, porque são as lógicas de criar um produto e só um admin pode criar um produto.

[12:45] Então vamos lá, produto controller vamos adicionar aqui também a autenticação com admin, quais são os métodos que temos aqui? Salvar o novo produto e o formulário de novos produtos, são todos os métodos que precisam de autenticação de administrador, então está tudo ok.

[13:05] Podemos tentar acessar aqui novo produto e não vamos conseguir. O usuário não tinha na variável admin dele, então ele não tem false, ele não tem true, ele não tem nada e por isso o banco quebrou, podemos fazer uma alteração aqui update usuário set admin = false, ele deve resolver esse problema.

[13:35] Não sou um admin, eu vou tentar acessar o painel de admin de produtos, somos redirecionados para o nosso painel, agora se viermos aqui e alterar para true, não faça um update sem where se você não souber o que se tem no banco, claro. Agora vamos vir aqui, meu usuário é admin, então vamos lá acessar admin produtos, agora eu posso criar um novo produto e eu posso ver as listagens dos nossos produtos.

[14:10] O que fizemos nesse vídeo? Fizemos toda lógica de autenticação e as telas de admin que são uteis para gerenciarmos nossos usuários e produtos, o próximo vídeo vai focar em deixar um código um pouco mais legível, como aquela parte do usuário que tinha que fazer um get com a lista de acessos e também vai formatar a tabela de produtos aqui, com esse preço que está estourando com os caracteres, vamos melhorar um pouco essa legibilidade do código e experiência do usuário.

[14:38] E também vamos fazer uma coisa muito interessante, que é tornar o cabeçalho dinâmico, de acordo com as credencias do usuário, caso ele seja um admin, adicionamos aqui as telas de listar produtos e usuários, para fazer isso vamos criar métodos em scala nas views, coisa que ainda não tínhamos visto, vimos como criar views separadas mas não como criar métodos dentro delas.