06

Verificação e liberação

Transcrição

[00:00] Na última aula aprendemos como criar um template de email simples utilizando a extensão da classe email, aqui com remetente, assunto, destinatário e o corpo do Html, criados direto no construtor e fizemos isso para enviar aquele token de confirmação para o nosso usuário, por exemplo se eu criar aqui um usuário marco, com email marco@caelum.com.br, recebemos aqui no nosso log um link para fazer a confirmação do nosso usuário recém criado.

[00:37] Eu vou copiar esse link aqui porque vamos usar ele na aula de hoje, que vamos fazer a efetiva confirmação do cadastro de usuário, já temos o link, precisamos agora criar a rota desse link, então, vamos lá no arquivo de rotas. Abrimos aqui route e vamos fazer um novo get, porque get? Porque o usuário vai clicar em um link e não enviar um formulário.

[01:05] Criamos um get com a rota/usuário/confirma/:email:codigo. Isso vai estar atrelado ao usuário controller, então controllers.UsuarioController e eu vou criar um método chamado confirmaUsuario, que vai receber como parâmetro o email e o código.

[01:50] Agora só precisamos ir no próprio controller e criar esse método, vamos lá? Abrir aqui o nosso controller e vou criar o método public result confirma usuário que recebe um email e um código. E o que fazemos dentro desse método? A primeira coisa é tentar pegar o usuário a partir do email e o token a partir desse código que temos, deixa eu só dar um return ok para ele não ficar com erro de compilação.

[02:30] Então vamos pegar o usuário a partir do email, para isso precisamos do usuarioDAO, então vou injetar aqui usuário down, @inject private UsuarioDAO usuarioDAO, importar aqui a classe do pacote DAOs. Vamos lá de volta, então pegamos o usuário a partir do usuarioDAO a partir do email que recebemos e guardamos numa variável que é um possivelUsuario, a mesma coisa vamos fazer com o token, vamos receber uma variável do tipo optional, com token de cadastro que chama possível token.

[03:25] Que vai receber do tokenDeCadastroDAO do método com código e passamos o código aqui para ele, o token de cadastro DAO ainda não existe, mas vamos fazer ele daqui a pouco depois que criarmos a lógica aqui no nosso controle.

[03:44] Depois que temos o possível usuário e o possível token, vamos fazer aqui uma tripla verificação, que é conferir se o usuário do email existe, se o token que pegamos com esse código existe e caso os dois existam ainda conferimos se o usuário que tem dentro desse token, é o mesmo usuário que pegamos com email. Para garantir que a nossa url não foi adulterada, garantido assim uma melhor segurança do nosso aplicativo.

[04:12] Então vamos lá, if (possivelToken.isPresent() && possivelUsuario.IsPresent()), já garantimos que os dois existem, só precisamos confirmar agora que os dois são iguais, então vamos pegar agora o token e o usuário, possivelToken.get, guardamos isso em uma variável chamada token, mesma coisa possivelUsuario.get, guarda isso em uma variável usuário.

[04:52] E se token.getUsuario for igual ao usuário que recebemos aqui, é um sucesso e podemos liberar nosso usuário, como liberamos um usuário? Primeiro apagamos o token, depois vamos marcar o usuário como verificado, então setVerificado usando aquela booleana que criamos antes, então true. E enfim, atualizamos o modelo do usuário, usuario.update, podemos vir aqui e já aproveitar e mandar uma mensagem de sucesso para o usuário, sucess, falando que o cadastro dele foi confirmado com sucesso.

[05:40] Então “Cadastro confirmado com sucesso”, outra coisa que podemos fazer aqui é marcar que o usuário vai ser logado imediatamente aqui, podemos deixar ele redirecionar para página de login ou podemos simplesmente já fazer o login por ele e deixar ele navegar no nosso sistema, afinal nós temos uma tripla verificação, então é bem provável que seja o usuário que se cadastrou aqui no nosso sistema.

[06:13] Então eu vou fazer o login, outra coisa que podemos fazer aqui é já redirecionar ele para um painel de usuário, como ele ainda não existe, então vou fazer a mesma coisa, usuário/painel e marcar aqui como TODO para criar essa rota mais tarde.

[06:35] Agora caso todas essas verificações falhem, precisamos mandar uma mensagem de erro, então flash(“danger”, “Ocorreu um erro ao tentar confirmar o cadastro!”). E redirecionamos isso para tela de cadastro ou de login, bom eu vou redirecionar para a tela de login porque ela é um pouco mais genérica, é meio que o portal de entrada do nosso aplicativo.

[07:12] E como uma rota ainda não existe eu vou deixar aqui um TODO rota. Agora só precisamos do token de cadastro DAO, vamos pegar ele aqui e criar o objeto e depois criamos a classe a partir dos atalhos do eclipse, então “private TokenDeCadastro DAO”, é só mudar para maiúscula e aqui o nome da variável, então vamos vir aqui, criamos a classe no pacote DAOs que não estende nada, não precisa.

[07:53] E agora podemos criar o método que estávamos usando ali, aqui com códigos, então criamos o método com código, ele já sabe que tem que retornar um optional de um token de cadastro e que ele recebe uma string chamada código, precisamos aqui para poder retornar o nosso token o finder, então vamos fazer o finder.

[08:12] private Finder<Long, TokenDeCadastro> tokens = new Finder<>(TokenDeCadastro.class). Importamos aqui o finder do pacote model e podemos só fazer a lógica aqui finalmente, tokens.whgere().eq, então o atributo código tem que ser igual o código que recebemos aqui e precisamos achar um único token que tenha essas condições.

[09:03] Guarda isso aqui em uma variável token e retorna um objeto opcional dela, Optional.ofNullable(tokenDeCadastro). Então agora podemos até testar e ver como está funcionando, vamos pegar aquele email de novo e vamos ver se está tudo certo. Eu vou acessar aqui a minha url e vemos no que dá, olha só caímos no painel, então quer dizer que a princípio funcionou, confirmamos nosso usuário.

[09:40] Vamos ver aqui no mysql se está tudo certo, o token não existe e parece que dá verificado aqui, quando está ele não faz essa quebrinha estranha, porque tem um carácter invalido aqui no meu PC. Então é isso agora temos um usuário, com meu email e minha senha, está verificado, acreditem em mim e o token dele foi removido, foi apagado.

[10:05] O que fizemos hoje? Fizemos a lógica de quando o cliente acessa o link que ele recebeu por email, confirmando com uma verificação tripla a autenticidade dele, e apagamos o token do banco e marcamos o usuário como verificado, só que o usuário ainda não consegue fazer login no sistema que é a funcionalidade principal que queremos aqui. Na próxima aula vamos ver como fazer a autenticação do usuário, já utilizando a segurança embutida no Play Framework.