04

Envio e template de email

Transcrição

[00:00] No vídeo anterior geramos um token criptografado para o nossos usuário a partir dos dados dele e mais um sal que é uma string aleatória e configuramos nosso enviador de e-mails, mais ainda estamos enviando um email vazio que não tem sequer destinatário. Então precisamos completar o conteúdo desse email, poderíamos fazer aqui criando um novo objeto email em uma variável e fazendo email.setRemetente email.setDestinatario, mas isso não é legal porque todos e-mails de cadastro vão ser iguais, então podíamos fazer um template.

[00:39] E um jeito bom de fazer um template é estendendo a classe email, então vamos fazer isso, eu vou estender a classe email chamando ela de EmailDeCadastro, damos um create class, vamos estender aqui a classe email e eu vou jogar no pacote models, porque isso é um modelo de email.

[01:01] O que vamos precisar para o nosso cadastro? Primeira coisa que vamos precisar é o token do usuário então vamos passar ele aqui, já que já temos ele salvo no banco e já usamos o atalho para criar um construtor com esse email. Então agora podemos a partir disso cadastrar o email ou cada um dos campos dele, em primeiro precisamos falar de quem o email foi enviado, então vamos lá eu vou criar aqui uma constante porque vai ser sempre a mesma coisa, “private static final REMETENTE”.

[01:43] E esse remetente vai ser o email de noreply da caelum, então caelum com email no-reply@caelum.com.br. Com essa anotação o enviador de email identifica que o que está fora do maior menor é o nome e o que está dentro é o email. Agora já podemos utilizar isso para fazer o nosso remetente, this.setfrom e usamos aqui o REMETENTE.

[02:22] Mas então está com um erro, eu esqueci de dizer que isso é uma string. Então agora temos aqui o nosso remetente, o que mais precisamos? De um assunto, eu vou fazer a mesma coisa aqui e vou fazer o assunto padrão, e o assunto vai ser confirmação de cadastro no site de produtos.

[02:54] Então agora já podemos setar o assunto, this.setSubject assunto, agora precisamos de duas coisas que vão ser um pouco mais complexo que são o destinatário e o corpo do email, o destinatário podemos pegar a partir do usuário e temos um usuário dentro do token, então vamos pegar aqui token.getUsuario, guardamos isso em uma variável agora temos nosso usuário. Para fazer o destinatário vamos fazer a mesma coisa que fizemos aqui.

[03:30] Então vamos lá, vamos fazer um String.format, vamos fazer por %s, e passamos aqui o usuario.getNome e usuario.getEmail. Temos nosso destinatário, podemos jogar isso no email, addTo, então agora já temos um destinatário do email, agora falta só o corpo, a primeira coisa que queremos mandar é aquele link lá, então vamos fazer o link.

[04:07] Vamos fazer ele também com um String.format, qual vai ser o formato padrão? Estamos fazendo tudo local, então eu vou colocar aqui local mas na hora de mandar para produção isso pode ser alterado, http://ocalhost:9000/usuario/confirma/%s/%s (http://ocalhost:9000/usuario/confirma/%s/%s).

[04:45] E aqui vai ser usuário.getEmail e token.getCodigo, porque eu estou mandando o email junto? Quando pegamos o token a partir do código, ele pode ser de qualquer usuário, então se eu ficar fazendo requisições para tokens aleatórios eu posso liberar um monte de usuário mesmo que eles não tenham sido liberados, que o usuário não tenha ido até o email dele e confirmado, clicando no link.

[05:20] Então passamos o email junto e podemos comparar se o usuário desse email pertence ao usuário que tem aqui dentro do token, então vamos guardar isso em uma variável e chamar ela de link. Agora precisamos do corpo do email,

esse corpo do email eu gostaria que fosse algo assim, String corpoFormat, vai ser algo assim “Olá %s, por favor clique no link a seguir para confirmar seu cadastro”.

[06:10] E geramos um link aqui, então e falamos que confirmar cadastro e fechamos o nosso ('%s'). Então agora já temos o formato do nosso corpo, eu posso extrair isso para uma constante, faz mais sentido, então é isso que eu vou fazer, eu vou jogar aqui como private static final String CORPO_FORMAT e vai ser igual a isso.

[07:00] Então agora precisamos gerar o corpo a partir desse formato que temos, String.format e temos o CORPO_FORMAT, quais são os argumentos que ele recebe aqui? Aqui é o nome do usuário e aqui é o link que já geramos ali em cima, então vamos lá, CORPO_FORMAT, aqui vai ser usuario.getNome e o link que temos.

[07:35] Agora é só colocar esse corpo no email, então setBodyHtml e manda o corpo, e temos um email pronto, vamos testar? Eu vou acessar aqui a nossa página de cadastro e vou criar um usuário qualquer, enquanto isso eu vou fazer uma última modificação que é alterar aqui que a mensagem de sucesso não é mais usuário foi cadastrado e sim que o email foi enviado, então “Um email foi enviado para confirmar seu cadastro”.

[08:17] Agora vamos vir aqui, acessamos a página e vamos cadastrar o marco@caelum.com.br marco, fomos para página de login, então parece estar tudo certo, eu vou vir aqui e olha só, o corpo do email, subject confirmação de cadastro from caelum no-reply, o corpo html, “Olá marco, porfavor clique no link a seguir”, aqui temos o link, se clicarmos no link nada vai acontecer, porque ainda não sequer fizemos a rota usuário/confirma, mas parece que está tudo certo.

[09:00] Então vimos aqui o que? Como criar um template de email fazendo uma extensão de classe, claro que existem bibliotecas que permitem uma criação um pouco mais poderosa, mais dinâmica como por exemplo FreeMarker, mas elas não vão ser abordadas nesse curso. Para continuar a próxima aula, vamos fazer a lógica de confirmação de cadastro do usuário.