07

Flash messages

Transcrição

[00:00] No vídeo anterior, refatoramos a nossa lógica de validação extraindo um trecho de código para deixar ele desacoplado do controller. Fazendo isso, nós nos adequamos um pouco melhor às boas práticas de programação e finaliza a nossa validação do produto.

[00:19] Que que vamos ter que fazer em seguida? Tínhamos visto, que tinha uma certa falta de interação com o usuário, com quem está cadastrando o produto ou com, por exemplo, quando formos cadastrar usuários, com quem vai se cadastrar. Falta tipo uma mensagem de sucesso ou de fracasso, que indique o que aconteceu com aquela ação.

[00:42] Então, o que que vamos fazer? Existe um recurso comum, que aparece com regularidade em frameworks web, que são as flash messages, que são mensagens instantâneas, que você coloca na requisição e ela é mostrada para o usuário, dura uma request e é removida de lá.

[01:02] E no flash essa funcionalidade está implementada e é relativamente tranquila de usar, basta você no seu controller utilizar o método flash, passando uma chave, no caso estamos cadastrando um produto com sucesso aqui, então eu vou usar a chave de sucesso, eu vou passar uma mensagem, “seu produto produto” + produto.getTitulo() + “foi cadastrado com sucesso”.

[01:46] Nós já registramos a mensagem na requisição, mas nós ainda não mostramos ela para o usuário. E como que mostramos essa mensagem para o usuário? Do mesmo modo que nós aqui no formulário utilizamos a tag main, como um envelope do conteúdo dela.

[02:07] Para que que servia esse envelope? Tínhamos aqui as declarações de que tipo de HTML estávamos usando, importação de JavaScript, de CSS, declaração do título da página, é uma coisa que vai aparecer em todas as telas.

[02:23] E todas as views e páginas podem ter mensagens de notificação para o usuário, então faria sentido colocarmos as notificações aqui, mas simplesmente jogar o código de notificação aqui não é legal. Aprendemos que dá para criar tags como a tag main, então vamos criar uma nova, uma tag de notificações.

[02:49] Como fazemos isso? Criamos um novo arquivo chamado “notificacoes.scala.html” porque vamos usar código Scala mas a página vai ser renderizada em HTML, jogamos aqui dentro do pacote views, que é onde ficam todas as nossas views.

[03:13] E que que fazemos com esse arquivo? Precisamos percorrer todas as mensagens de flash e mostrar elas para o usuário, então vamos fazer, que nem no Java, um for, mas a sintaxe é um pouquinho diferente, então para cada mensagem que tivermos dentro do nosso flash, vamos imprimir a mensagem, “@mensagem”.

[03:45] Mas tem um porém, nós não estamos utilizando aquela chave que indica se a mensagem é de sucesso ou fracasso. E o BootStrap, ainda por cima, tem uma classe que serve exatamente para isso, para mostrar notificações para o usuário e categorizar essas mensagens, essas notificações, como sucesso, fracasso, avisos importantes ou só uma informação genérica.

[04:10] Como que funciona essas mensagens? Vou criar aqui uma tag, que é um parágrafo para indicar que vamos mostrar uma frase para o usuário, é uma classe chamada “alerta”.

[04:25] Esse alerta também pode ser um alerta de sucesso, ele pode ser um alerta de perigo, quando formos mostrar uma mensagem de erro, mas nós não podemos simplesmente vim aqui e falar “Ah, é uma mensagem de sucesso”, quando colocarmos uma mensagem de fracasso no nosso controller, ela vai aparecer como mensagem de sucesso? Verdinha, bonitinha? Não pode, tem que aparecer vermelha, indicando tipo um perigo, aconteceu alguma coisa errada.

[04:58] Então, como que fazemos isso? Temos que pegar a chave de lá de dentro, ao invés de iterar pelas mensagens em si, podemos iterar pelo conjunto das chaves e pegamos cada uma das chaves.

[05:13] Então aqui, ao invés de utilizar a string sucesso, podemos pegar a chave e se usarmos a chave sucesso lá no nosso controlador, conseguimos mostrar um alerta de sucesso. Eu vou incluir aqui que o texto também vai ser estilizado de acordo com o tipo de mensagem.

[05:34] E como que pegamos a mensagem em si depois disso? Acessamos o mapa de flash e pegamos o item que está nessa chave. Vamos testar isso cadastrando um novo produto. Eu vou aqui cadastrar um livro de JavaScript, com o código livro de JavaScript, “livro-js”, preço 10 reais, cadastrar.

[06:02] Cadê a mensagem de sucesso? Então, criamos aqui a tag, mas nós não utilizamos ela em lugar nenhum. Pois é, como que se utiliza uma tag? Do mesmo modo que usamos a tag main falando “@main” e passando parâmetro para ela, vamos fazer a mesma coisa aqui, chamamos “@notificacoes” e só, porque ela não recebe nenhum parâmetro e não tem corpo. Então, acabou.

[06:33] Vamos ver se isso muda alguma coisa, vou cadastrar aqui um livro de Node agora porque nós já temos um de JavaScript, livro de Node, a descrição é “um livro de node” e o preço 10, como sempre. Olha só, a mensagem apareceu bonitinha, mas ela parece meio desalinhada, isso é porque aqui temos o nosso conteúdo da página dentro de um container e as notificações não estão dentro de um container.

[07:07] O que eu quero fazer aqui não é colocar ele dentro do conteúdo principal da página porque a notificação é uma coisa externa, é uma coisa secundária, então eu vou colocar ele dentro de uma seção, uma section que tenha a class container também e envolvemos tudo isso dentro dessa section.

[07:31] Agora sim ela vai aparecer bonitinha, caso cadastremos um produto com sucesso. Eu vou apagar aqui os livros de JavaScript e de Node e vou fazer a mesma coisa novamente, vamos cadastrar eles de volta. Então agora temos um livro de JavaScript, com código livro-js, preço 10.

[08:02] Olha só, ficou até um pouquinho arredondadas as bordas, ficou bem mais bonito e nós já mostramos aqui o nosso título do livro JavaScript, tudo bonitinho.

[08:13] Mas, ainda tem um porém, se viermos aqui e der um F5, não aparece nenhuma mensagem, mas se você inspecionar o HTML, a section ainda está aqui. Por que que ela está aqui? Nós não precisaríamos dessa section, é semanticamente errado.

[08:33] Então, é de bom tom, é uma boa prática, você só mostrar essa section caso existam mensagens. Como fazemos isso? Com o if, “if flash.isEmpty”, se flash estiver vazio, nós não mostramos. Então, se flash não estiver vazio, mostramos a nossa section. Agora parece ok. Se dermos um F5 aqui, a section some.

[09:04] Mostramos nossa mensagem de sucesso, ela está bonitinha. Será que funciona com o fracasso também? Vamos tentar, aqui nós já temos um caso de fracasso, quando o produto está inválido. Então, vamos fazer assim, vamos adicionar uma mensagem de flash aqui, só que não é a tag “error”, que é o mais comum de usar porque estamos usando baseado no BootStrap, então vamos usar a tag de fracasso “danger”, que é a classe de estilização de erro do BootStrap.

[09:42] Qual é o erro aqui? “Existem erros no seu formulário”, nós não sabemos qual é o erro, o erro em si vai aparecer nos campos, nós só queremos dar um destaque porque aconteceu um erro.

[10:00] Vamos ver se deu certo? Atualizamos aqui e vamos cadastrar um livro que sabemos que já está cadastrado, o livro de play com modelagem, livro de play, preço 11 reais. Olha só, “existem erros no seu formulário” e o erro está no campo código.

[10:22] Passamos por bastante coisa aqui, viu como criar uma tag nova, como inserir ela dentro de alguma outra tag e vimos como incluir mensagens de sucesso e fracasso para o nosso usuário. No caso, a chave pode ser o que quisermos, eu estou utilizando essas chaves para se adequar ao BootStrap.

[10:43] Nas próximas aulas, nós finalmente vamos começar a codar o próprio ponto de acesso, que vai permitir para o cliente recuperar a lista de produtos da nossa loja e vamos decidir um formato para elas e vamos também criar filtros para que o usuário possa pegar somente um tipo de produto ou um certo produto com determinado preço. Veremos todo esse conteúdo nos próximos capítulos.