03

Validação do pedido

Transcrição

[00:00] Nós já conseguimos receber as informações que o usuário preencheu, criar um pedido e salvar no banco de dados. Só que se o usuário não preencher nada, nós vamos salvar um pedido vazio e teremos que obrigar o usuário a preencher pelo menos esses três campos.

[00:15] Não faz sentido a pessoa fazer um pedido sem alguma coisa, porque isso não é um pedido. Se não tem informação nenhuma, não serve para a nossa aplicação. Então, o que vamos fazer? A primeira coisa é dar uma cara para esse formulário utilizando o Bootstrap.

[00:28] O Bootstrap tem um "component" chamado "forms" e nesse "component forms" ele tem duas classes que aplicamos no formulário para ela ganhar aquela cara de formulário bonito e tal, do mesmo jeito que queremos que fique aqui. Então é bem fácil estilizarmos o formulário.

[00:47] Nós copiamos. Estão vendo? Esse é um exemplo desse formulário aqui em cima. Ele está separando também os "inputs" em "divs" do jeito que estamos fazendo. E vamos colocar dentro da "div" que agrupa o "label" e o "input", ele colocou um "form-group".

[01:02] Então nós vamos colocar isso para todas essas "divs". "Form-group" para todo mundo. E nos "inputs" ele coloca essa classe "form-control". Então vamos colocar isso também nos "inputs" e no "textarea" também. Então, vamos lá!

[01:23] O que aconteceu com o nosso formulário? "Cancela". Vamos abrir "/formulario" mesmo. Pronto! Ele já ficou estilizado, já ficou bem parecido com

o que está aqui. A única coisa que falta é nós colocarmos isso em volta de um "card". Porque esse aqui tem uma borda e existe um componente do Bootstrap que nos dá essa borda.

[01:42] Esse componente é o "card". Aqui tem um exemplo mais simples, que é só você criar em volta do que você quer e colocar essa "div card". Então em volta do formulário eu vou adicionar "div card". Vou endentar o formulário. E vou aplicar essa classe "card-body" ao formulário porque ele está mostrando isso no exemplo e eu vou aplicar aqui também.

[02:07] Se eu apertar a tecla "F5" aqui, ele já fica legal. Só que aqui tem uma margem que nós temos que aplicar porque ele está colado no Jumbotron. Ele zerou o "margin botton". Na outra página queríamos isso, mas aqui não.

[02:23] Aqui queremos o "mt-3". Então atualizou e já ganhou margem. A última coisa que falta é estilizarmos o botão. Lembrando como que é. O botão aqui é o azul. Nós abrimos o botão para pegar o azul. O azul é um "btn-primary". Vou copiar essa classe toda. É só aplicar essa classe nesse "button".

[02:52] Agora ele ficou com o estilo. Agora está igual. Você vê que foi rápido de nós estilizarmos o formulário. Bem mais simples! O que eu quero fazer agora?

[03:01] Quando eu clicar em "Cadastrar", eu quero que ele não aceite o cadastramento, caso o usuário não tenha preenchido esses campos. O que acontece? Essa página de formulário, quando ele clicar em "Cadastrar" e não estiver preenchido, a página de formulário é mostrada para o usuário de novo com o que ele já preencheu anteriormente e com as mensagens de erro nos campos que ele tem que preencher.

[03:23] Para nós fazermos essas validações, você pode fazer na mão. Vai fazer o seguinte: "if(requisicao.getNomeProduto()". Você coloca isso dentro de um "if". Se for igual a nulo, significa que o usuário não preencheu. Você coloca assim, ou "requisicao.getNomeProduto().isEmpty())".

[03:43] E aí você faz o quê? “Return” para a página do formulário. Então é isso o que nós temos que fazer. Só que é isso que temos que fazer para todos eles, todos os atributos. Aí é muito chato ter que ficar implementando isso na mão. É ruim colocar isso também. Essa validação é associada a requisição. E onde estão os dados da requisição? Estão nessa classe.

[04:04] Então você pode, por exemplo, criar um método “public void valida()” e aqui dentro você verificar se foram preenchidos, o “nome”, a “urlProduto” e a “urlImagem”. Só que, mais uma vez, para não precisarmos ter que ficar reescrevendo esses “ifs” todos de validação – o que é uma coisa muito repetitiva - o Java já vem com algumas validações que você pode associar diretamente aos atributos utilizando o “Annotation”.

[04:35] Como nós fazemos isso? Abra o “Bean Validation”, essa página do Java EE e aqui tem uma explicação de como se usa o “Bean Validation”. São anotações e cada anotação representa uma validação que você vai aplicar, como nesse exemplo, a um determinado atributo.

[04:51] Então, por exemplo: “@Max”. Caso o seu atributo seja do tipo inteiro, você quer definir um valor máximo para esse atributo. Caso ele seja inteiro, você pode também definir um valor mínimo. Caso ele seja String, você pode dizer que ele não pode ser nulo - que é exatamente o nosso caso. Só que não são só nulos. Nós não queremos que ele também seja branco, vazio. Então eu vou utilizar uma anotação chamada de “@NotBlank”.

[05:17] Se liga na importação, “Javax.validation”. Então eu vou aplicar “@NotBlank” para esses três e pronto! Eu estou associando uma validação que gera um erro de validação! Caso esse campo seja nulo, tem até uma explicação, eu vou colocar o mouse em cima do “NotBlank”.

[05:37] O elemento não pode ser nulo. Tem que ter pelo menos um caractere que não pode ser um espaço vazio. Então esse “NotBlank” significa isso. Não pode ser nulo e nem vazio senão ele gera um erro de validação.

[05:49] Nós já associamos a esses atributos uma validação que verifica se esse campo foi preenchido. Só que precisamos pedir para o Spring executar isso porque o Spring está criando um objeto, mas não significa que ele automaticamente vai validar.

[06:07] Você tem que pedir para o Spring executar essas validações e existe uma anotação que podemos aplicar, chamada de "@Valid". Ela, basicamente, está integrando o Spring com o pedido de execução de validação. E aí o que o Spring vai fazer com isso?

[06:25] Vai nos dar, através de um objeto do tipo "BindingResult", vai nos dar o resultado dessa validação. Se tiver erros, ele vai nos dar através de um método chamado "hasErrors", vai nos dizer "true" ou "false" ("verdadeiro" ou "falso"), se teve erro ou não.

[06:51] Se teve erro, nós queremos que o formulário seja apresentado de novo para o usuário. Então "return pedido/formulario". Pronto! É isso que queremos que aconteça. Então, o que vai acontecer agora? Se o usuário fizer uma requisição e ele não tiver preenchido, o formulário vai ser apresentado de novo para o usuário. O formulário vai ser apresentado de novo. É isso que acontece. Estão vendo?

[07:17] Ele fica o tempo inteiro voltando para cá. Só tem um problema. O usuário não sabe o que ele errou. Nós não estamos dizendo que é obrigatório, por exemplo, a "UrlImagem". O usuário poderia imaginar que: "Se eu colocar a 'UrlProduto', o produto está bom. Não deveria ser obrigado a colocar a 'UrlImagem' ", ou seja, se eu preenchi isso e a descrição.

[07:36] Mas ele não está deixando porque a "UrlIMagem" é obrigatória. E tudo o que ele preencheu antes, foi embora. Ele esquece. Então, qual é a primeira coisa que vamos fazer? Nós vamos integrar o nosso formulário ao "Thymeleaf" para que ele possa gerenciar o que o usuário preencheu de forma que, primeiramente, caso o usuário tente cadastrar e tenha erro de validação, o

valor que ele preencheu volte a aparecer aqui. Então essa é a primeira coisa que vamos fazer.

[08:04] Vamos fazer o seguinte: vamos associar o formulário a um objeto do tipo “th:objetc” a um objeto do tipo “requisicaoNovoPedido”. Então é só você pegar o nome da classe “requisicaoNovoPedido” e colocar aqui, só que com o “R” minúsculo, como se fosse um objeto, uma variável.

[08:24] Então, “requisicaoNovoPedido” com “R” minúsculo. Peguei o mesmo nome da classe e coloquei aqui. Ele já vai reconhecer isso. Pronto! E aí, o que eu quero fazer? Eu quero associar esse “input” a um campo daquele objeto, a um “field” daquele objeto. E aí é só utilizarmos o asterisco.

[08:43] Lembrando que se eu usei o asterisco para acessar um determinado atributo, eu tenho que usar também esse aqui. Então esse símbolo, nós acessamos um atributo do “request”. O asterisco nos permite acessar um atributo desse objeto. Então como eu associei esse formulário a esse objeto, e eu quero associar esse “input” a um atributo desse objeto, eu uso o asterisco.

[09:13] Para isso serve essa sintaxe aqui. Então, qual é o atributo que eu quero associar o campo “nomeProduto”? Qual é o atributo? O atributo “nomeProduto”. Só que o “th:field” é bem legal. Porque como eu passei o nome “nomeProduto”, o que ele vai fazer? Ele automaticamente vai criar os atributos “name id” e vai gerenciar o “value” desse aqui.

[09:35] Ou seja, ele já vai preencher o “name” com o nome “nomeProduto”, o “id” com o “nomeProduto” e caso já tenha algum valor preenchido no “nomeProduto”, no atributo, ele já vai mostrar esse valor. Ou seja, eu não preciso mais preencher o “name”. Então isso que fizemos permite que o “Thymeleaf” passe a gerenciar ele.

[09:55] Então, caso o usuário já tenha preenchido o valor anteriormente, o valor volta a aparecer, por exemplo. Por quê? Porque ele gerencia o “value”

desse atributo. Então eu vou mostrar. Vou dar um “Cadastrar”. Beleza, não deu erro. Está funcionando.

[10:09] E volta para cá porque ele não preencheu os campos obrigatórios. Se eu preencher o nome do produto e clicar em “Cadastrar”, ele volta e fica preenchido. Por quê? Porque eu fiz uma requisição que bateu no novo. Eu validei, vi que tem erro e voltei.

[10:23] Só que quando eu voltei para essa página, eu volto com o mesmo objeto que já foi preenchido anteriormente. Então se já está preenchido, ele mostra preenchido no formulário. O que mais que eu tenho que fazer?

[10:32] Eu vou pegar esse “th:field” e vou aplicar a todos os outros “inputs”. Só que aqui vai ser “UrlProduto” e aqui vai ser “UrlImagem”. Lembrando que nós não precisamos mais preencher o atributo “name” porque esse “th:field” já tem inteligência para fazer isso. Então se fizermos a requisição, ele volta.

[10:55] Se você preencher a “Url”, ele volta preenchido a “Url” também. Se você colocar - eu vou apagar o produto para ele voltar - ele vem preenchido a “Imagem”. Se você inspecionar o elemento, olhe como o elemento está, ele tem o “id” e tem o “name”. O “value” já está lá definido, mas está vazio nesse caso.

[11:18] O que vamos nós fazer no próximo vídeo? Nós vamos mostrar mensagens de erros para o usuário. Para indicar para o usuário quais são os campos que estão com erro, para ele poder ajustar ou preencher direito o formulário e conseguir cadastrar um pedido. Mas isso no próximo vídeo. Até lá!