05

# Apresentando mensagens de erro

## Transcrição

[00:00] No vídeo anterior nós configuramos a validação. Se tentarmos criar ofertas aqui, vamos tomar erro - que conseguimos visualizar aqui no log, esse "MethodArgumentNotValidException".

[00:13] Uma coisa que não vimos ainda, porque eu estou criando essa funcionalidade em cima dessa tela, é que na hora que você tentar clicar em "Faça sua Oferta" de novo, para listar, ele não vai listar nada. Se você olhar no log, você vai ver uma série de erros, como se ele tivesse entrado em *loop* aqui, com relação à serialização.

[00:35] Isso acontece porque na hora em que ele pega a lista de pedido e tenta serializar para JSON, ele vai percorrer cada pedido que está aqui e vai tentar serializar isso para um objeto no JSON, um atributo.

[00:48] E na hora em que ele tentar serializar a lista de ofertas, ele vai abrir o objeto. Cada item de oferta e vai tentar serializar também, só que não queremos que isso aconteça - até porque oferta tem pedido, que é esse próprio pedido, então ele entra em *loop*.

[01:05] Eu vou colocar `@JsonIgnore` nesse atributo, `private List<Oferta> ofertas;` . Por quê? Porque eu só mesmo o conteúdo dos pedidos, eu não quero o conteúdo do usuário, não quero o conteúdo da lista de ofertas e do pedido. Se tentarmos acessar de novo, ele deve funcionar. Deixe-me limpar aqui o log. Se eu continuar tentando enviar a oferta, ele vai dar esse erro.

[01:28] E o que acontece? Eu vou abrir aqui com a tecla "F12", mais uma vez, estou rodando no Chrome. Na hora em que eu tento fazer a requisição, ele me dá um 400, um "error: Bad Request" e vem um conteúdo em JSON, que é aquela exceção serializada em JSON.

[01:49] E aqui tem todo o conteúdo do erro, inclusive todos os atributos que deram erro dentro desse "errors". Então tem o *errors* 0 e o *errors* 1. O 0 se refere ao campo valor, que não pode ser nulo, esse é o erro atual, e o data de entrega também não pode ser nulo.

[02:10] O que vamos fazer agora? Usando o Vue.js, usando a biblioteca que estamos utilizando, o Axios, para podermos fazer a requisição; vamos aprender a pegar o erro e vamos transformar esse erro. Vamos pegar esse erro que está vindo de cada *field* desse e vamos criar um objeto dentro do pedido, dizer que deu erro e colocar a mensagem lá. Para quê? Para conseguirmos mostrar a mensagem aqui embaixo dos *inputs* que deram erro.

[02:43] Como vamos fazer isso? Eu vou continuar reaproveitando a nossa lista de pedidos e vou colocar as informações do erro dentro do próprio pedido, do qual estamos criando uma oferta.

[02:59] Por quê? Porque se deu erro no envio de uma oferta desse pedido, os erros que vão aparecer aqui embaixo vão ser especificamente na tentativa de criar a oferta desse pedido e não dos outros. Então é algo que fica localizado aqui e eu não vou criar outros objetos ou outra estrutura de dados para poder organizar os erros. Eu vou aproveitar a própria estrutura que eu já tenho, que é esse *array* de pedidos.

[03:28] Como vamos fazer isso? Usando o `.catch()`, nós já usamos o `.then` para pegar a resposta e agora eu vou utilizar o *catch* para pegar o conteúdo desse erro.

[03:42] E vimos aqui, usando a tecla "F12", que o conteúdo JSON tem um atributo chamado "errors". Como eu faço para pegar esse objeto todo, para

depois acessar os erros? Eu faço `error`, que é exatamente esse que estamos recebendo na linha de cima, `.response`. Então `error.response.data` é esse conteúdo.

[04:09] Quando eu acessei o `.data`, eu acessei esse objeto. Se eu acessar o `errors`, eu vou acessar a lista de erros que vier. Se eu imprimir isso daqui, `console.log(error.response.data.errors);`. Deixe-me atualizar a página, ele vai imprimir aqui no console para mim.

[04:33] "Faça sua Oferta", ele listou. Eu vou tentar enviar e olhe o que ele me manda aqui, os erros. "field: 'valor'", não pode ser nulo é a mensagem, "field: 'dataDaEntrega'", não pode ser nulo também, a mesma mensagem.

[04:48] O que eu vou fazer com isso? Eu vou adicionar esses erros no próprio pedido que eu estou tentando criar a oferta. Mas esses atributos de erros não existem, assim como `ofertaEnviada` também não existia. O que eu vou fazer é criar. Eu abri chaves aqui, depois eu faço `pedido.resposta`. Deixe-me fechar os parênteses.

[05:16] Assim como eu criei o `pedido.ofertaEnviada`, eu vou fazer o `pedido.erros = {`, recebendo um objeto, e eu vou colocar dois atributos que se referem aos dois *inputs* que podem ter erros associados, que é o atributo `valor: ' '` - que vou colocar vazio e o atributo `dataDaEntrega: ' '` - que eu vou colocar vazio também, que é o nome desse *field* valor e *field* data da entrega.

[05:51] E o que eu vou fazer? Eu vou pegar esses erros, `error.response.data.errors`, eu vou percorrer esses erros usando o mesmo `forEach` que eu usei aqui para criar os atributos, `.forEach`; e para cada `(error => {` que eu encontrar, eu vou pegar o erro em si e vou pegar o *field* associado do erro, `error.field`.

[06:24] Esse `error.field` é valor e `dataDaEntrega`, porque ele vai percorrer cada erro. Então é exatamente isso! E valor e `dataDaEntrega` já estão criados dentro do `pedido.erros`.

[06:37] Aí, o que eu faço? `this.pedido.erros` no atributo `[error.field]`, que no caso um é `valor` e outro é `dataDaEntrega`. Eu vou jogar mensagem aqui, então vou fazer `= error.`. Qual é o campo? `DefaultMessage`. Eu não vou conseguir copiar, mas `defaultMessage;`.

[07:05] Estou associando o erro agora, agora eu o estou preenchendo. Eu quero mostrar esse erro, ou seja, quando o atributo `erros.valor` estiver preenchido com algum conteúdo, eu quero apresentar o erro, embaixo do *input*. Então aqui no *input* `Valor`, deixe-me abrir essa `div` para adicionar mais um atributo e na "Data da entrega" eu vou adicionar um atributo embaixo com a mensagem que eu quero.

[07:36] Se nós formos no formulário, já temos um aqui, um `input`; e embaixo ele tem uma `div`, onde ele mostra o erro também. Só que aqui é usando o Thymeleaf e agora eu vou usar a sintaxe do Vue.js.

[07:53] O que acontece? Quando essa mensagem vai aparecer? Ele está com essa classe `invalid-feedback`, ele vai aparecer quando tiver um erro mesmo. Então esse `th:errors="*{nomeProduto}"` não faz sentido, vou apagá-lo.

[08:05] Eu vou usar o `v-if=` do Vue e vou fazer. `v-if ="pedido.erros.valor"`, então o erro que está associado ao valor desse pedido. Se estiver preenchido, eu venho aqui embaixo e imprimo esse `{{pedido.erros.valor}}`.

[08:35] A mesma coisa vou fazer com a data da entrega. Vou copiar aqui `dataDaEntrega` e mudar. Ao invés de ser `pedido.erros.valor`, vou colocar `pedido.erros.dataDaEntrega`, com a classe `"invalid-feedback"`.

[08:48] Só que vimos no formulário mesmo que para aparecer, `<div class="invalid-feedback" th:erros="*{nomeProduto}">`, `Erros do nome do produto`, `</div>`, temos que adicionar no `input` uma classe chamada de `is-invalid`. Como eu vou fazer isso?

[09:01] Se você olhar na documentação do Vue, você vai ver que tem um atributo chamado de `v-bind:class`, que você consegue adicionar uma classe e que você pode passar aqui o valor dela, `="{'is-invalid':}"`, caso um determinado valor for verdadeiro.

[09:20] Então eu vou digitar o seguinte: `pedido.erros.dataDaEntrega!==' '}"` diferente de vazio. E esse mesmo `v-bind` eu vou utilizar nesse `input` do valor. Então vou jogar aqui, depois do `input`, só que não vai ser `dataDaEntrega`, vai ser `valor`.

[09:38] Então data da entrega é diferente de vazio e valor diferente de vazio. Ele adiciona a classe `is-invalid`. Quando ele adicionar a classe `is-invalid`, a mensagem vai aparecer embaixo. Será que vai funcionar? Vamos testar agora, porque não precisamos fazer mais nada.

[09:54] Vamos atualizar de novo aqui, clicar em "Faça sua Oferta". Ele abriu, não deu erro. Na hora em que eu clico em "Enviar Oferta", ele também não mostrou.

[10:05] Ele não consegue ver os erros de um atributo que está *undefined*. Vamos ver onde que está dando isso, não vamos conseguir ver aqui. Então vamos ver se colocamos os nomes certos aqui.

[10:16] Então `pedido` existe e `.erros` também existe; `pedido.erros` está aqui criado o objeto, recebe um novo objeto que criamos na hora que recebemos os pedidos. Nós fazemos `this.pedidos.erros`, é isso que estava dando erro.

[10:36] Vamos ver se era isso mesmo? Não! Olhe: `set property valor of undefined`, ele está dizendo que o valor não existe. Vamos ver, colocamos o

atributo `valor` aqui, `dataDaEntrega` do `pedido.erros`, e aqui está `pedido.erros.dataDaEntrega`. Apertamos as teclas "Ctrl + C", "Ctrl + V" e aqui está certo também. Só que estamos vendo no `erros.valor`, `pedido.erros.valor` no `v-if`, `pedido.erros.valor`.

[11:09] Não estou sabendo exatamente onde está esse `this.pedidos.erros`. Desculpa, não é isso. Não é `this.pedidos`, é `pedido`. Agora vai. Vamos ver? Agora funcionou. E funcionou só para esses *inputs* daqui, não funciona para os *inputs* de baixo. Quer dizer, funciona, mas essa mensagem só aparece.

[11:37] Se colocamos aqui algum valor inválido, ele vai mudar a mensagem. Se colocarmos um valor válido, ele não limpou a mensagem. Era para ter limpado isso. Como vamos fazer isso? Nós pegamos esse `pedido.erros`, resetamos esse valor antes de fazermos o *post*.

[11:56] Podemos fazer assim: `pedido.erros` agora está limpo. Então antes de fazer a requisição do *post* nós limpamos os erros. Se ele retornar mais erros, nós adicionamos de novo. Será que funciona? Vamos atualizar de novo, vou fazer a requisição, vou colocar um valor válido e agora sumiu. Funcionou. Agora conseguimos apresentar as mensagens de erro!

[12:23] Se eu colocar aqui na data de entrega um valor válido – aliás, esse está inválido também – ele faz a requisição, "Oferta Enviada", banco de dados está com valor. Sem comentário. Beleza, não tínhamos colocado comentário e nem dissemos que isso é obrigatório! Então é assim que adicionamos mensagens de erro aqui. Obviamente, eu fiz um jeito que, particularmente, eu acho bem simples.

[12:48] Você pode organizar o código para não ter que ficar repetindo esse `pedido.erros = { , valor: ' ', , dataDaEntrega: ' '`, colocando em um método separado. Mas está funcionando bem, eu achei bem simples essa solução.

[12:59] Até o próximo vídeo!