07

Trabalhando com validações

Transcrição

[00:00] A aplicação da Alura Shows não estava dando o que o usuário pode estar passando no campo do título e da mensagem do depoimento e, com isso, o Alex conseguiu inserir um código JavaScript para mostrar essa imagem do grupo do Anonymous. Então, nós precisamos verificar que informações o usuário pode estar passando no campo de título e de mensagem, com o objetivo de evitar esse ataque aqui de cross-site-scripting.

[00:24] Para isso nós vamos utilizar a interface validator que o próprio Spring oferece para nós. Então, vamos voltar no nosso projeto aqui e nós temos que criar essa nova classe para fazer a implementação dessa interface validator.

[00:37] Nós colocamos “Ctrl + N” e colocamos e que vamos criar essa nova classe e nós vamos guardar essa nova classe em um pacote de validação, que vamos chamar de validator. E o nome dessa nossa classe que vamos criar vai ser a depoimento validator. Nós voltamos aqui e implementamos essa interface validator aqui do Spring.

[01:01] E quando nós fazemos essa implementação dessa interface validator, nós somos obrigados a implementar esses dois métodos. O método suports e o método validate. Esse método suports recebe como argumento a classe do objeto que nós estamos querendo validar e vai indicar para nós se essa nossa classe depoimento validator que nós estamos criando, se ela consegue de fato validar os objetos dessa classe que serão recebidas como argumento.

[01:28] Então, vou só mudar o nome desse argumento, desse parâmetro, pra ser mais semântico. E o que nós queremos verificar? Nós queremos verificar se essa classe aqui passada como argumento é equivalente à classe depoimento, porque assim nós teríamos um retorno true e conseguimos seguir para a validação do objeto.

[01:50] Nós queremos verificar se essa classe que foi recebida como argumento é equivalente a classe depoimento. Então, nós chamamos aqui “depoimento.class” e nós queremos verificar esse grau de equivalência, nós queremos verificar se elas são assinaláveis. Nós chamamos o método aqui “isAssignableFrom(clazz)”.

[02:06] Com isso o que eu estou fazendo? Eu estou verificando se essa classe recebida como argumento nesse nosso método suports, é equivalente a essa nossa classe depoimento. Se elas forem equivalentes, tem o que? Tem um retorno true e nós conseguimos ter nesse método validate a validação do objeto em si.

[02:25] Nós temos o quê nesse método validate? Nós temos esses dois parâmetros, o parâmetro object, que de fato é o objeto que estamos querendo validar e temos esse outro parâmetro que é o parâmetro de erros que nós vamos estar guardando esses nossos erros de validação. Então vamos também trocar os nomes para ficar mais semântico.

[02:44] Aqui vamos colocar target e aqui colocamos errors. Eu preciso fazer a validação do que o usuário está passando no campo do título e da mensagem. Então, o Alex aqui pra ele ter sucesso nesse ataque dele, o que ele teve que fazer? Ele teve que abrir uma tag, colocar o nome script e fechar a tag. Então, a nossa estratégia para evitar esse ataque de cross-site-scripting é justamente o quê?

[03:10] Verificar se no campo do título e da mensagem tem alguma abertura ou fechamento de tags. Se tiver alguma abertura ou fechamento de tags, nós vamos falar pro usuário: "olha, usuário você está colocando alguma mensagem aqui meio estranha, então, nós não vamos aceitar". Essa é a validação que nós vamos fazer com o objetivo de evitar esse ataque de cross-site scripting. Esse nosso objeto target aqui nós temos que fazer um casting para dizer que esse objeto é do tipo depoimento.

[03:39] Nós vamos aqui e dizemos que esse nosso target, na verdade, é um objeto do tipo depoimento. Agora que eu já tenho esse objeto depoimento porque já fizemos a validação da classe, nós conseguimos fazer a validação desse objeto depoimento aqui. Nós temos que pegar desse objeto depoimento o título, o campo do título e da mensagem para verificarmos se contém a abertura e fechamento de tags.

[04:10] Nós vamos aqui e chamamos o gater. Colocamos “depoimento,getTitulo();” “Ctrl + 1” e assinalamos uma variável local, que chamamos de título e também chama o gater da mensagem “mensagem = depoimento.getMensagem();” “Ctrl + 1” e assinalamos o nome dessa nossa variável como sendo mensagem.

[04:36] Então, o que nós queremos é verificar? Se o campo do título aqui conter abertura, ou se ele contiver fechamento de tags, o que nós queremos fazer? Nós queremos rejeitar esse valor, então, nós chamamos esse nosso objeto do tipo erros para chamar esse método, indicando que vamos rejeitar esse valor, “erros.rejectValue(arg0, arg1);” e nós temos gente dois parâmetros reject value.

[05:09] O primeiro parâmetro é o atributo que nós temos na nossa classe depoimento que nós estamos querendo validar que, no caso, é quem? É o nosso título aqui. Então, nós colocamos o nome do atributo que está na nossa classe que também é título e aqui esse segundo parâmetro é a chave de identificação desse erro para que o Spring consiga encontrar depois uma mensagem personalizada para mandar para o nosso usuário.

[05:33] Nós ainda não temos essa mensagem, mas já vamos criar esse código de erro pra que o Spring depois consiga localizar. Nós vamos colocar esse código de erro, poderia ser, em tese qualquer nome, mas vamos colocar aqui, por exemplo, “erros.titulo”. Caso nós tenhamos aqui a abertura ou fechamento de tags no campo do título, nós estamos rejeitando esse valor. Nós temos que fazer essa mesma validação para a mensagem.

[06:01] Então, se a mensagem contiver a abertura ou fechamento de tags nós também vamos rejeitar esse valor e vamos falar pro usuário que a mensagem dele aqui não é válida. Nós chamamos nosso objeto “erros.rejectValue”. E passamos o primeiro parâmetro, que é o atributo, que temos na nossa classe depoimento, que é “mensagem” e aqui colocamos o código de erro que vamos estar passando para o nosso usuário. Colocamos “erros.mensagem”, depois nós vamos ter que criar uma mensagem personalizada com essa chave aqui que nós acabamos de colocar.

[06:45] Com isso nós criamos essa nossa classe depoimento validator. Nós temos que utilizá-la no nosso controller depoimento. Vamos colocar aqui “Ctrl + Shif + R” para localizarmos a classe depoimento Controller. Nós queremos que, quando esses parâmetros forem enviados do formulário, na criação desse objeto, seja feita essa validação. Então, nós temos que usar a notação valid da especificação bin validation.

[07:15] Aqui no nosso método enviar mensagem, quando esse nosso objeto depoimento for criado, nós queremos fazer a validação dele. Então, colocamos aqui “@Valid”. Coloca aqui “Ctrl + Shift + O” para fazer a importação certa. E agora como é que o Spring vai saber que nós estamos querendo fazer a validação de fato com a configuração que nós fizemos aqui nessa classe depoimento validator?

[07:43] Nós temos que especificar isso pro Spring. Nós fazemos essa especificação através da notação InitBinder. Vamos colocar aqui essa notação InitBinder pra justamente nós podermos chamar essa classe depoimento validator para que esse nosso objeto depoimento que foi criado seja validado de acordo lá com o depoimento validator, a classe que nós acabamos de criar com as configurações de validação que precisamos.

[08:07] Aqui nós vamos ter o nosso método, public. Vamos colocar aqui void, o retorno dele, e o nome InitBinder e ele vai receber como argumento o objeto do tipo aqui WebDataBinder. Nós vamos colocar esse Binder justamente pra citar esse validador. Então, colocamos setvalidator e vamos colocar mil. Colocamos aqui que queremos fazer essa validação baseada no que foi especificado na classe depoimento validator.

[08:39] Vamos só recapitular o que nós fizemos aqui. Nós criamos essa nossa classe depoimento validator. Essa classe depoimento validator implementa essa interface validator do Spring e nós somos obrigados a sobrescrever esses dois métodos. O método suports recebe como argumento aqui a classe do objeto que estamos querendo validar e retorna, se essa nossa classe depoimento consegue lidar com a validação dos objetos que são recebidos aqui como argumento dessa classe.

[09:07] Caso a classe que recebermos como argumento for equivalente a classe depoimento.class aqui, nós temos o que? O retorno true e conseguimos fazer a validação do objeto. Esse nosso objeto vai ser um objeto do tipo depoimento, nesse objeto depoimento nós queremos fazer a validação do campo do título e da mensagem, para verificar se tem aberturas ou fechamento de tags, se tiver abertura ou fechamento de tags nós vamos rejeitar esse valor, porque o Alex não vai conseguir colocar mais aquela tag script para poder inserir aquele código JavaScript.

[09:38] E uma vez que nós criamos essa classe depoimento validator, nós temos o que? Temos que voltar no nosso controller e especificar que, quando esse nosso objeto depoimento for criado com as informações vindas do formulário nós vamos validar essas informações e nós utilizamos essa notação InitBinder pra especificar para o Spring que nós estamos setando o nosso validator aqui, a validação que tem que ser feita, que tem que ser com base nessa nossa classe depoimento validator, porque nós especificamos a abertura ou fechamento de tags.

[10:11] Agora falta só nós configurarmos depois a mensagem personalizada, porque nós ainda não fizemos isso, nós só colocamos esses nossos códigos errors título e errors mensagem, mas nós ainda não criamos nada com relação a essas chaves, não fizemos ainda essa mensagem personalizada. Vamos fazer na próxima etapa para depois mostrar essas mensagens para o usuário.