01

Mostrando mensagens

Transcrição

[00:00] Nós criamos a nossa classe depoimento validator para fazer a verificação se o usuário está abrindo ou fechando tags no campo do título e no campo da mensagem. Então caso nós tenhamos essa condição, do usuário estar abrindo ou fechando tags, nós temos que mostrar uma mensagem para o usuário dizendo que a mensagem que ele colocou está inválida.

[00:18] Nós configuramos essas nossas chaves com as mensagens de erro, mas nós ainda não criamos essa mensagem de erro personalizada. Vamos fazer isso agora. Então para nós podermos criar essa mensagem de erro personalizada para poder ser gerenciada depois pelo spring, nós temos que criar um arquivo chamado de messages.properties, que deve ficar dentro dessa nossa pasta WEB-INF.

[00:41] Então, vamos clicar na nossa pasta WEB-INF, nós colocamos “Ctrl + N” e nós vamos criar esse arquivo messages.properties. Então nós pesquisamos aqui, arquivo em inglês é file, então nós vamos aqui, file, e o nome desse nosso arquivo vai ser messages.properties. Então eu tenho que passar agora aquelas chaves de erro que nós configuramos na classe depoimento validator, olha só quais eram elas, vou aproveitar e copiar aqui para não ter nenhum erro de digitação.

[01:08] Era o errors.titulo e a segunda chave de erro que nós tínhamos configurado era o errors.mensagem. Então agora vamos colocar uma mensagem mais descritiva caso o usuário coloque abertura ou fechamento de tags. Qual que é a mensagem personalizada que nós queremos colocar na jsp?

[01:28] Nós queremos colocar por exemplo, que o “título passado não é válido”. E nós colocamos, na mensagem, “a mensagem passada não é válida”. Então com isso, aqueles nossos códigos de erro vão ser substituídos por essas mensagens personalizadas.

[01:54] Só que o que é que acontece aqui no eclipse? Nós estamos trabalhando com caracteres especiais. E então, nós temos que o quê? Nós temos que, por padrão, o eclipse não configura esse arquivo para trabalhar com a codificação UTF 8 para nós podermos ter esses caracteres especiais. Então nós temos que fazer esse ajuste no eclipse, porque senão a nossa mensagem vai aparecer desconfigurada na jsp.

[02:16] Então, para isso, nós temos que vir aqui nesse arquivo messages.properties, nós clicamos com o botão direito do mouse, nós vamos aqui em properties e aqui o tipo de codificação, nós temos que colocar other e nós configuramos que nós queremos trabalhar com UTF 8, para que seja permitido o uso desses caracteres especiais que nós temos na nossa mensagem.

[02:37] Nós escolhemos UTF 8, apply and close, e a nossa mensagem apareceria assim no jsp. Vamos só corrigir, porque agora nós configuramos que estamos trabalhando com UTF 8, nós não devemos ter mais problema com esses caracteres especiais na nossa jsp. Vamos colocar aqui: válido, e aqui também, não é válido. Colocar aqui e aqui para finalizar.

[03:00] Agora nós já temos esse nosso arquivo do messages.properties. Nós precisamos configurar o spring para que ele saiba que ele tem que procurar essas mensagens nesse arquivo aqui, messages.properties, que está dentro da nossa pasta WEB-INF.

[03:14] Nós temos que ir no arquivo de configuração do spring e explicar isso para ele. Então vamos lá, voltar aqui no nosso projeto. Nós temos que vir aqui nesse nosso pacote infra, na classe casa de show configuration, e nós temos que

configurar um novo método para ensinar para o spring que as nossas mensagens estão dentro desse caminho WEB.INF e lá que o nosso arquivo começa com o nome messages.

[03:38] Para isso, eu vou colocar aqui anotação bin, para dizer que esse método vai ser um bin gerenciado pelo spring. Aqui nós colocamos esse nosso método, message source, e nós colocamos o mesmo nome aqui, message source. E aqui, nós temos que fazer a instância da classe, que vai receber o nome aqui de reloadable resource bundle, então vamos colocar aqui, new reloadable, vou dar um “Ctrl + espaço”, para ele já dar o autocomplete, é logo essa primeira classe, e nós colocamos ponto e vírgula, “Ctrl + 1” para ele assinalar essa variável local que nós vamos chamar de bundle.

[04:18] De bundle, só para ficar mais fácil. Vamos colocar aqui, bundle. Então agora que nós já temos esse nosso bundle, nós temos que especificar aonde que esse nosso message source, aonde que está a nossa fonte de mensagem de erro, está localizada. Então qual que é o nome desse caminho? Nós temos que chamar o método que é o set base name.

[04:40] Aqui eu tenho que especificar aonde que está esse arquivo que o Spring tem que procurar para pegar essas mensagens. Então nós o configuramos aonde? No WEB.INF. Então eu coloco aqui/WEB-INF e o nome do nosso arquivo sem a extensão, é só messages. Com isso, o spring já sabe que ele tem que procurar o arquivo messages.properties que está dentro da nossa pasta WEB-INF.

[05:00] Então agora, nós temos que o quê? Nós temos que especificar também que esse nosso arquivo messages.properties está configurado com aqueles caracteres especiais, para trabalhar com UTF 8. Então vamos configurar aqui que a codificação default desse nosso arquivo é o quê? É UTF 8. Com isso, basta nós retornarmos esse nosso bundle e o spring já vai saber agora que ele tem que procurar esse nosso arquivo, as mensagens de erro, no messages.properties, que está dentro da pasta WEB-INF.

[05:32] Feito isso, o que é que nós temos que fazer agora? Nós temos que voltar na nossa classe depoimento controler e verificar se aquela validação que nós fizemos apresentou ou não algum resultado de erro. Então vamos voltar aqui, vamos colocar “Ctrl + Shift + R” para pesquisar, essa nossa face, depoimento controller.

[05:54] Então aqui estamos o quê? Nós estamos fazendo avaliação dessa criação desse objeto depoimento de acordo com o que nós especificamos na classe depoimento validator. Então agora falta nós verificarmos se essa validação apresentou algum erro ou não. Então nós temos que vir aqui e passar mais um parâmetro para esse método enviar mensagem que é um objeto aqui do tipo binding result e nós vamos verificar.

[06:18] Se teve algum erro nessa nossa validação do depoimento, nós queremos o quê? Nós queremos voltar para jsp depoimento e disponibilizar aquela mensagem que nós configuramos no messages.properties para o usuário.

[06:31] Nós colocamos aqui: se esse objeto aqui, result, teve erros, chamamos o método aqui “if(result.hasErros()”, o que é que nós queremos fazer? Nós queremos retornar para jsp em depoimento. Então agora caso tenha erros, nós estamos retornando para jsp depoimento, o que é que nós temos que fazer? Ir na jsp depoimento e configurar a tag para pegar essa mensagem de erro para mostrar para o usuário.

[06:57] Então: “Ctrl + Shift + R”, nós colocamos aqui depoimento.jsp. E aqui, vamos colocar essa tag de erro logo abaixo dessas labels, no caso da label do título, e no label da mensagem. Então vamos começar pelo título. Para nós pegarmos essa mensagem de erro que nós configuramos, nós vamos utilizar aqui a própria tag do spring. Então nós vamos chamar aqui form errors. E aqui, essa mensagem de erro é vinculada ao título que nós havíamos configurado no nosso depoimento validator aqui com esse atributo título, que está na nossa classe depoimento e nós configuramos esse código de erro para pegar aquela mensagem personalizada.

[07:38] Eu venho aqui no nosso depoimento jsp e eu falo que esse formulário de erros é vinculado com um caminho, um atributo da nossa classe aqui, path, ele é configurado com o caminho, com o atributo da nossa classe, que é o título.

[07:56] E vamos aproveitar aqui e passar também estilo para essa mensagem para ficar mais visível para o usuário. Então vamos colocar aqui, vou chamar o CSS Class e vamos chamar a classe de erro do CSS para ficar vermelho e o usuário visualizar de uma forma mais fácil que o título que ele passou está inválido.

[08:15] Agora nós temos que fazer a mesma coisa aqui para mensagem. Então logo abaixo dessa label da mensagem, nós vamos utilizar essa tag, form errors, e essa form errors é vinculada a quem? Ela é vinculada ao nosso atributo da mensagem da nossa classe depoimento. E nós vamos passar aqui a cor também, vamos chamar a classe CSS de erro, para ficar com aquela cor vermelha de erro para o nosso usuário.

[08:43] Agora nós já terminamos essa nossa configuração e falta nós verificarmos se essa validação está sendo feita conforme deveria. Então, antes de nós rodarmos a nossa aplicação para fazer o teste, vamos no nosso banco de dados e nós precisamos o quê? Nós precisamos remover aquela mensagem com script que o Alex tinha colocado.

[09:05] Vou só colocar com o botão direito do mouse nessa tabela depoimento e nós temos aquela mensagem com o script. E no meu caso, ele deu o ID 15 para essa mensagem de script. Então vou fazer o seguinte: eu vou falar para deletar essa mensagem esse script para que nós possamos ver novamente a nossa jsp depoimento como ela estava antes desse ataque que o Alex realizou.

[09:27] Vou colocar para deletar, from wasp depoimento, onde o ID, where ID = 15. Então meu caso está com mais de 15, então vou pedir para deletar, ele deletou, então agora nós não devemos ter mais essa mensagem com script na nossa jsp depoimentos, e nós devemos conseguir visualizar a aplicação como ela estava funcionando anteriormente.

[09:52] Vamos agora que nós já deletamos, removemos essa mensagem com o script do nosso banco, vamos voltar aqui para o Tomcat e vamos inicializar o Tomcat para podermos fazer esse teste e verificar se a nossa validação de fato está sendo feito como deveria. Só esperar o Tomcat terminar de subir.

[10:13] Agora nós vamos voltar para a nossa aplicação da casa do show, e só clica aqui para ver se está tudo funcionando, e vamos voltar para nossa aba dos depoimentos que deve estar funcionando agora que nós removemos nossa mensagem com o script que o Alex tinha feito.

[10:29] Agora, vamos fazer um teste aqui e cadastrar uma mensagem comum, uma mensagem de um usuário que não vai nem abrir nem fechar tags. Então vou colocar aqui: teste, por exemplo, título e a mensagem eu vou colocar aqui: mensagem de teste. Então nem no nosso título nem na nossa mensagem nós temos agora abertura ou fechamento de tags. Então a nossa validação deve aceitar essa mensagem aqui que nós estamos passando e cadastrar no nosso banco sem nenhum problema.

[10:58] Vou colocar aqui enviar e vamos verificar se está certo. De fato, cadastrou, nós estamos visualizando a mensagem, o depoimento, que acabamos de passar. Agora, o que é que que eu vou fazer? Eu vou vir aqui e nesse campo do título, eu vou colocar uma tag de script.

[11:13] Vou aqui, abrir a tag, se eu abrir a tag, isso já deve ser rejeitado pela nossa classe depoimento validator. Então vou colocar aqui tag script, para nós colocarmos como se fosse um código JavaScript, e na mensagem, vou colocar uma mensagem de teste, só para nós vermos se a validação está sendo feita.

[11:31] Então vou colocar aqui enviar, e olha só agora o que é que nós temos, o título passado não é válido. De fato, essa mensagem agora não foi cadastrada no meu banco. Vamos fazer um novo teste agora? Eu vou colocar de novo aqui o título como sendo teste e eu vou colocar uma tag de script agora nesse nosso corpo da mensagem. Então vamos ver agora. Enviar.

[11:50] E olha só: a mensagem passada não é válida e, de fato, nós não estamos cadastrando mais essa informação no banco de dados. Então com isso, com essa validação de abertura e fechamento de tags, nós já conseguimos proteger

mais a aplicação da Alura Shows para esse ataque onde o usuário pode estar inserindo um código Java Script.