06

Validando nosso formulário

Transcrição

Vamos realizar um teste, entre no formulário e tente cadastrar um livro selecionando apenas os autores. O que acontece? Exato, o cadastro é feito. E não faz muito sentido cadastrar um livro sem título, páginas e até o preço. Precisamos melhorar isso.

Abra o arquivo `form.xhtml`, e vamos olhar nosso form. Para cada *input* na tela, temos os componentes `<h:inputText>` no código, e este componente possui um atributo, que ainda não usamos, chamado `required`. Quando setamos o valor dele para true, ele torna o campo obrigatório. Com isso conseguimos impedir o envio do formulário sem os campos necessários preenchidos. Mas como o usuário vai saber qual campo ele precisa preencher?

Vamos precisar também de um `<h:messages />` para que o usuário saiba o que foi validado e possa preencher o campo obrigatório com o valor necessário.

Mude no `form.xhtml` apenas o `<h:inputText />` do título conforme abaixo, suba seu servidor e tente cadastrar o livro *sem informar o título*.

```
<h:inputText value="#{adminLivrosBean.livro.titulo}" required="true" />
```

Perceba que de fato, o campo foi validado, mas a mensagem exibida não foi tão amigável, pois `j_idt2:j_idt7: Erro de validação: o valor é necessário.` não parece fazer muito sentido. E que valor é necessário?

O componente `<h:inputText />` tem um outro atributo chamado `requiredMessage` que nos permite definir a mensagem que devemos exibir para o usuário, assim, podemos ser específicos sobre a obrigatoriedade do campo.

Além disso, existem campos que precisam de uma validação específica. Por exemplo, os campos `preço` e `número de páginas`. Imagine cadastrar um Livro com páginas 0 (zero)? Faz sentido? Ou mesmo, podemos definir um valor mínimo para o preço. Para isso usamos outra tag do *jsf core* chamada `<f:validateLongeRange />` e `<f:validateDoubleRange />`. Como os nomes já indicam, `validateLongeRange` é para valores do tipo `Long` e `validateDoubleRange` para valores do tipo `Double / BigDecimal`.

Vamos alterar nosso código, deixando o formulário como a seguir:

```
<h:form>
    <h:messages />
    <div>
        <h:outputLabel value="Título" />
        <h:inputText value="#{adminLivrosBean.livro.titulo}"
                required="true" requiredMessage="O Título é um campo obrigatório!" />
    </div>
    <div>
        <h:outputLabel value="Descrição"/>
        <h:inputTextarea rows="4" cols="20" requiredMessage="A Descrição é Obrigatória"
                required="true" value="#{adminLivrosBean.livro.descricao}" />
    </div>
    <div>
```

```
            <h:outputLabel value="Número de Páginas"/>
            <h:inputText value="#{adminLivrosBean.livro.numeroPaginas}"
                    required="true" requiredMessage="O Número de Páginas é Obrigatório">
                <f:validateLongRange minimum="80" />
            </h:inputText>
        </div>
        <div>
            <h:outputLabel value="Preço"/>
            <h:inputText value="#{adminLivrosBean.livro.preco}"
                    required="true" requiredMessage="O Preço é Obrigatório">
                <f:validateDoubleRange minimum="20" maximum="150" />
            </h:inputText>
        </div>
        <div>
        <h:outputLabel value="Autores" />
        <h:selectManyListbox value="#{adminLivrosBean.autoresId}"
                    converter="javax.faces.Integer">
            <f:selectItems value="#{adminLivrosBean.autores}"
                    var="autor"
                    itemValue="#{autor.id}" itemLabel="#{autor.nome}" />
            </h:selectManyListbox>
        </div>
        <h:commandButton value="Cadastrar" action="#{adminLivrosBean.salvar}" />
    </h:form>
```

Se tentarmos enviar o formulário vazio agora, receberemos as mensagens que definimos pelo `requiredMesage` e também as validações de tamanho. Agora nosso formulário só é enviado se preenchermos corretamente todos os campos.

Realize o *Full Publish* novamente e teste se sua aplicação está funcionando corretamente.