

# CETEJ35 - Java Web - JAVA\_XXX (2024\_01)

[Meus cursos](#) / [CETEJ35 - Web \(2024\\_01\)](#) / [Semana 05: 30/09 a 06/10](#) / [Validação de Dados](#)

## Validação de Dados

✓ **Feito:** Ver

✓ **Feito:** Gastar pelo menos 20 minutos na atividade

**A fazer:** Passar pela atividade até o fim

**Fecha:** segunda-feira, 2 dez. 2024, 00:00

Nesta aula, vamos ver como usar recursos do *Bean Validation Framework* juntamente com o Spring Boot, Freemarker e Bootstrap para garantir que o usuário consiga visualizar os erros e corrigir os dados sempre que necessário.

## MANTENDO VALORES DIGITADOS

Ao executar o código na seção anterior você deve ter notado que os valores digitados antes da validação não são mantidos quando a página é recarregada. Isso acontece porque quando o formulário é enviado para o controlador, os dados informados são validados e depois são descartados. Quando a página é recarrega com os novos dados vindos do controlador, a solicitação possui as mensagens de erro, mas não os valores digitados anteriormente.

É importante manter os valores anteriores porque o usuário consegue ver o que está errado. Para fazer isso, tudo o que precisamos é armazenar os valores na variável memória e mostra-los quando a página for recarregada.

Primeiro, precisamos salvar na variável `memoria` os valores informados pelo usuário. Para isso, abra a classe `CidadeController` e adicione o código mostrado nas linhas 44 e 45 da Figura abaixo.

```
35         if (validacao.hasErrors()) {
36             validacao
37                 .getFieldErrors()
38                 .forEach(error ->
39                     memoria.addAttribute(
40                         error.getField(),
41                         error.getDefaultMessage())
42                 );
43
44             memoria.addAttribute("nomeInformado", cidade.getNome());
45             memoria.addAttribute("estadoInformado", cidade.getEstado());
46             memoria.addAttribute("listaCidades", cidades);
47
48             return ("/crud");
```

Esse código adiciona os atributos `nomeInformado` e `estadoInformado` na variável `memoria`. Esses atributos armazenam os valores informados pelo usuário, que foram recebidos pelo parâmetro `cidade`, e podem ser recuperados usando o método `getNome()` e `getEstado()`.

Em seguida, precisamos adicionar uma expressão condicional para exibir esses valores no input caso eles existam. Para isso, abra a página `crud.ftl` e acione às tags `input` a expressão `${nomeInformado!}` e `${estadoInformado}`, conforme

mostra a Figura a seguir.

```
29 <div class="form-group">
30   <label for="nome">Cidade:</label>
31   <input
32     value="{{$cidadeAtual.nome!}}{{$nomeInformado!}}"
33     name="nome"
34     type="text"
35     class="form-control {{$nome??}?then('is-invalid', '')}"
36     placeholder="Informe o nome da cidade"
37     id="nome">
38
39   <div class="invalid-feedback">
40     {{$nome!}}
41   </div>
42 </div>
43
44 <div class="form-group">
45   <label for="estado">Estado:</label>
46   <input
47     value="{{$cidadeAtual.estado!}}{{$estadoInformado!}}"
48     name="estado"
49     type="text"
50     class="form-control {{$estado??}?then('is-invalid', '')}"
51     placeholder="Informe o estado ao qual a cidade pertence"
52     id="estado">
53
54   <div class="invalid-feedback">
55     {{$estado!}}
56   </div>
57 </div>
```

Pronto! Teste seu código e veja como as coisas funcionam agora.

É importante lembrar que ao fazer os ajustes no método `criar()`, acabamos quebrando o método `alterar()`, que usava o `criar()` para salvar uma cidade alterada. Para corrigir isso, basta adicionar os parâmetros `org.springframework.validation.BindingResult` e `org.springframework.ui.Model` na assinatura do método `alterar` e repassar essas variáveis para o método `criar()`.



Observe que ao fazer isso o método `alterar()` não aproveita automaticamente a validação do método `criar()`.

O código desenvolvido nesta Seção está disponível no [Github](#), na branch `semana04-50-validacao-web-completa`.

[Retroceder](#)[Avançar](#)

◀ Verificação de aprendizado - Criando, Alterando e Excluindo

[Integração ▶](#)

 Contate o suporte do site 

Você acessou como RAFAEL ROCHA DA SILVA PROENCA (Sair)  
CETEEJ35 - Web (2024\_01)

Tema

Adaptable

Boost

Clássico

Campus

Apucarana

Campo Mourão

Cornélio Procópio

Curitiba

Dois Vizinhos

Francisco Beltrão

Guarapuava

Londrina

Medianeira

Pato Branco

Ponta Grossa

Reitoria

Santa Helena

Toledo

UTFPR

Ajuda

Chat UTFPR

Calendário Acadêmico

Biblioteca

e-Mail

Nuvem (OwnCloud )

Produção Acadêmica

Secretaria Acadêmica

Sistemas Corporativos

Sistema Eletrônico de Informação - SEI

Suporte ao usuário

Criação de curso

Comunidade

Português - Brasil (pt\_br)



Deutsch (de)

English (en)

Português - Brasil (pt\_br)

Resumo de retenção de dados

Baixar o aplicativo móvel.

 Dê um feedback sobre este software 

Universidade Tecnológica Federal do Paraná - UTFPR  
Suporte ao usuário

