

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.

BEAN VALIDATION FRAMEWORK

O [Bean Validation Framework](#) é uma especificação que define como validações podem ser implementadas para validar atributos em uma classe Java. O *Bean Validation Framework* fornece um [conjunto de anotações](#) que, quando empregado nos atributos, define restrições e mensagens de alerta.

Para usar o *Bean Validation Framework*, tudo que precisamos é inserir a dependência `spring-boot-starter-validation` no `pom.xml`.



```
36      <dependency>
37          <groupId>org.springframework.boot</groupId>
38          <artifactId>spring-boot-devtools</artifactId>
39      </dependency>
40
41      <dependency>
42          <groupId>org.springframework.boot</groupId>
43          <artifactId>spring-boot-starter-validation</artifactId>
44      </dependency>
45  </dependencies>
```

O [conjunto de validações é abrangente](#), mas vamos nos concentrar apenas naqueles mais importantes para o problema que temos no momento. Para inserir as validações, abra a classe `Cidade`, que define os atributos `nome` e `estado`. Imediatamente antes dos atributos, vamos usar algumas anotações para definir restrições, conforme mostra a Figura abaixo.

```
6  public final class Cidade {
7
8      @NotBlank(message = "Nome da cidade deve ser informado")
9      @Size(min = 5, max = 60, message = "O nome da cidade deve ter entre 5 e 60 caracteres")
10     private final String nome;
11
12     @NotBlank(message = "Sigla do estado deve ser informado")
13     @Size(min = 2, max = 2, message = "A sigla do estado está limitada a dois caracteres")
14     private final String estado;
```

A anotação `javax.validation.constraints.NotBlank` impede que sejam aceitos valores nulos ou espaços em branco, enquanto a anotação `javax.validation.constraints.Size` restringe um limite inferior e superior de caracteres. Ambas as anotações podem usar o atributo `message`, que define a mensagem que deve ser exibida caso a restrição não seja atendida.

Apesar de ser simples manter a mensagem direto na classe, isso não costuma ser uma boa ideia. Primeiro, qualquer ajuste na mensagem exige recompilação.

Segundo, isso dificulta a internacionalização da aplicação – afinal, a mensagem deve estar na mesma língua do usuário. Por fim, a alteração na mensagem pode causar problemas no código caso o desenvolvedor acabe alterando algo por engano.

Por isso, uma boa prática é colocar as mensagens em um arquivo separado. Para fazer isso, precisamos de um pouco de configuração. Abra a classe `br.edu.utfpr.cp.espjava.crudcidades.CrudCidadesApplication` e insira os métodos `messageSource()` e `getValidator()`, conforme mostra a Figura a seguir.

```

12 @SpringBootApplication
13 public class CrudCidadesApplication {
14
15     public static void main(String[] args) {
16         SpringApplication.run(CrudCidadesApplication.class, args);
17     }
18
19     @Bean
20     public MessageSource messageSource() {
21         ReloadableResourceBundleMessageSource messageSource = new ReloadableResourceBundleMessageSource();
22         messageSource.setBasename("classpath:messages");
23         messageSource.setDefaultEncoding("UTF-8");
24         return messageSource;
25     }
26
27     @Bean
28     public Validator getValidator() {
29         LocalValidatorFactoryBean bean = new LocalValidatorFactoryBean();
30         bean.setValidationMessageSource(messageSource());
31         return bean;
32     }
33 }

```

O método `messageSource()` define o nome do arquivo (linha 22) e o tipo de codificação (linha 23) que será usada no arquivo onde são armazenadas as mensagens. Esse método é anotado com `org.springframework.context.annotation.Bean`, sinalizando que o Spring Boot deve gerenciar esse método. O método `getValidator()` registra o arquivo de mensagens para ser usado juntamente com as validações.

Agora, precisamos transferir as mensagens para um arquivo próprio e mapear as validações com as mensagens. Por padrão, o arquivo é `resources/messages.properties`. O arquivo tem um formato de par de valores. Do lado esquerdo é definida uma chave e, do lado direito, um valor. A chave é usada pelas anotações de validação na classe para associar a mensagem que deve ser usada. O resultado por ser visto na próxima Figura.

```

src > main > resources > messages.properties
1 app.cidade.blank=0 nome da cidade deve ser informado
2 app.cidade.size=0 nome da cidade deve ter entre 5 e 60 caracteres
3 app.estado.blank=Sigla do estado deve ser informado
4 app.estado.size=A sigla do estado está limitada a dois caracteres

Cidade.java
6 public final class Cidade {
7
8     @NotBlank(message = "{app.cidade.blank}")
9     @Size(min = 5, max = 60, message = "{app.cidade.size}")
10    private final String nome;
11
12    @NotBlank(message = "{app.estado.blank}")
13    @Size(min = 2, max = 2, message = "{app.estado.size}")
14    private final String estado;

```

Observe que você tem liberdade para definir o nome da chave que será usada.

O código desenvolvido nesta Seção está disponível no [Github](#), na branch `semana04-20-validacao-javax`.


Retroceder

Avançar

◀ Verificação de aprendizado - Criando, Alterando e Excluindo

Seguir para...

Integração ▶

✉ Contate o suporte do site 



Você acessou como RAFAEL ROCHA DA SILVA PROENCA (Sair)
CETEJ35 - Web (2024_01)

- Tema
- Adaptable
- Boost
- Clássico
- Campus
- Apucarana
- Campo Mourão
- Cornélio Procopio
- 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

