05

Mãos na massa: Validando e convertendo dados

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Para realizar a validação dos dados, adicione mais algumas dependências ao projeto. Para isso, cole o seguinte XML dentro da tag **`<dependencies>`** do arquivo **pom.xml**:

```
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>1.0.0.GA</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>4.2.0.Final</version>
</dependency>
```

2) Para validar um produto, crie uma classe que será responsável por realizar essa validação, a **`ProdutoValidation`**, no pacote **`br.com.casadocodigo.loja.validation`** e implemente a interface **`Validator`** do Spring. Para adicionar os erros de validação, rejeitar se o campo estiver vazio, você pode usar o método **`rejectIfEmpty()`**, da classe **`ValidationUtils`**. E no método **`supports()`**, você pode usar o método **`isAssignableFrom()`** para verificar se classe recebida por parâmetro é de fato um **`Produto`**:

```
public class ProdutoValidation implements Validator {

    @Override
    public boolean supports(Class<?> clazz) {
        return Produto.class.isAssignableFrom(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        ValidationUtils.rejectIfEmpty(errors, "titulo", "field.required");
        ValidationUtils.rejectIfEmpty(errors, "descricao", "field.required");

        Produto produto = (Produto) target;

        if(produto.getPaginas() <= 0) {
            errors.rejectValue("paginas", "field.required");
        }
    }

}
```

3) Para que o `ProdutosController` possa reconhecer o validador, crie o método **`initBinder()`**, que ficará responsável por vincular o validador com o *controller*. Esse método recebe um `WebDataBinder`, e então, anote-o com **`@InitBinder`**, para que o Spring o chame automaticamente. A partir do parâmetro `WebDataBinder`, utilize o método `addValidators()` para adicionar o `ProdutoValidation` para o `ProdutosController`:

```
@InitBinder
public void initBinder(WebDataBinder binder) {
    binder.addValidators(new ProdutoValidation());
}
```

4) Com a estrutura para realizar a validação pronta, falta executá-la. Ainda no `ProdutosController`, faça com que o produto seja validado, então vá até o método **`gravar()`** e, no parâmetro do tipo **`Produto`**, anote com **`@Valid`**:

```
@RequestMapping(method=RequestMethod.POST)
public ModelAndView gravar(@Valid Produto produto, RedirectAttributes redirectAttributes) {

    produtoDao.gravar(produto);

    redirectAttributes.addFlashAttribute("sucesso", "Produto cadastrado com sucesso!");

    return new ModelAndView("redirect:produtos");
}
```

5) Além de validar o produto, você precisa saber quais foram os erros obtidos durante a validação. Para pegá-los, adicione o parâmetro **`BindingResult`** no método **`gravar()`**. Durante a validação, o Spring vai coletar todos os erros e irá adicionar nesse parâmetro mas para que isso funcione, é **necessário adicioná-lo logo em seguida do objeto que será validado**, em outras palavras, o **`BindingResult`** tem que ser declarado logo após o produto:

```
@RequestMapping(method=RequestMethod.POST)
public ModelAndView gravar(@Valid Produto produto, BindingResult result,
    RedirectAttributes redirectAttributes) {

    produtoDao.gravar(produto);

    redirectAttributes.addFlashAttribute("sucesso", "Produto cadastrado com sucesso!");

    return new ModelAndView("redirect:produtos");
}
```

6) Por fim, antes de executar qualquer instrução, faça um `if`, passando como argumento o método `hasErrors()`, do `BindingResult`, e se houver erros, retorne o método **`form()`**, ou seja, redirecione o usuário para o formulário. Dessa forma, você fará com que o usuário volte para o formulário todas as vezes que ocorrer uma falha de validação:

```
@RequestMapping(method=RequestMethod.POST)
public ModelAndView gravar(@Valid Produto produto, BindingResult result,
        RedirectAttributes redirectAttributes) {

    if (result.hasErrors()) {
        return form();
    }

    produtoDao.gravar(produto);

    redirectAttributes.addFlashAttribute("sucesso", "Produto cadastrado com sucesso!");
```

```
        return new ModelAndView("redirect:produtos");
    }
```