07

Convertendo o Autor

Transcrição

Uma vez que aprendemos sobre `Converters` criando a classe `CalendarConverter`, podemos evoluir essa ideia para melhorar o nosso código. Observe que temos um caso parecido, a nossa lista de autores do nosso formulário. Além de já usarmos um `Converter` que é do próprio **JSF**, a `javax.faces.Integer`, estamos trabalhando sempre com o ID's do `Autor` em vez de utilizar o próprio objeto `Autor`.

Vimos como fazer isto anteriormente. Criaremos a classe `AutorConverter` no pacote `br.com.casadocodigo.loja.converters`, já vamos anotá-la com `@FacesConverter("autorConverter")`. Definimos o nome "autorConverter" para poderemos referenciá-lo em outros locais caso seja necessário. Implementaremos também a interface `Converter`.

Usando o atalho do Eclipse, pressione `Ctrl + 1` em cima do nome da classe, e escolha a opção `add unimplemented methods`.

```
@FacesConverter("autorConverter")
public class AutorConverter implements Converter {

    @Override
    public Object getAsObject(FacesContext context,
            UIComponent component, String id) {

        return null;
    }

    @Override
    public String getAsString(FacesContext context,
            UIComponent component, Object autorObject) {

        return null;
    }

}
```

Começaremos pelo método `getAsObject()` ... Queremos recuperar o autor como um objeto, então faremos `new Autor` e da instância obtida, chamaremos o `setId` que recebe o `id` - basta transformar de `String` para `Integer` com `Integer.valueOf`. Já temos a ideia principal, só iremos verificar antes, se a String recebida não é nula ou vazia. O método ficará assim:

```
public Object getAsObject(FacesContext context, UIComponent component, String id) {
    if (id == null || id.trim().isEmpty())
        return null;

    Autor autor = new Autor();
    autor.setId(Integer.valueOf(id));
```

```
    return autor;
}
```

Seguiremos para o método `getAsString()` que fará justamente o inverso. Queremos recuperar o id do autor no formato `String`. Já recebemos o objeto `Autor` como `Object`, agora, faremos um *casting* de volta para `Autor`. Deste, chamaremos o `getId` e depois o `toString()` - para transformá-lo em texto. Se o objeto recebido estiver **null**, teremos uma `NullPointerException`, então, evitaremos esse caso desde o início do método.

Juntando tudo, nosso código do `getAsString` ficará assim:

```
public String getAsString(FacesContext context, UIComponent component, Object autorObject) {
    if (autorObject == null)
        return null;

    Autor autor = (Autor) autorObject;
    return autor.getId().toString();
}
```

Agora que conseguimos converter de `Autor` com o nome `autorConverter`, podemos voltar ao arquivo `form.xhtml` e trocar o `javax.faces.Integer` simplesmente para `autorConverter`. Além disso, o próprio `value` do componente select, precisa de uma lista de ID's, porém, a entidade `Livro` já possui uma lista de autores relacionada. Podemos fazer uso dessa lista diretamente, mudando simplesmente para:

```
value="#{adminLivrosBean.livro.autores}"
```

Dentro da tag, também precisamos realizar um mudança em `<f:selectItems />`, tirando o `id` no `itemValue`, e usando diretamente o `autor`. Com as alterações, o código ficou assim:

```
<div>
    <h:outputLabel value="Autores" />
    <h:selectManyListbox value="#{adminLivrosBean.livro.autores}"
                converter="autorConverter" id="autores">
        <f:selectItems value="#{adminLivrosBean.autores}" var="autor"
                    itemValue="#{autor}" itemLabel="#{autor.nome}" />
    </h:selectManyListbox>
    <h:message for="autores" />
</div>
```

Além disso, o uso do converter também facilitará nossa vida no `AdminLivrosBean`. Abra a classe e dentro do método `salvar()`, encontraremos um `for` que percorre a lista de `id`s dos autores. Como não estamos mais usando a lista na tela, removeremos seguinte trecho:

```
for (Integer autorId : autoresId) {
        livro.getAutores().add(new Autor(autorId));
}
```

E também o atributo `List<Integer> autoresId`, junto com seu *getter* e *setter*: `getAutoresId` e `setAutoresId`.

```
private List<Integer> autoresId = new ArrayList<>();

// deixar os demais métodos e remover os get e set abaixo
public List<Integer> getAutoresId() {
    return autoresId;
}

public void setAutoresId(List<Integer> autoresId) {
    this.autoresId = autoresId;
}
```

Faça novamente um *Full Publish* e tente realizar o cadastro de um Livro para ver o que acontece.

Você deve ter recebido um erro de validação. Trata-se de um erro bem comum no JSF quando estamos usando `Converters`. Isso acontece porque o JSF espera que os valores da lista sejam comparados com os valores passados para o nosso conversor. Assim, o **JSF** precisa comparar o autor convertido com o autor da lista. Uma vez que ele não consegue, ele dará um erro de validação.

Mas por que estamos recebendo esse erro, se sabemos que o autor convertido existe na lista?

Lembre-se que quando não especificamos uma forma de comparação de objetos no *Java*, ele irá comparar os objetos usando o `==` - mas `este irá comparar a referência de memória, o que não servirá neste caso. Então, precisaremos mudar a forma de comparação, e criar nosso método` equals() `na classe` Autor`, para que o **JSF** use o novo método corretamente.

Como o Eclipse já realiza esta ação, peça para o Eclipse gerar o `equals()`. Entre na classe `Autor` e pressione `Ctrl + 3`, em seguida, será aberta uma caixa. Digite `equals` e selecione a opção "Generate hasCode() and equals()", depois, selecione apenas a opção do `id` e desmarque o "nome" caso esteja selecionado. Clique em `OK` e veja o código gerado.

```
@Override
public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((id == null) ? 0 : id.hashCode());
        return result;
}

@Override
public boolean equals(Object obj) {
        if (this == obj)
         return true;
        if (obj == null)
         return false;
        if (getClass() != obj.getClass())
         return false;
        Autor other = (Autor) obj;
        if (id == null) {
         if (other.id != null)
             return false;
        } else if (!id.equals(other.id))
         return false;
        return true;
}
```

Vamos pedir para o Eclipse gerar o `toString()`. Com isto, todo o cadastro de Livros já deve estar funcionando e você também deve ser capaz de relacionar os Autores com o Livro, o que é feito internamente usando os objetos em vez de primitivos.

Desta forma, nosso código fica muito mais Orientado a Objetos de fato, e mais elegante.

Realize um *Full Publish* e teste seu cadastro. Depois disso, vá para os exercícios e tire suas dúvidas no fórum.

Bons estudos!