03

Conversores complexos

Vamos criar o conversor para Compra que gera o xml a seguir:

```
String resultadoEsperado = "<compra estilo=\"novo\">\n"
        + "  <id>15</id>\n"
        + "  <fornecedor>guilherme.silveira@caelum.com.br</fornecedor>\n"
        + "  <endereco>\n"
        + "    <linha1>Rua Vergueiro 3185</linha1>\n"
        + "    <linha2>8 andar - Sao Paulo - SP</linha2>\n"
        + "  </endereco>\n"
        + "  <produtos>\n"
        + "    <produto codigo=\"1587\">\n"
        + "      <nome>geladeira</nome>\n"
        + "      <preco>1000.0</preco>\n"
        + "      <descrição>geladeira duas portas</descrição>\n"
        + "    </produto>\n"
        + "    <produto codigo=\"1587\">\n"
        + "      <nome>geladeira</nome>\n"
        + "      <preco>1000.0</preco>\n"
        + "      <descrição>geladeira duas portas</descrição>\n"
        + "    </produto>\n"
        + "  </produtos>\n"
        + "</compra>";
```

Para isso adicionaremos um novo teste no nosso CompraTest:

```
@Test
public void deveUsarUmConversorDiferente() {
}
```

Criamos uma compra e pedimos para o xstream registrar um conversor que criaremos:

```
Compra compra = compraDuasGeladeirasIguais();

XStream xstream = xstreamParaCompraEProduto();
xstream.registerConverter(new CompraDiferenteConverter());
String xmlGerado = xstream.toXML(compra);

assertEquals(xmlEsperado, xmlGerado);
```

Crie o CompraDiferenteConverter, implemente a interface Converter. Implemente o método canConvert para suportar objetos do tipo Compra.

Ao serializar (marshal), adicione um atributo chamado estilo com o valor novo.

Logo depois, faça o cast do objeto que recebeu para o tipo Compra.

Inicie a tag id, coloque o valor dela e feche. Não esqueça de gerar os getters (getId e getProdutos).

```
writer.startNode("id");
context.convertAnother(compra.getId());
writer.endNode();
```

Adicione a tag fornecedor com o valor "guilherme.silveira@caelum.com.br". Adicione as tags endereco, linha1 e linha2 para chegar no trecho de xml a seguir:

```
<endereco>
  <linha1>Rua Vergueiro 3185</linha1>
  <linha2>8 andar - Sao Paulo - SP</linha2>
</endereco>
```

Inicie a tag produtos e use o contexto para converter os produtos da compra.

Crie um método de teste para nosso xml esperado:

```
@Test
public void deveUsarUmConversorDiferente() {
    String resultadoEsperado = "<compra estilo=\"novo\">\n"
            + "  <id>15</id>\n"
            + "  <fornecedor>guilherme.silveira@caelum.com.br</fornecedor>\n"
            + "  <endereco>\n"
            + "    <linha1>Rua Vergueiro 3185</linha1>\n"
            + "    <linha2>8 andar - Sao Paulo - SP</linha2>\n"
            + "  </endereco>\n"
            + "  <produtos>\n"
            + "    <produto codigo=\"1587\">\n"
            + "      <nome>geladeira</nome>\n"
            + "      <preco>1000.0</preco>\n"
            + "      <descrição>geladeira duas portas</descrição>\n"
            + "    </produto>\n"
            + "    <produto codigo=\"1587\">\n"
            + "      <nome>geladeira</nome>\n"
            + "      <preco>1000.0</preco>\n"
            + "      <descrição>geladeira duas portas</descrição>\n"
            + "    </produto>\n"
            + "  </produtos>\n"
            + "</compra>";
```

Crie uma compra, o xstream com a configuração padrão que sempre utilizamos, e adicione o converter customizado através do método registerConverter. Serialize e confirme que o resultado da compra de duas geladeiras iguais é o mesmo que o xml esperado.

Compartilhe o código de seu método marshal aqui.

Responda

INSERIR CÓDIGO FORMATAÇÃO