02

Coleções implícitas e Polimorfismo

Esse capítulo continua onde paramos com o capítulo anterior. Caso você não possua o projeto pronto, pode baixá-lo e importá-lo no Eclipse a partir deste arquivo (https://s3.amazonaws.com/caelum-online-public/XSTREAM/xstream-fim-do-3.zip).

Continuando a serialização da Compra e Produto, vamos ver agora algumas variações do que é possível fazer com nossos tipos. As vezes um xml não possui o elemento de agrupamento, como no caso do elemento produtos dentro de uma Compra. Nesse instante nosso xml está assim:

```
<compra>
  <id>15</id>
  <produtos>
    <produto codigo="1587">
      <nome>geladeira</nome>
      <preco>1000.0</preco>
      <descrição>geladeira duas portas</descrição>
    </produto>
    <produto codigo="1588">
      <nome>ferro de passar</nome>
      <preco>100.0</preco>
      <descrição>ferro com vaporizador</descrição>
    </produto>
  </produtos>
</compra>
```

Mas sem o elemento produtos, teríamos:

```
<compra>
  <id>15</id>
  <produto codigo="1587">
    <nome>geladeira</nome>
    <preco>1000.0</preco>
    <descrição>geladeira duas portas</descrição>
  </produto>
  <produto codigo="1588">
    <nome>ferro de passar</nome>
    <preco>100.0</preco>
    <descrição>ferro com vaporizador</descrição>
  </produto>
</compra>
```

Isto é, a coleção produtos está implícita dentro da tag compra. Não é necessário aparecer a tag produtos para que o leitor do xml entenda que cada produto deve estar dentro da lista de produtos. O XStream suporta coleções implícitas através do método addImplicitCollection, dizendo que na classe Compra.class, o atributo produtos é implícito.

```
XStream xstream = xstreamParaCompraEProduto();
xstream.addImplicitCollection(Compra.class, "produtos");
```

Com o código acima, a serialização de nossa compra de geladeira e ferro passa a ser sem a tag produtos, como desejávamos. Assim como o addImplicitCollection, existem métodos similares para outros tipos de coleções e arrays.

Outro ponto importante no processo de serialização é o trabalho com polimorfismo. Imagine um cenário onde temos duas novas classes: Livro e Musica, ambas herdando de Produto:

```
package br.com.caelum.xstream;

public class Livro extends Produto {

    public Livro(String nome, double preco, String descricao, int codigo) {
        super(nome, preco, descricao, codigo);
    }

}
```

```
package br.com.caelum.xstream;

public class Musica extends Produto {

    public Musica(String nome, double preco, String descricao, int codigo) {
        super(nome, preco, descricao, codigo);
    }

}
```

Agora criamos uma compra com dois desses produtos:

```
private Compra compraComLivroEMusica() {
    Produto passaro = new Livro("O Pássaro Raro", 100.0, "dez histórias sobre a existência", 158
    Produto passeio = new Musica("Meu Passeio", 100.0, "música livre", 1590);
    List<Produto> produtos = new ArrayList<>();
    produtos.add(passaro);
    produtos.add(passeio);

    return new Compra(15, produtos);
}
```

Alteramos a configuração do XStream básica com os alias para Musica e Produto:

```
private XStream xstreamParaCompraEProduto() {
    XStream xstream = new XStream();
    xstream.alias("compra", Compra.class);
    xstream.alias("produto", Produto.class);
    xstream.alias("livro", Livro.class);
    xstream.alias("musica", Musica.class);
    xstream.aliasField("descrição", Produto.class, "descricao");
    xstream.useAttributeFor(Produto.class, "codigo");
    return xstream;
}
```

O resultado da serialização era o que esperávamos, uma tag indicando musica e uma tag indicando livro:

```
<compra>
  <id>15</id>
  <produtos>
    <livro codigo="1589">
      <nome>O Pássaro Raro</nome>
      <preco>100.0</preco>
      <descrição>dez histórias sobre a existência</descrição>
    </livro>
    <musica codigo="1590">
      <nome>Meu Passeio</nome>
      <preco>100.0</preco>
      <descrição>música livre</descrição>
    </musica>
  </produtos>
</compra>
```

Mas ainda pensando em polimorfismo, até agora estamos utilizando uma ArrayList para a lista de produtos. E se nossa implementação da interface List contida na classe Compra fosse uma LinkedList? Como indicar no XML que durante a deserialização a tag produtos deveria vir na forma de uma LinkedList e não de uma ArrayList? Vamos testar a serialização e ver como o XStream sinaliza no XML que a implementação da interface List é uma LinkedList. No nosso último método de criação de livro e música, alteramos a configuração da Compra:

```
Produto passaro = new Livro("O Pássaro Raro", 100.0, "dez histórias sobre a existência", 1589);
Produto passeio = new Musica("Meu Passeio", 100.0, "música livre", 1590);
List<Produto> produtos = new LinkedList<>();
produtos.add(passaro);
produtos.add(passeio);

return new Compra(15, produtos);
```

Rodamos o teste e ele está quebrado. Claro. O XStream precisou notificar a tag produtos:

```
  <produtos class="linked-list">
    <livro codigo="1589">
      <nome>O Pássaro Raro</nome>
      <preco>100.0</preco>
      <descrição>dez histórias sobre a existência</descrição>
    </livro>
    <musica codigo="1590">
      <nome>Meu Passeio</nome>
      <preco>100.0</preco>
      <descrição>música livre</descrição>
    </musica>
  </produtos>
</compra>
```

Podemos alterar nosso xml esperado para indicar que esperamos uma LinkedList ao invés de uma ArrayList:

```
String resultadoEsperado = "<compra>\n"
        + "  <id>15</id>\n"
        + "  <produtos class=\"linked-list\">\n"
        + "    <livro codigo=\"1589\">\n"
        + "      <nome>O Pássaro Raro</nome>\n"
        + "      <preco>100.0</preco>\n"
        + "      <descrição>dez histórias sobre a existência</descrição>\n"
        + "    </livro>\n"
        + "    <musica codigo=\"1590\">\n"
        + "      <nome>Meu Passeio</nome>\n"
        + "      <preco>100.0</preco>\n"
        + "      <descrição>música livre</descrição>\n"
        + "    </musica>\n"
        + "  </produtos>\n"
        + "</compra>";
```

O XStream sabe que por padrão as Lists são implementadas através de ArrayLists, portanto somente quando a implementação escolhida for outra, ele adicionará o atributo class para indicar qual classe ele deve instanciar durante a deserialização. Caso você queira (re)definir a implementação padrão de qualquer interface usada no processo de de/serialização, você pode fazê-lo através do método addDefaultImplementation.

Note como o XStream é bem maleável em relação ao trabalho com coleções e polimorfismo. Mesmo assim podem surgir situações em que tais configurações não resultem no XML exato que gostaríamos de gerar. Nesses casos podemos customizar completamente a conversão de XML <=> Objeto, que veremos em breve.