13

Desafio Parte 2: XStreamDataFormat e split

Na primeira parte do desafio, usamos Camel para acessar o serviço HTTP periodicamente (chamado de *HTTP Polling*). Vimos que o serviço devolve uma lista de negociações em XML:

```
<list>
  <negociacao>
    <preco>226.5</preco>
    <quantidade>13</quantidade>
    <data>
      <time>1444089600000</time>
      <timezone>Etc/UTC</timezone>
    </data>
  </negociacao>

...
</list>
```

Nesta parte, vamos **transformar o XML em objetos Java**. Você já sabe que esse processamento se chama de ***unmarshal*** e, para isso, é preciso usar uma biblioteca. No mundo Java, já temos duas bibliotecas que se destacam: JAX-B e XStream. Neste exercício, faremos uso do XStream.

1) O primeiro passo é adicionar as dependências no `pom.xml`. Adicione dentro da tag `dependencies` do `pom.xml`:

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-xstream</artifactId>
    <version>2.16.1</version>
</dependency>

<dependency>
    <groupId>com.thoughtworks.xstream</groupId>
    <artifactId>xstream</artifactId>
    <version>1.4.8</version>
</dependency>
```

2) Depois disso, é preciso criar a classe para mapear o XML. Nele, teremos uma lista que será mapeada para `java.util.List` e várias negociações. No entanto, ainda não temos algo para representar uma negociação, para tal **crie uma nova classe**:

```
package br.com.caelum.camel;

import java.util.Calendar;

public class Negociacao {

    private double preco;
    private int quantidade;
    private Calendar data;
```

```
    //getters e setters

    @Override
    public String toString() {
        return "Negociacao [preco=" + preco + ", quantidade=" + quantidade + "]";
    }
}
```

3) Agora podemos começar a usar XStream. No método `main()` da classe `RotaHttpPollingNegociacoes`, instancie o XStream e adicione um *alias* para mapear o elemento `<negociacao>` do XML para a classe `Negociacao.class`:

```
final XStream xstream = new XStream();
xstream.alias("negociacao", Negociacao.class);
```

4) O próximo passo será alterar a rota. Na Camel DSL, logo depois do método `convertBodyTo(..)`, chame o método `unmarshal` e passe o `XStreamDataFormat`:

```
..
unmarshal(new XStreamDataFormat(xstream)).
..
```

O método `unmarshal` usará o XStream e todas as negociações serão adicionadas em uma `java.util.List`. Teremos como resultados um lista de negociações (`List<Negociacao>`).

5) Sabendo da lista de negociações disso podemos dividi-la usando. Logo depois do `unmarshal` adicione:

```
split(body()). //cada negociação se torna uma mensagem
```

6) Por ora, comente as linhas nas quais enviávamos o XML para ser persistido em arquivo e no lugar coloque um marcador de "fim de rota":

```
                end(); // só deixa explícito que é o fim da rota...

//                setHeader(Exchange.FILE_NAME, constant("negociacoes.xml")).
//                to("file:saida");
```

Tente implementar essa rota e execute o código. Depois, fique atento ao console.