01

# Validando tags de um XML

### Transcrição

No capítulo anterior, criamos um documento `vendas.xml` contendo algumas informações, como a forma de pagamento e os produtos. Também escrevemos um bloco de código que nos retorna o atributo `moeda`, o `nome` dos produtos e o seu `preco`. Porém, imagine que por um erro, na criação desse XML, tivéssemos adicionado uma tag `<nomes>` ao invés de `<nome>`. Rodando novamente o `Sistema.java`, recebemos uma `NullPointerException`.

Isso acontece pois estamos pegando o produto por meio do nome da tag, que é justamente o `nome`. Sendo assim, se a tag tem um nome diferente, o Java não consegue encontrá-la e retorna nulo. Seria interessante encontrarmos uma maneira de garantir que nosso XML tenha o formato correto. Uma solução seria incluirmos diversos operadores `if` que fizessem essa verificação, mas será que essa é a melhor alternativa?

Pensando nisso, o W3C definiu um formato de arquivo utilizado para descrever a sintaxe do XML, como quais tags podem ou não estar presentes. Os arquivos que validam um XML são chamados de XML Schema Definition, ou XSD. O Eclipse nos permite criar um arquivo desse tipo prssionando "Ctrl + N" e pesquisando por "XLM Schema File". Vamos nomeá-lo como `formatoVenda.xsd`, e teremos o seguinte corpo básico:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/formato
</schema>
```

Antes de começarmos a trabalhar com ele, removeremos alguns atributos que por enquanto nos são desnecessários, resultando no seguinte código:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
</schema>
```

Para facilitarmos o aprendizado, simplificaremos o nosso XML, mantendo somente as tags `<venda>` e `<formaDePagamento>` e comentando o resto do código.

```
<?xml version="1.0" encoding="UTF-8"?>
<venda moeda="real">
    <formaDePagamento>Cartão</formaDePagamento>
</venda>
<!--  <produtos>
        <produto>
            <nome>Livro de xml</nome>
            <preco>29.90</preco>
        </produto>
            <produto>
            <nome>Livro de O.O. java</nome>
            <preco>29.90</preco>
        </produto>
    </produtos>
</venda>-->
```

No `formatoVenda.xsd`, precisaremos definir a existência de uma tag chamada `venda`. Para isso, usaremos uma tag `<element>` e passaremos o atributo `name="venda"`. Em seguida, criaremos outra tag `<element>`, dessa vez com o atributo `name="formaDePagamento"`, definindo mais uma tag que deverá estar presente em `vendas.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="venda" />
    <element name="formaDePagamento" />
</schema>
```

Mas como definiremos que `vendas.xml` deve utilizar `formatoVendas.xsd`? Isso é possível por meio do atributo `xmlns` (de "xml namespace"), para o qual daremos o nome `xsi` ("xml schema instance"). Pressionando "Enter", o atributo `xmlns:xsi="http://www.w3.org/2001/XMLSchema">` será adicionado automaticamente.

```
<?xml version="1.0" encoding="UTF-8"?>
<venda moeda="real" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <formaDePagamento>Cartão</formaDePagamento>
</venda>
<!--  <produtos>
        <produto>
            <nome>Livro de xml</nome>
            <preco>29.90</preco>
        </produto>
            <produto>
            <nome>Livro de O.O. java</nome>
            <preco>29.90</preco>
        </produto>
    </produtos>
</venda>-->
```

Também precisaremos passar a localização do nosso arquivo XSD, o que faremos utilizando o atributo `noNamespaceSchemaLocation`.

```
<?xml version="1.0" encoding="UTF-8"?>
<venda moeda="real" xsi:noNamespaceSchemaLocation="formatoVenda.xsd" xmlns:xsi="http://www.w3.o
    <formaDePagamento>Cartão</formaDePagamento>
</venda>
<!--  <produtos>
        <produto>
            <nome>Livro de xml</nome>
            <preco>29.90</preco>
        </produto>
            <produto>
            <nome>Livro de O.O. java</nome>
            <preco>29.90</preco>
        </produto>
    </produtos>
</venda>-->
```

Agora que definimos a existência das tags `<venda>` e `<formaDePagamento>`, vamos definir o tipo delas. No caso, ambas serão strings.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="venda" type="string" />
    <element name="formaDePagamento" type="string" />
</schema>
```

Apesar de termos feito essa definição em `formatoVenda.xsd`, a `<venda>` em `vendas.xml` é um elemento simples e não pode ter atributos ou tags. A nossa `DocumentBuilderFactory` não pegou esse erro pois, quando estamos trabalhando com arquivos XSD, precisamos informar que queremos validar os documentos com `setValidating(true)` e `setNamespaceAware(true)`.

Além disso, precisaremos setar um atributo que indicará qual a linguagem de schema que estamos utilizando. Nesse caso, passaremos as URLS http:java.sun.com/xml/jaxp/properties/schemaLanguage e http://www.w3.org/2001/XMLSchema (http://www.w3.org/2001/XMLSchema).

```
DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
fabrica.setValidating(true);
fabrica.setNamespaceAware(true);
fabrica.setAttribute("http://java.sun.com/xml/jaxp/properties/schemaLanguage", "http://www.w3.or
```

Se executarmos o código dessa forma, receberemos um erro indicando que `venda` é um elemento simples e, portanto, não pode ter atributos ou elementos filhos. Para corrigirmos isso, precisaremos alterar o tipo de `venda`, que não será mais uma string, mas sim um tipo complexo. Em `formatoVenda.xsd`, definiremos uma tag `<complexType>`, dentro da qual teremos outra tag `<attribute>` cujo nome ( `name` ) será `moeda`, e cujo tipo ( `type` ) será `string`.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="venda">
        <complexType>
            <attribute name="moeda" type="string"/>
        </complexType>
    </element>
    <element name="formaDePagamento" type="string" />
</schema>
```

Feito isso, se rodarmos nosso `Sistema`, teremos somente o erro indicando que `venda` não pode ter elementos filhos. Corrigiremos isso adicionando uma sequência ( `<sequence>` ) de tags que `<venda>` pode possuir, dentre elas `<formaDePagamento>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="venda">
        <complexType>
            <sequence>
                <element name="formaDePagamento" type="string" />
            </sequence>
```

```
            <attribute name="moeda" type="string"/>
        </complexType>
    </element>
</schema>
```

Assim, teremos definido toda a estrutura do nosso arquivo XML atual. Se rodarmos novamente nosso código, não teremos nenhum erro. Conseguimos validar a parte básica desse XML, e no próximo vídeo aprenderemos a validar os produtos e seus atributos.