01

Percorrendo os eventos

Transcrição

Nesse capítulo conheceremos outra forma de ler arquivos XML. Para não perdermos o código criado nos capítulos anteriores, criaremos uma nova classe `LeArquivoXmlTerceiraForma.java`, que armazenaremos no pacote `Teste` e que terá um método `main()`. Nesse método, criaremos um `InputStream` que receberá a referência do arquivo `vendas.xml`.

```
public class LeArquivoXmlTerceiraForma {
    public static void main(String[] args) throws Exception {
        InputStream is = new FileInputStream("src/vendas.xml");
    }

}
```

A terceira classe que implementa uma lógica de ler arquivos XML é chamada de `XMLEventReader`, responsável por gerar uma espécie de lista de todos os eventos, como abertura e fechamento de tags. Para conseguirmos essa classe, criaremos uma `factory` que receberá uma nova instância de `XMLInputFactory`. A partir dela, chamaremos `createXMLEventReader()`, passando como parâmetro o nosso *input stream* (`is`), e armazenaremos em uma variável `eventos`.

```
public class LeArquivoXmlTerceiraForma {
    public static void main(String[] args) throws Exception {
        InputStream is = new FileInputStream("src/vendas.xml");
        XMLInputFactory factory = XMLInputFactory.newInstance();
        XMLEventReader eventos = factory.createXMLEventReader(is);
    }

}
```

A partir de `eventos`, utilizaremos o método `nextEvent()` para armazenar a ocorrência do próximo evento executado, que guardaremos em uma variável `evento`. Com isso, poderemos realizar várias verificações. Como essa lógica também é baseada em eventos, ela será bastante parecida com a classe `ProdutoHandler`. Nesse caso, checaremos se um elemento foi iniciado (`isStartElement()`), a partir do elemento inciado (`asStartElement()`), pegaremos o seu nome (`getName()`) e o nome da tag sem o namespace (`getLocalPart()`). Se esse nome for igual a `produto`, criaremos uma instância de `Produto`.

```
public static void main(String[] args) throws Exception {
    InputStream is = new FileInputStream("src/vendas.xml");
    XMLInputFactory factory = XMLInputFactory.newInstance();
    XMLEventReader eventos = factory.createXMLEventReader(is);

    XMLEvent evento = eventos.nextEvent();

    if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equals("prod
        Produto produto = new Produto();
    }
}
```

Como queremos repetir esse bloco de código para todos os eventos, usaremos o `while` com a condição de que essa repetição deve acontecer enquanto existirem eventos, o que verificaremos com `hasNext()`. Para utilizarmos a variável `produto` futuramente, vamos declará-la no nosso método `main()`.

```
public static void main(String[] args) throws Exception {
    InputStream is = new FileInputStream("src/vendas.xml");
    XMLInputFactory factory = XMLInputFactory.newInstance();
    XMLEventReader eventos = factory.createXMLEventReader(is);
    Produto produto = new produto();

    while(eventos.hasNext()) {
        XMLEvent evento = eventos.nextEvent();

        if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equals("
            produto = new Produto();
        }
    }
}
```

Prosseguindo, trataremos também da tag `nome`. Nesse caso, avançaremos um evento com o `nextEvent()` e criaremos uma string `nome` que receberá a chamada de `asCharacters().getData()`, armazenando os dados de texto do evento. Por fim, chamaremos `produto.setNome()` passando o `nome` que recebemos do XML.

```
public class LeArquivoXmlTerceiraForma {
    public static void main(String[] args) throws Exception {
        InputStream is = new FileInputStream("src/vendas.xml");
        XMLInputFactory factory = XMLInputFactory.newInstance();
        XMLEventReader eventos = factory.createXMLEventReader(is);
        Produto produto = new Produto();

        while(eventos.hasNext()) {
            XMLEvent evento = eventos.nextEvent();

            if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equa
                produto = new Produto();
            }else if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart(
                evento = eventos.nextEvent();
                String nome = evento.asCharacters().getData();
                produto.setNome(nome);
            }
        }
    }
}
```

Também precisamos tratar da tag `preco`. Para isso, repetiremos o processo anterior, dessa vez armazenando o retorno de `asCharacters().getData()` em uma variável `conteudo`, que por sua vez será passada para o método `Double.parseDouble()`, retornando um double `preco`. Enfim, chamaremos `produto.setPreco()` para armazenar esses dados.

```
public static void main(String[] args) throws Exception {
    InputStream is = new FileInputStream("src/vendas.xml");
    XMLInputFactory factory = XMLInputFactory.newInstance();
    XMLEventReader eventos = factory.createXMLEventReader(is);
    Produto produto = new Produto();

    while(eventos.hasNext()) {
        XMLEvent evento = eventos.nextEvent();

        if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equals("
            produto = new Produto();
        }else if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equ
            evento = eventos.nextEvent();
            String nome = evento.asCharacters().getData();
            produto.setNome(nome);
        }else if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equ
            evento = eventos.nextEvent();
            String conteudo = evento.asCharacters().getData();
            double preco = Double.parseDouble(conteudo);
            produto.setPreco(preco);
        }
    }
}
```

Feito isso, trataremos do fechamento da tag `produto`. Para isso, verificaremos a ocorrência desse tipo de evento com `isEndElement()` e, a partir dele (`asEndElement()`), verificaremos se seu nome é igual a `produto`. Nesse caso, com o método `add()`, adicionaremos `produto` à lista `produtos`, uma variável do tipo `ArrayList<>` que criaremos no método `main()`.

Ao término da execução desse loop, faremos um `System.out.println()` dos nossos `produtos`.

```
public static void main(String[] args) throws Exception {
    InputStream is = new FileInputStream("src/vendas.xml");
    XMLInputFactory factory = XMLInputFactory.newInstance();
    XMLEventReader eventos = factory.createXMLEventReader(is);
    Produto produto = new Produto();
    List<Produto> produtos = new ArrayList<>();

    while(eventos.hasNext()) {
        XMLEvent evento = eventos.nextEvent();

        if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equals("
            produto = new Produto();
        }else if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equ
            evento = eventos.nextEvent();
            String nome = evento.asCharacters().getData();
            produto.setNome(nome);
        }else if(evento.isStartElement() && evento.asStartElement().getName().getLocalPart().equ
            evento = eventos.nextEvent();
            String conteudo = evento.asCharacters().getData();
            double preco = Double.parseDouble(conteudo);
            produto.setPreco(preco);
        }else if(evento.isEndElement() && evento.asEndElement().getName().getLocalPart().equals
            produtos.add(produto);
```

```
            }

        }

        System.out.println(produtos);
    }
```

Como retorno, teremos o conteúdo do nosso XML:

```
[Nome:Livro de xml
Preço:29.9
, Nome:Livro de O.O. java
Preço:29.9
]
```

Dessa forma, conseguimos acessar o arquivo XML percorrendo a lista de eventos dele. Em questões de performance, essa forma e a anterior são bastante parecidas. No próximo vídeo vamos melhorar um pouco o nosso código.