02

Acessando tags de um XML

Transcrição

Vamos começar a trabalhar com o arquivo XML utilizando o Java. Primeiro, criaremos um novo pacote `br.com.alura.Teste`, e todo código que escrevermos e precisarmos testar será armazenado nele. Dentro do pacote, criaremos uma classe `Sistema.java`.

```
package br.com.alura.Teste;

public class Sistema {

}
```

Se a classe `Sistema` é o ponto inicial da nossa aplicação, precisará ter um método `main()`. No Eclipse, podemos digitar apenas "main" e pressionar "Ctrl + Espaço" para autocompletar o método.

```
public class Sistema {
    public static void main(String[] args) {

    }

}
```

Para carregarmos o arquivo XML na memória, precisaremos de um *document builder*, ou seja, uma classe que constrói documentos. Podemos conseguir isso por meio de uma `DocumentBuilderFactory`, que importaremos de `javax.xml.parsers` e representaremos por uma `fabrica`. Chamaremos então o método `newInstance()` para obtermos uma referência desse tipo.

```
import javax.xml.parsers.DocumentBuilderFactory;

public class Sistema {
    public static void main(String[] args) {
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
    }

}
```

Da `fabrica`, podemos pedir um novo construtor de documentos com `newDocumentBuilder()`. Com "Ctrl + 1", podemos associar essa chamada a uma variável local que chamaremos de `bulder`.

```
public class Sistema {
    public static void main(String[] args) {
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = fabrica.newDocumentBuilder();
    }

}
```

Feito isso, pediremos para esse `builder` parsear (`parse()`) o nosso arquivo XML, passando como argumento o diretório dele.

```
public class Sistema {
    public static void main(String[] args) {
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = fabrica.newDocumentBuilder();
        builder.parse("src/vendas.xml");
    }

}
```

O Eclipse apontará alguns problemas pois os métodos `newDocumentBuilder()` e `parse()` lançam algumas exceções checadas, ou seja, que somos obrigados a tratar. Como não estamos nos preocupando com exceções no momento, usaremos a opção "Add throws declaration".

```
public class Sistema {
    public static void main(String[] args) throws SAXException, IOException, ParserConfiguratio
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = fabrica.newDocumentBuilder();
        builder.parse("src/vendas.xml");
    }

}
```

Por fim, tendo carregado o documetno na memória, vamos salvá-lo a uma variável `document´.

```
public class Sistema {
    public static void main(String[] args) throws SAXException, IOException, ParserConfiguratio
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = fabrica.newDocumentBuilder();
        Document document = builder.parse("src/vendas.xml");
    }

}
```

Já temos a referência para o nosso documento, e agora queremos, de alguma maneira, ler a `formaDePagamento` e imprimir o valor na tela com `System.out.println()` . Mas como acessaremos cada tag do nosso XML?

A partir de `document` podemos acessar alguns métodos, e um deles é o `getElementosByTagName()` . Se passarmos o nome da tag para esse método, ele procurará uma referência para cada elemento. Sendo assim, passaremos `formaDePagamento` como parâmetro e salvaremos todas as referências encontradas em uma variável `formasDePagamento` - no plural, já que o método `getElementosByTagName()` nos traz uma lista com todas as tags com o nome passado para ele, ainda que no nosso documento tenhamos apenas uma.

Pensando nisso, chamaremos `formasDePagamento.item(0)` para buscarmos somente a primeira referência dessa lista, associando-a a uma variável `fdp` (abreviação de `formaDePagamento`). Isso nos retornará um objeto do tipo `Node` , que não possui os métodos que nos interessam. Portanto, faremos um casting para `Element` , outra interface do Java que

pode ser importada de `org.w3c.dom` e que possui o método `getTextContent()` para pegarmos o conteúdo de texto do XML.

```
public class Sistema {
    public static void main(String[] args) throws SAXException, IOException, ParserConfiguratio
        DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = fabrica.newDocumentBuilder();
        Document document = builder.parse("src/vendas.xml");

        NodeList formasDePagamento = document.getElementsByTagName("formaDePagamento");
        Element fdp= (Element) formasDePagamento.item(0);
        String formaDePagamento = fdp.getTextContent();
        System.out.println(formaDePagamento);
    }

}
```

Feita essa construção, executaremos nosso código clicando no botão de "Play". Como retorno, teremos:

> Cartão

Se mudarmos o conteúdo da tag `<formaDePagamento` em `vendas.xml` para "Débito", salvarmos o arquivo e rodarmos o código novamente, teremos como retorno:

> Débito

Já se quiséssemos pegar o nome do produto, ao invés de procurarmos pela tag `formaDePagamento` em `getElementsByTagName()`, procuraríamos pela tag `nome`. Assim, nosso retorno seria:

> Livro de xml

Ou seja, é bastante fácil acessar qualquer tag de um documento XML utilizando as classes que o próprio Java nos fornece. O Java possui diversas especificações, que são conjuntos de interfaces cujo objetivo é desempenhar alguma função. Para manipular XML, temos uma especificação chamada JAXP (Java API for XML Processing).

No próximo vídeo aprenderemos a trabalhar com mais de um produto na nossa venda.