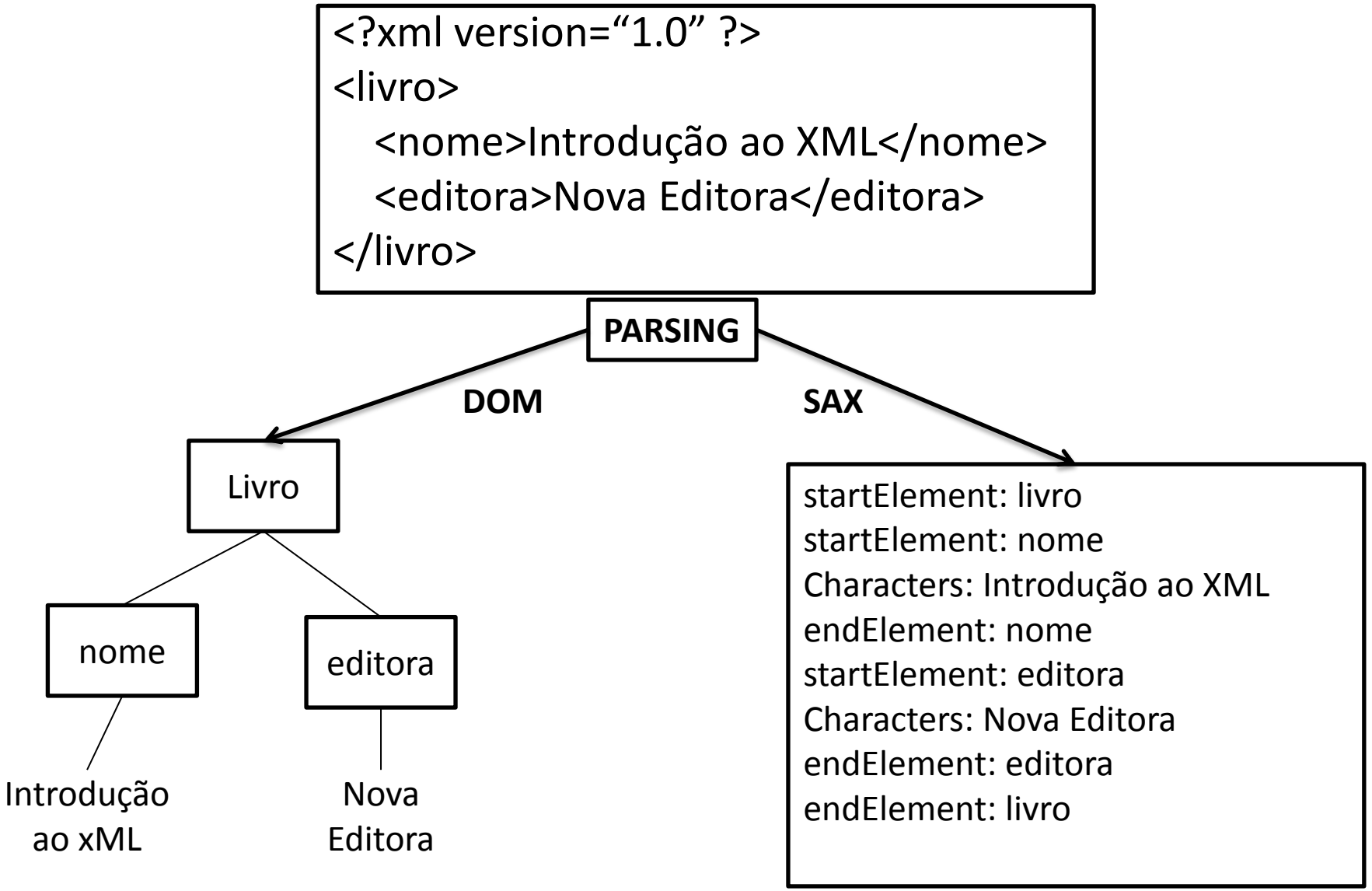
Java e XML

Professor Vinícius Costa

SAX x DOM

- São APIs para interpretar XML
- SAX – Simple API fo XML
  - Decompõe um documento XML em uma sucessão linear de chamadas de métodos
- DOM – Document Object Model
  - Decompõe um documento XML em uma árvore de nós

SAX x DOM


<?xml version="1.0" ?>
<livro>
<nome>Introdução ao XML</nome>
<editora>Nova Editora</editora>
</livro>
PARSING
DOM
SAX
Livro
nome
editora
Introdução ao xML
Nova Editora
startElement: livro
startElement: nome
Characters: Introdução ao XML
endElement: nome
startElement: editora
Characters: Nova Editora
endElement: editora
endElement: livro

JAXP

- Java APIs for XML Processing
- As principais APIs JAXP são definidas no pacote *javax.xml.parsers*
- Esse pacote contém as fabricas
  - SAXParserFactory
  - DocumentBuilderFactory
- Essas fabricas geram instancias de objetos
  - SAXParser
  - DocumentBuilder

Document Object Model

PARSER DOM

Parser DOM

1. Utilize o método estático newInstance() da classe DocumentBuilderFactory para criar um objeto DocumentBuilderFactoy
2. Utilize o método newDocumentBuilder() do objeto DocumentBuilderFactory para criar um objeto DocumentBuilder
3. Utilize um dos métodos parse() do objeto DocumentBuilder para ler o arquivo XML e criar uma árvore do tipo org.w3c.dom.Document

Parser DOM

```
try{
   DocumentBuilderFactory factory =
   DocumentBuilderFactory.newInstance();
   DocumentBuilder builder =
   factory.newDocumentBuilder();
   doc = builder.parse(“documento.xml");
}catch ...
```

Parser DOM - Validante

```
try{
   DocumentBuilderFactory factory =
   DocumentBuilderFactory.newInstance();
   factory.setValidating(true);
   DocumentBuilder builder =
   factory.newDocumentBuilder();
   Document doc = builder.parse(“documento.xml");
}catch ...
```

DOM

- O objeto **doc** é uma instancia de **Document** e representa o documento XML
- Com uma instancia de Document é possível manipular a árvore de documento DOM
- Document deriva (é estendida) da classe **Node**

Node

- Representa um nó na árvore, pode ser um:
  - Documento
  - Elemento
  - Atributo
  - Instrução de processamento
  - CDATA
  - Entidade
  - DTD
  - Texto
  - Comentario
  - Lista de nós
  - etc

Métodos do Node

Métodos	O que faz
getNodeName	Retorna o nome do nó
getNodeValue	Retorna o valor do nó
setNodeValue	Modifica o valor do nó
getNodeType	Retorna o tipo de nó (Node.ELEMENT_NODE, Node.ATTRIBUTE_NODE, Node.TEXT_NODE, etc)
getParentNode	Retorna o no pai
getChildNodes	Retonra uma lista (NodeList) de nós filhos
getAttributes	Retorna uma lista mapeada (NamedNodeMap) de atributos
insertBefore	Insere um novo nó filho antes de um nó filho especificado
appendChild	Adiciona um nó filho como último filho
removeChild	Remove um nó filho especificado

Document

- Representa o documento XML.

Métodos	O que faz
createAttribute	Cria um nó atributo
createElement	Cria um nó elemento
createTextNode	Cria um nó texto
getElementById	Retorna um elemento pelo seu identificador
getElementByTagName	Retorna uma lista de nós do tipo especificado

Element

- Representa um elemento. Estende a interface Node

Métodos	O que faz
getAttibute	Retorna o valor de um atributo
getAttributeNode	Retorna o nó atributo
setAttribute	Adiciona um atributo ou modifica o valor do atributo se ele já existir
setAttributeNode	Adiciona um novo nó atributo
removeAttribute	Remove um atributo pelo nome especificado
removeAttributeNode	Remove um atributo pelo nó atributo especificado
getElementsByTagName	Retorna uma lista de nós do tipo especificado

NodeList

- Representa uma coleção de nós

Métodos	O que faz
item	Recebe um inteiro como parâmetro e retorna um nó da lista
getLength	Retorna a quantidade de nós na lista

NamedNodeMap

- Representa uma coleção de nós que podem ser acessados pelo nome

Métodos	O que faz
Item	Recebe um inteiro como parâmetro e retorna um nó da lista
getLength	Retorna a quantidade de nós na lista
getNamedItem	Retorna um nó cujo nome foi especificado
removeNamedItem	Romove um nó cujo nome foi especificado
setNamedItem	Adiciona um novo nó.

Attr

- Representa um atributo de um Element

Métodos	O que faz
getName	Retorna o nome do atributo
getValue	Retorna o valor do atributo
setValue	Altera o valor do atributo

Attr

- Representa um atributo de um Element

Métodos	O que faz
getName	Retorna o nome do atributo
getValue	Retorna o valor do atributo
setValue	Altera o valor do atributo

Serializando

- Consiste em pegar uma árvore DOM e transforma-la em texto de forma que seja possível salvar em um arquivo ou envia-la pela rede para outra aplicação

Serializando

1. Utilize o método estático newInstance() da classe TransformerFactory do pacote javax.xml.transform.TransformerFactory que irá retornar um objeto do tipo TransformerFactory
2. Utilize o método newTransformer() do objeto criando anteriormente que irá retornar um objeto do tipo javax.xml.transform.Transformer
3. Crie um objeto do tipo javax.xml.transform.dom.DOMSource passando como parâmetro o Document DOM
4. Crie um objeto do tipo javax.xml.transform.stream.StreamResult conectado ao OutputStream que será a saída do documento
5. Chame o método transform() do objeto Transformer e passe como parâmetro os objetos de entrada (source) e saída (out)

Serializando

```
try {
        TransformerFactory tFactory =
         TransformerFactory.newInstance();
        Transformer trans = tFactory.newTransformer();
        DOMSource fonte = new DOMSource(doc);
        StreamResult resultado = new
         StreamResult(out);
        trans.transform(fonte, resultado);
      } catch ...
```

- Sendo **doc** um objeto do tipo Document
- Sendo **out** a saída padrão, um arquivo, etc

XSLT

1. Utilize o método estático newInstance() da classe TransformerFactory para criar um objeto
2. Chame o método newTransformer() do objeto criado e passe como parâmetro um objeto Source que representa o documento XSLT, isso retornará um objeto do tipo Transformer
3. Crie um objeto Source que indique o documento XML
4. Crie um objeto Result que indique a saída gerada pela transformação, por exemplo, saída padrão, um arquivo em disco, etc
5. Chame o método transform do objeto do tipo Transformer e passe como parâmetro os objetos Source e Result

XSLT

```
TransformerFactory tFactory =
   TransformerFactory.newInstance();
Transformer transformer =
   tFactory.newTransformer(new
   StreamSource("estilo.xsl"));
transformer.transform(new
   StreamSource("notas.xml"), new
   StreamResult(saida));
//  saida pode ser System.out ou “saida.txt” ou ainda do tipo PrintWriter em um Servlet
```

Simple API for XML

PARSER SAX

Parser SAX

1. Crie um objeto do tipo DefaultHandler ou que derive dele
2. Utilize o método estático newInstance() da classe SAXParserFactory para criar um objeto do tipo SAXParserFactory
3. Crie um objeto do tipo SAXParser utilizando o método newSAXParser() do objeto do tipo SAXParserFactory
4. Chame um dos métodos parse() do objeto do tipo SAXParser e passe como parâmetro o documento XML e o objeto do tipo DefaultHandler (ou derivado dele)

Parser SAX

```
try {
        meuHandler handler = new meuHandler();
        SAXParserFactory factory =
        SAXParserFactory.newInstance();
        SAXParser parser = factory.newSAXParser();
        parser.parse(“documento.xml", handler);
        return handler.toString();
      } catch
```

Parser SAX

- ContentHandler
  - Interface que define os metodos startDocument, endDocument, startElement, endElement, dentre outros
- DefaultHandler
  - É uma classe que implementa, dentre outras interfaces, a interface ContentHandler
- Para trabalhar com SAX é necessário criar uma classe que estenda a classe DefaultHandler

Parser SAX

```
public class meuHandler extends DefaultHandler{
   private String texto;
   @Override
   public void startDocument(){
      texto="<h1>Começou o documento</h1>";
   }
   @Override
   public void endDocument()
   {
      texto+="<h2>Terminou o documento</h2>";
   }
   @Override
   public void startElement(String namespace,
      String localName,String qName,Attributes
      attributes)
   {
      texto+="<p>Abriu: "+qName+"</p>";
   }
   @Override
   public void endElement(String
      namespace,String localName, String qName)
   {
      texto+="<p>Fechou: "+qName+"<br>";
   }
   @Override
   public void characters(char [] texto,int inicio,int
      tam)
   {
      this.texto+=(new
      String(texto)).substring(inicio,
      inicio+tam)+"<br>";
   }
   @Override
   public String toString()
   {
      return texto;
   }
}
```

Bibliografia

- VELOSO, Renê Rodrigues. Java e XML: Processamento de documentos XML com Java, 2ª Edição. Novatec, 2007.