Processamento XML

Carlos A. Heuser

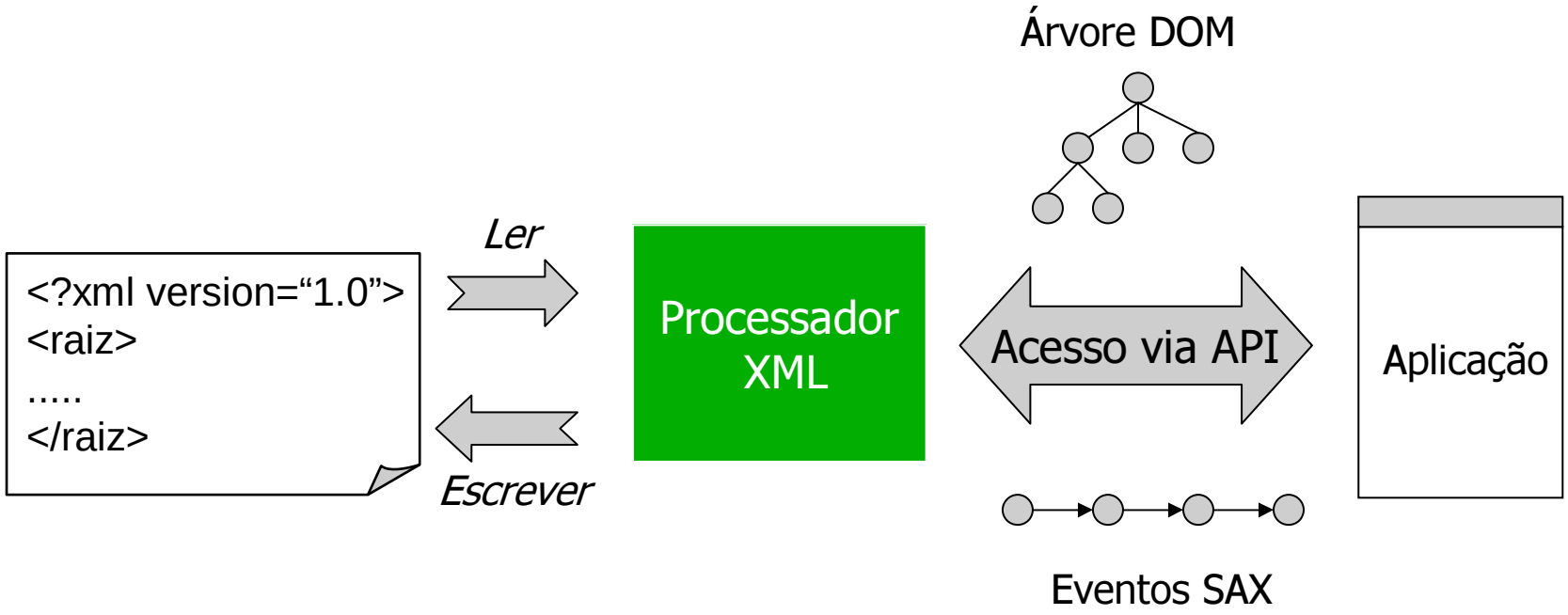
UFRGS

Processamento XML

- Aplicações necessitam **ler** e **gerar** documentos XML.
- Leitura de documentos XML pode ser muito **trabalhosa** (teste por bem-formado, tratamento de espaços em branco, tratamento de entidades, validação contra uma DTD,...)

- **Processador XML**
  - Software que reúne as função de leitura, validação e geração de documentos XML
  - Normalmente acessado pela aplicação através de uma API

Processador XML



Processador XML

- Contém dois módulos principais:
  - Manipulador de entidades
  - Reconhecedor (*parser*)

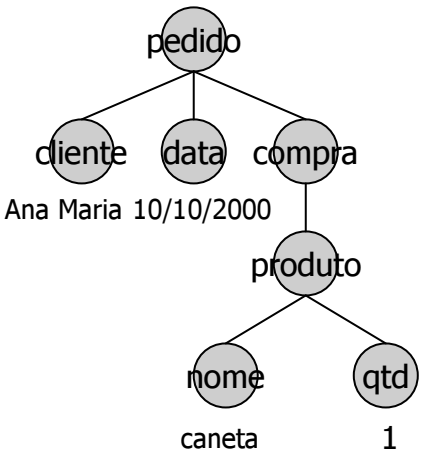
APIs e processadores

- APIs
  - DOM
  - SAX

- Processadores
  - Xerces
  - Oracle XML Processor
  - Microsoft XML Processor
  - Outros...

DOM versus SAX

Árvore DOM




```
<?xml version="1.0"?>
<pedido>
  <cliente>Ana Maria</cliente>
  <data>10/10/2000</data>
  <compra>
    <produto>
      <nome>caneta</nome>
      <qtd>2</qtd>
    </produto>
  </compra>
</pedido>
```

Eventos SAX

```
startElement pedido
startElement cliente
caracteres Ana Maria
endElement cliente
startElement data
caracteres 10/10/2000
endElement data
startElement compra
startElement produto
startElement nome
caracteres caneta
endElement nome
startElement qtd
caracteres 1
endElement qtd
endElement produto
endElement compra
endElement pedido
```

SAX - *Simple API for XML*

Carina Friedrich Dorneles

Introdução

- Projetada para trabalhar com Java
  - Outras linguagens: VB, C++, C# ...

- Desenvolvida na lista de emails XML-DEV
  - Não é orgão oficial de desenvolvimento de padrão

    mas a SAX se tornou um padrão de fato:

    http// www.megginson.com/SAX/index.htm

    ou

    http://www.saxproject.org

- Lançada em Maio/1998

SAX

- O *parser* entrega os dados em "*digestible chunks*" (pedaços digeríveis)
  - Cada evento disparado se refere a um "pedacinho" do documento
- Eventos são disparados quando as seguintes situações são encontradas:
  - *Tags* iniciais
  - *Tags* finais
  - Seções #PCDATA e CDATA
  - Instruções de processamento, comentários, declarações de entidades

Interfaces/classes do SAX

- SAX 1.0
  - `Parser` interface
    Substituída por `XMLReader` interface no SAX 2.0
  - `DocumentHandler` class
    Substituída por `ContentHandler` interface no SAX 2.0
  - `AttributeList` interface
    Substituída por `Attribute` interface no SAX 2.0
  - `ErrorHandler` interface
  - `EntityResolver` interface
  - `Locator` interface
  - `DTDHandler` interface
  - `HandlerBase` interface
    Substituída por `DefaultHandler` interface no SAX 2.0
- SAX 2.0
  - `XMLFilter` interface
  - `SAXNotSupportedException` class
  - `SAXNotRecognizedException` class

SAX – três passos

- Criar um modelo de objeto (uma classe) customizado do elemento XML desejado
- Criar um *SAX parser*
- Criar um *DocumentHandler*
  - Para tornar o documento XML uma instância do modelo de objeto customizado

SAX 2.0

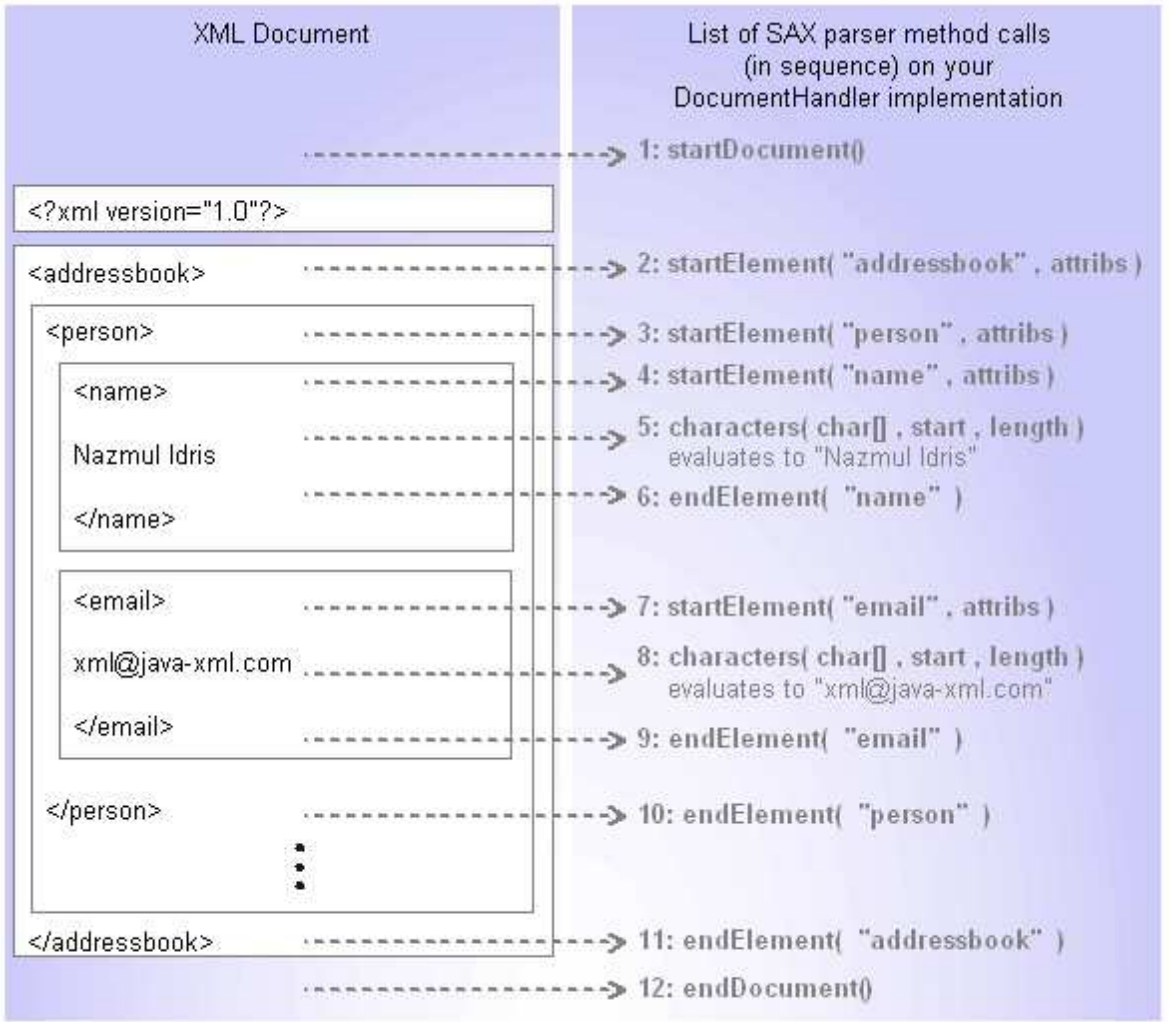


Figure 1 : SAX DocumentHandler interface methods and their sequence

Implementação Java

- Para utilizar o *parser* :

```
import org.xml.sax.*;
```

- A classes abaixo implementam a interface `Parser`
  - *Parser* sem validação

`org.apache.xerces.parsers.SAXParser`

  - *Parser* com validação

`com.ibm.xml.parser.ValidatingSAXParser`

DOM – *Document Objet Model*

Carina Friedrich Dorneles

Introdução

- Também projetada para trabalhar com Java
  - Outras linguagens: VB, C++, C# ...

- Padrão desenvolvido pela W3C

  http://www.w3.org/DOM/

DOM

- O *parser* entrega os dados em uma árvore na memória

- O *parser* trabalha em função dos nodos
- Todos os componentes de um documento XML são nodos
  - Cada "objeto" XML deve ter o seu tipo
  - Exemplos:

```
ELEMENT_NODE = 1
ATTRIBUTE_NODE = 2
TEXT_NODE = 3
DOCUMENT_NODE = 9
```

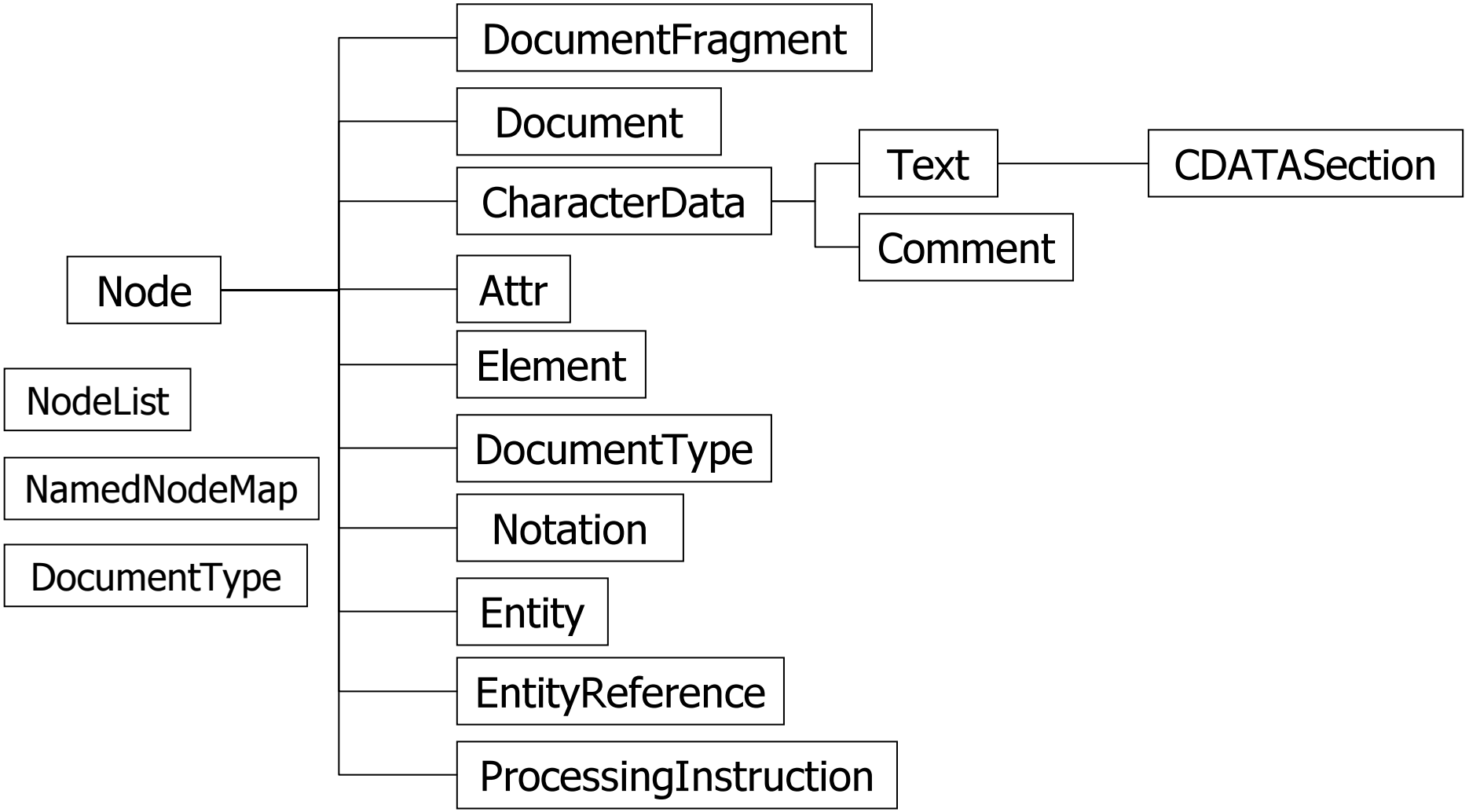
DOM - Levels

- "Basicamente":
  - DOM Level 1

    Suporte às principais características de um elemento XML
  - DOM Level 2

    Suporte a *namespaces* e extensões ao modelo de documento

Hierarquia do DOM



DOM

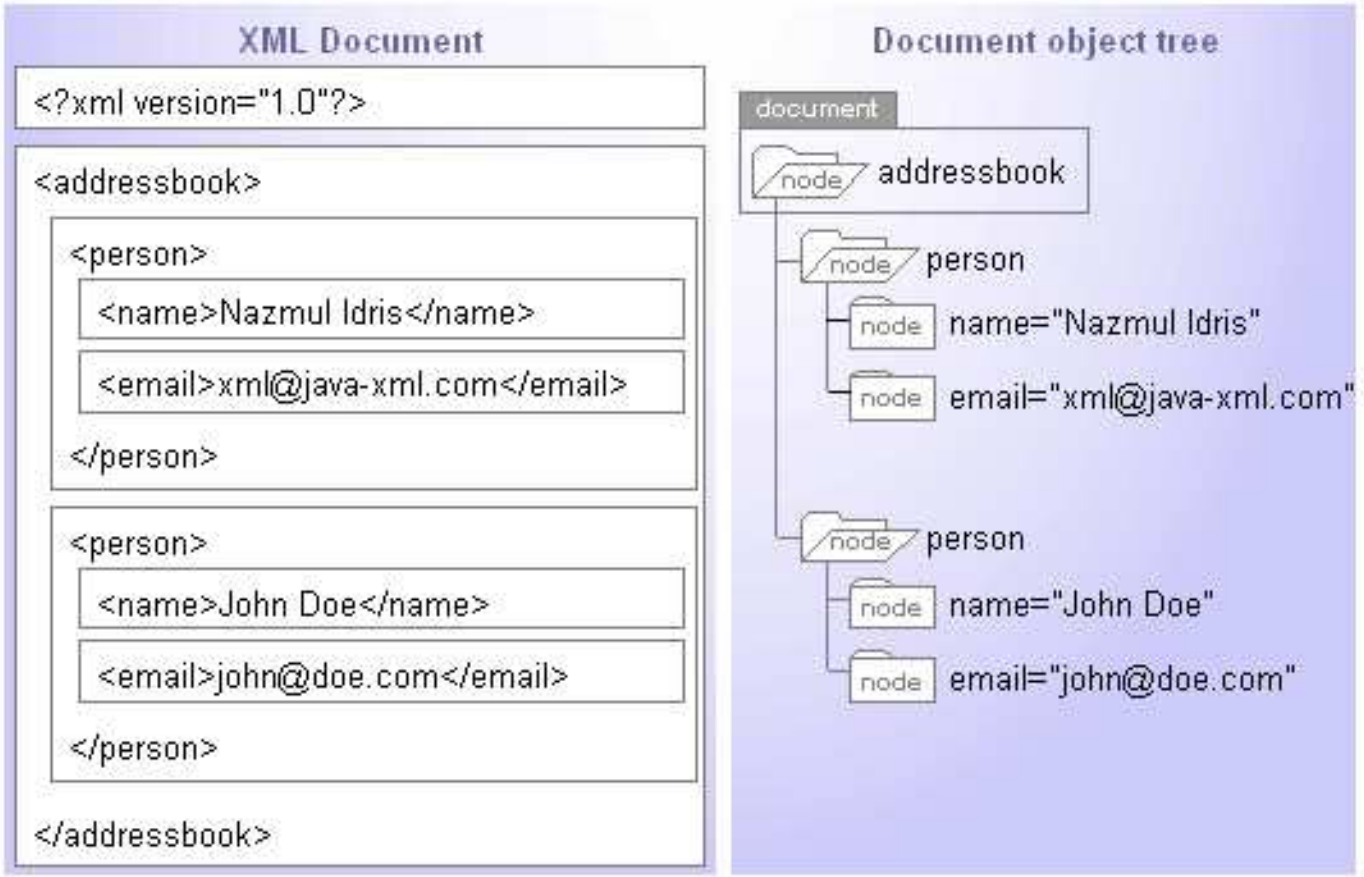


Figure 2 : Hierarchical structure of a document object

DOM

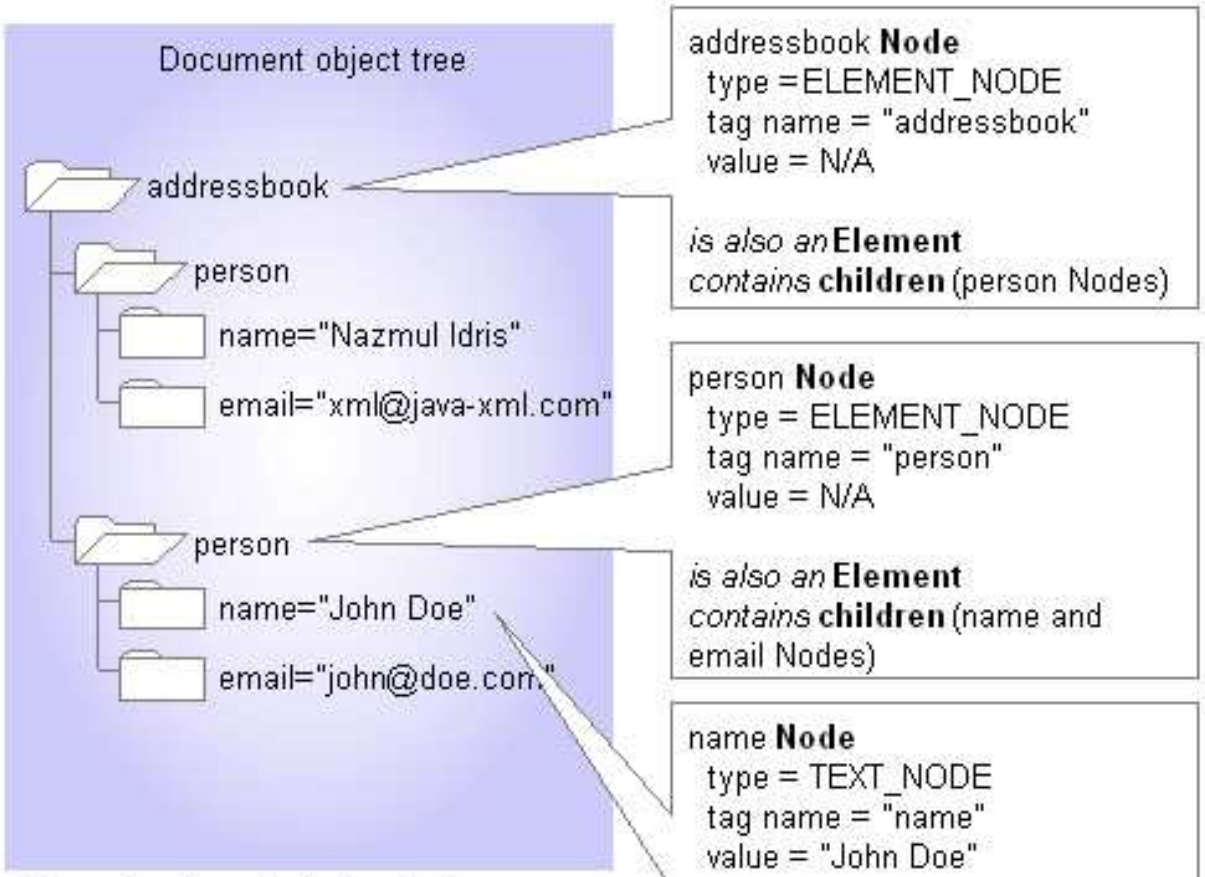


Figure 4 : A Document object tree

Eventos vs. Árvores

- Como vimos, duas formas de API para XML
- *Tree-based APIs*
  - Mapear um documento XML em uma estrutura de árvore interna
  - Fornecer subsídeos para que uma aplicação navegue nela.
  - API desenvolvida pela W3C:

    Document Object Model (DOM)

- *Event-based APIs*
  - Manda eventos para a aplicação através dos callbacks
  - Não constrói estrutura interna
  - A aplicação implementa diferentes "manipuladores" para tratar diferentes eventos
  - API:

    Simple API for XML (SAX)

Eventos vs. Árvores

- API baseada em árvore:
  - Requer muitos recursos do sistema, principalmente quando o documento é grande
    - Existe um limite no tamanho do documento (depende dos recursos de memória da máquina)
  - Muitas aplicações acabam construindo estruturas auxiliares, além da árvore
    - Torna-se ineficiente construir uma árvore apenas para mapeá-la para esta nova estrutura
- API baseada em eventos:
  - Fornece uma maneira de acesso mais simples
    - Acessar documentos de qualquer tamanho
    - Estrutura de dados usada é única. Aquela que o desenvolvedor escolher
    - A estrutura é construída, usando os manipuladores de eventos

Eventos vs. Árvores

- Situação exemplo: Localizar o texto "*Águas marítimas*" em um documento XML com 20Mb (ou até menos, 4Mb)

- Usando árvore:
  - Construir a árvore em memória e navegar por toda ela apenas para localizar uma pequena informação
- Usando evento:
  - Em um único passo, é possível encontrar a informação sem utilizar quase nada da memória

- Dica:
  - É possível fazer a análise de uma árvore usando uma API baseada em eventos