

4

XML **APIs de programação**

Helder da Rocha
(helder@argonavis.com.br)

Programação com XML

- Há duas maneiras populares para manipular (interpretar, gerar, extrair dados e tratar eventos) arquivos XML:
 - Document Object Model (**DOM**) – representação em árvore
 - Simple API for XML (**SAX**) – representação sequencial
- As duas técnicas servem a finalidades diferentes.
- **SAX** é mais simples. Oferece métodos que respondem a eventos produzidos durante a leitura do documento
 - Notifica quando um elemento abre, quando fecha, etc.
 - Permite ler dados enquanto XML carrega
- **DOM** monta uma estrutura hierárquica de objetos, em forma de árvore, que permite a navegação na estrutura do documento
 - Propriedades dos objetos podem ser manipuladas
 - Viabiliza scripting em aplicações como XHTML e SVG
 - Oferece suporte a várias APIs que usam **XPath** para extrair dados, permitindo maior produtividade e eficiência



SAX ou DOM?

■ SAX

- *Gasta menos memória: não precisa carregar o documento inteiro*
- *Ideal para aplicações simples que não precisam manipular com toda a árvore de objetos (ex: busca simples)*
- *Programação em nível mais primitivo: foco em eficiência*
- *Pode-se usar SAX para montar uma árvore DOM*
- *Não é um 'padrão'; é mais uma técnica de processamento que uma API; há várias alternativas similares: StAX (Java), SAX2, etc.*

■ DOM

- *Mais novo (linguagens mais antigas suportam apenas SAX)*
- *Baseado em objetos e mais fácil de usar (API de nível mais alto)*
- *Há uma API padrão do W3C (há outras APIs similares: JDOM, etc.)*
- *É a única opção para manipulação no cliente (via scripts)*
- *Permite validação e busca sofisticada combinado com XPath*

■ APIs disponíveis em várias linguagens

- *Java, VB, C/C++, Objective-C, C#, Python, Ruby, JavaScript (DOM)*



Bibliotecas e APIs populares

- Em C e C++ (parsers, SAX e DOM)
 - *libxml2* (projeto Gnome): <http://www.xmlsoft.org>
 - Apache *Xerces* C++: <http://xerces.apache.org/xerces-c>
- Em Java
 - Pacotes nativos: *javax.xml* (parsers), *org.w3c.dom*, *org.w3c.sax*
- Microsoft .NET
 - *XmlTextReader* (navegação na árvore), *XmlReader* (SAX)
 - *XmlDocument* (DOM), *XPathNavigator*
- PHP
 - *xml_parser_create()* (SAX, desde PHP3)
 - *new XmlDocument* (DOM, desde PHP5)
- Python
 - Pacotes nativos: *xml.sax* e *xml.dom*
- Objective-C na plataforma Apple
 - Mac OS X: *NSXML* (SAX) *NSXMLDocument* (similar a DOM)
 - iOS 4: *NSXML* (SAX) – DOM somente usando bibliotecas de terceiros



- *Document Object Model – API padrão*
 - *Padrões do W3C: DOM Level 1, DOM Level 2*
 - *Usado também por HTML*
- *Objetivo da especificação: oferecer uma **interface de programação uniforme**, independente de plataforma e linguagem, para aplicações que manipulam XML*
- *Serve para*
 - *criar um novo documento XML*
 - *navegar na árvore XML*
 - *modificar, remover ou adicionar nós (elementos, atributos, texto, comentários, PIs, etc.)*



- *Simple API for XML*
- *Técnica de processamento que dispara eventos durante processamento do documento*
 - *eventos representam componentes lidos e podem ser capturados por objetos ouvintes cadastrados*
 - *ações podem ser tomadas em cada situação*
- *Exemplos de eventos*
 - *início e fim do documento*
 - *início e fim do elemento (pode-se descobrir qual)*
 - *nó de caractere*
 - *nó de comentário*
 - ...



Como escolher entre SAX e DOM

- *Não são concorrentes – são complementares*
 - *DOM é ideal para manipular a árvore XML recursivamente e fundamental para scripting*
 - *SAX é ideal para ler o documento seqüencialmente*
 - *DOM requer carga de todo o documento antes de iniciar o processamento: consome mais memória*
 - *SAX não "lembra" de tarefas realizadas: não serve para validar referências cruzadas*
- *Em aplicações típicas, havendo suporte a DOM, use-o!*
 - *É muito mais produtivo, moderno e fácil de entender*
 - *Use SAX quando precisar de **eficiência***
 - *Use SAX quando não puder ter o XML inteiro na memória (ex: extração seletiva, busca, processos paralelos, etc.)*



SAX: exemplo do funcionamento

- Se um processador SAX receber o documento ...

```
<carta>  
  <mensagem id="1">Bom dia!</mensagem>  
</carta>
```

- ... ele irá disparar os seguintes eventos:

```
▶▶▶ startDocument()  
  ▶▶▶ startElement("carta", [])  
    ▶▶▶ startElement("mensagem", [Attribute("id","1")])  
      ▶▶▶ characters("Bom dia!")  
    ▶▶▶ endElement("mensagem")  
  ▶▶▶ endElement("carta")  
▶▶▶ endDocument()
```

- Programador deve implementar um objeto "ouvinte" para capturar os eventos e extrair as informações desejadas



Como usar SAX em Java

- *Crie classe ouvinte estendendo org.w3c.sax.DefaultHandler*
`public class MySaxHandler extends DefaultHandler {...}`
- *Implemente os métodos de evento desejados nessa classe*
- *Crie outra classe para inicializar o parser e processar o documento XML*
- *Importe as classes SAXParserFactory, SAXParser e XMLReader de org.w3c.sax*

```
SAXParserFactory spf = SAXParserFactory.newInstance();  
SAXParser sp = spf.newSAXParser();  
XMLReader reader = sp.getXMLReader();  
reader.setContentHandler(new MySaxHandler()); //  
registro de interface  
reader.parse("documento.xml");
```



Exemplo em Java: handlers de eventos

```
public void characters(char[] ch, int start, int length) {  
    for (int i = start; i < length; i++) {  
        System.out.println(ch[i]);  
    }  
}
```

```
public void startElement(String uri, String localName,  
                        String qName, Attributes att) {  
    System.out.print("<" + qName);  
    for (int i = 0; i < att.getLength(); i++) {  
        System.out.print(" " + att.getQName(i) + "="  
                        + att.getValue(i) + "'");  
    }  
    System.out.println(">");  
}
```

```
public void endElement(String uri, String localName,  
                      String qName) {  
    System.out.println("</" + qName + ">");  
}
```



Processamento XML com DOM

- *Leitura do documento (pode usar SAX)*
 - *Passo 1: carregar o documento e identificar os tokens*
 - *Passo 2: identificar elementos e outros nós*
 - *Passo 3: montar a árvore (pode ser feito durante passo 2)*
- *Alteração (com a árvore na memória)*
 - *Localiza o nó de referência*
 - *Remove, altera, muda posição, cria nó antes ou depois.*
- *Criação*
 - *Cria raiz; cria elemento raiz; cria elementos filho e atributos; cria nós de texto*
 - *Monta árvore*
 - *coloca atributos em elementos, nós de texto em elementos, elementos filho em elementos pai, elemento raiz na raiz*

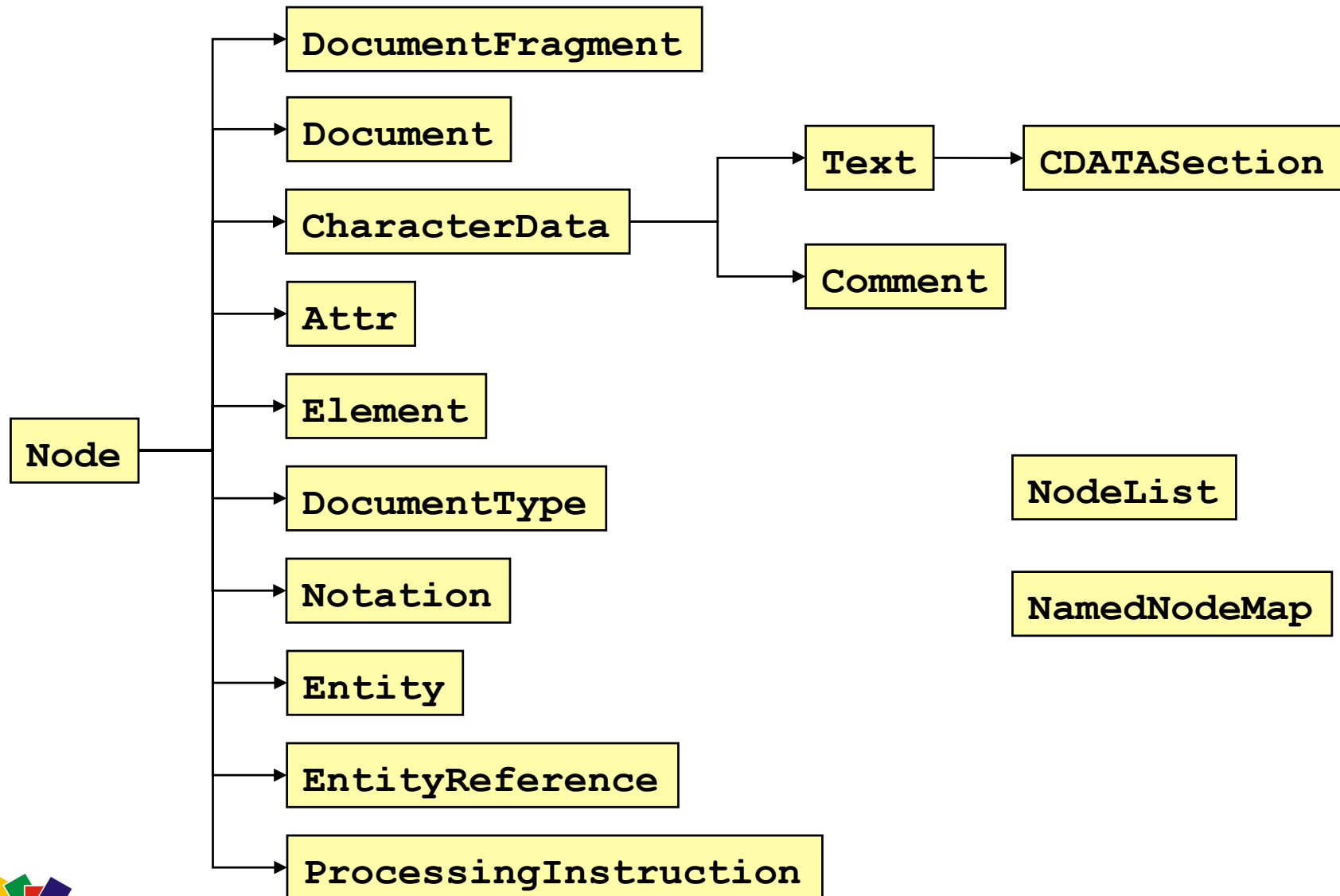


W3C DOM (padrão): tipos de dados

- Coleções: **NodeList**, **NamedNodeMap**
- Raiz da hierarquia de nós: **Node**
- Subclasses de Node
 - **Attr** (atributo)
 - **CharacterData** (classe abstrata)
 - **Text** (nó de texto)
 - **CDATASection** (seção CDATA)
 - **Comment** (comentário)
 - **Document** (documento inteiro)
 - **DocumentFragment** (sub-árvore)
 - **DocumentType** <!DOCTYPE>
 - **Element** (elemento)
 - **Entity** (valor da entidade - conteúdo)
 - **EntityReference** (nome da variável)
 - **Notation** (valor de uma notação)
 - **ProcessingInstruction** (instrução de processamento)



W3C DOM: Hierarquia



W3C DOM: tipos de nó

- *DOM usa constantes para identificar tipos de nó (nodeType)*

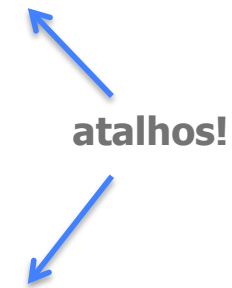
| Constante* (opcional) | Tipo | valor |
|-------------------------------|-----------------------|--------------|
| ■ ELEMENT_NODE | Element | 1 |
| ■ ATTRIBUTE_NODE | Attr | 2 |
| ■ TEXT_NODE | Text | 3 |
| ■ CDATA_SECTION_NODE | CDATASection | 4 |
| ■ ENTITY_REFERENCE_NODE | EntityReference | 5 |
| ■ ENTITY_NODE | Entity | 6 |
| ■ PROCESSING_INSTRUCTION_NODE | ProcessingInstruction | 7 |
| ■ COMMENT_NODE | Comment | 8 |
| ■ DOCUMENT_NODE | Document | 9 |
| ■ DOCUMENT_TYPE_NODE | DocumentType | 10 |
| ■ DOCUMENT_FRAGMENT_NODE | DocumentFragment | 11 |
| ■ NOTATION_NODE | Notation | 12 |



* Nomes das constantes às vezes variam, conforme a implementação

Alguns métodos da interface Node

| | | |
|----------------|---------------------------------|------------------------|
| ■ Node | appendChild(Node) | |
| ■ Node | cloneNode(boolean) | |
| ■ NamedNodeMap | getAttributes() | attributes |
| ■ NodeList | getChildNodes() | childNodes |
| ■ boolean | hasAttributes() | |
| ■ boolean | hasChildNodes() | |
| ■ Node | insertBefore(Node, Node) | |
| ■ Node | removeChild(Node) | |
| ■ Node | replaceChild(Node, Node) | |
| ■ Node | getFirstChild() | firstChild |
| ■ Node | getLastChild() | lastChild |
| ■ Node | getNextSibling() | nextSibling |
| ■ Node | getPreviousSibling() | previousSibling |
| ■ String | getNodeName() | nodeName |
| ■ short | getNodeType() | nodeType |
| ■ String | getNodeValue() | nodeValue |
| ■ Document | getOwnerDocument() | ownerDocument |
| ■ Node | getParentNode() | parentNode |



Métodos para listas e mapas

- **NamedNodeMap**
 - Node **item(int)**
 - Node **getNamedItem(String)**
 - Node **nextNode()**
 - void **reset()**
 - int **getLength()** **length**
- **NodeList**
 - Node **item(int)**
 - Node **nextNode()**
 - void **reset()**
 - int **getLength()** **length**



Interface Element

- *String* **getAttribute(String)**
- *String* **getAttributeNS(String, String)**
- *Attr* **getAttributeNode(String)**
- *Attr* **getAttributeNodeNS(String, String)**
- *NodeList* **getElementsByTagName(String)**
- *NodeList* **getElementsByTagNameNS(String, String)**
- *String* **getTagName()** **tagName**
- *boolean* **hasAttribute(String)**
- *boolean* **hasAttributeNS(String, String)**
- *void* **removeAttribute(String)**
- *void* **removeAttributeNS(String, String)**
- *void* **setAttribute(String, String)**
- *void* **setAttributeNS(String, String, String)**



Interfaces Attr e Text

■ Attr

- *String* **getName()** **name**
- *Element* **getOwnerElement()** **ownerElement**
- *String* **getValue()** **value**
- *void* **setValue(String)**

■ Text e CharacterData

- *void* **appendData(String)**
- *String* **getData()** **data**
- *int* **getLength()** **length**
- *void* **insertData(int, String)**
- *void* **replaceData(int, int, String)**
- *void* **setData(String)**



W3C DOM 2.0 com namespaces

- Use métodos que levam em conta o namespace
 - É necessário para acessar elementos e atributos que usam namespaces (ex: xlink)
 - É necessário quando se usa XML com namespaces (ex: quando usado junto com SVG, XHTML, XSL-FO, etc.)
- Em vez de `getAttribute`, `getElement`, etc.
 - Use `getAttributeNS`, `getElementNS`, etc.
 - Exemplo usando Java

```
String svgNS = "http://www.w3.org/2000/svg";  
String xlinkNS = "http://www.w3.org/1999/xlink";
```

```
Node circle = doc.createElementNS(svgNS, "circle");  
circle.setAttributeNS(null, "cx", 500);  
circle.setAttributeNS(null, "cy", 500);  
circle.setAttributeNS(xlinkNS, "href", "http://www.a/com");
```



Interface Document

- *Attr* **createAttribute(String)**
- *Attr* **createAttributeNS(String, String)**
- *Element* **createElement(String)**
- *Element* **createElementNS(String, String)**
- *Text* **createTextNode(String)**
- *DocumentType* **getDocType()** **docType**
- *Element* **getDocumentElement()** **documentElement**
- *Element* **getDocumentById(String)**
- *NodeList* **getElementsByTagName(String)**
- *NodeList* **getElementsByTagNameNS(String, String)**



Exemplo criação de árvore

- *Usando interfaces do W3C DOM padrão*

/

Document

Obter objeto do tipo **Document**
(depende de processador): **document**

<carta>

Element

```
carta := document.createElement("carta")
```

<mensagem>

Element

```
mens := document.createElement("mensagem")
```

Bom dia!

String

```
texto := document.createTextNode("Bom dia!")
```

- *Atributos*

<carta id="1">

```
carta.setAttribute("id", "1")
```



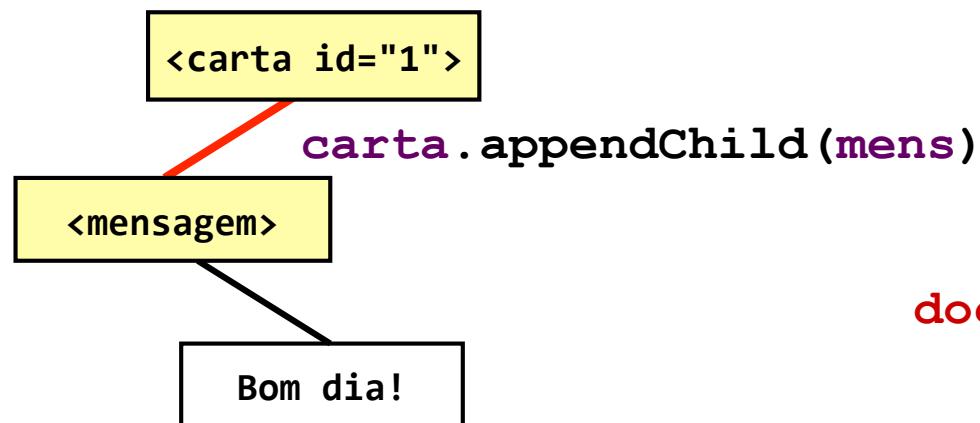
DOM: montagem da árvore

- Usando interface DOM padrão

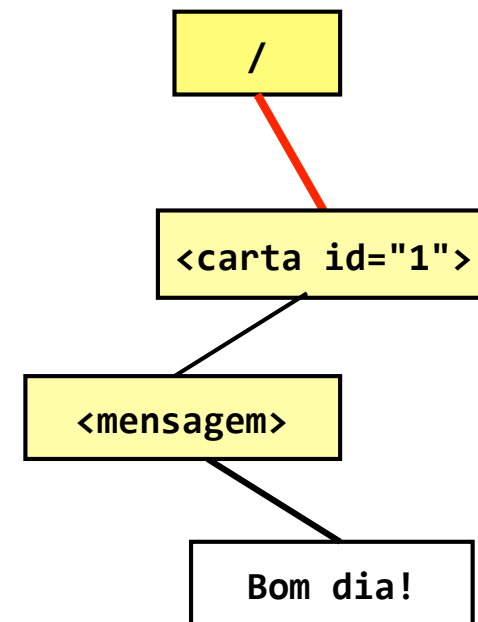
- 1. Sub-árvore <mensagem>



- 2. Sub-árvore <carta>



- 3. Árvore completa



`document.appendChild(cartas)`



Como obter o document

- Para usar DOM é preciso obter uma referência a um elemento do tipo **Document**
 - Em C#, use classes do **System.XML**
 - Em Java, inicialize um processador (pacote `javax.xml` e objeto **DocumentBuilder**) e use a API DOM em `org.w3c.dom`
 - Em PHP 5, crie um **DomDocument**
- Em aplicações XML que rodam no browser (XHTML, SVG) há um objeto pre-definido **document**
 - Em browsers HTML, o objeto **document** pode ser usado em scripts (ex: `document.getElementById('nome')`)
 - Nas implementações de visualizadores SVG o objeto default também se chama **document**



Obtenção do Document em Java

- Use os pacotes `javax.xml.parsers.*` e `org.w3c.dom.*`

- Para obter um **Document**

- *Crie um `javax.xml.parsers.DocumentBuilder`*

```
DocumentBuilderFactory factory =  
    DocumentBuilderFactory.newInstance();  
DocumentBuilder builder = factory.newDocumentBuilder();
```

- *Chame `builder.newDocument()` para obter um elemento raiz de um documento vazio (`org.w3c.dom.Document`)*

```
Document document = builder.newDocument();
```

- *Ou chame `builder.parse("documento.xml")` para obter o elemento raiz de um documento XML existente*

```
Document document = builder.parse("documento.xml");
```

- *Exemplo de uso de DOM com Java*

- `Element elemento = document.getElementById("secao");`
 - `elemento.appendChild(document.createElement("p"));`



Java: gravação em XML

- *Uma vez criada a árvore DOM, ela pode ser **serializada** para XML (arquivo de texto)*
- *Solução padrão é usar XSLT (javax.transform)*
 - *javax.xml.transform.**
 - *javax.xml.transform.dom.DOMSource;*
 - *javax.xml.transform.stream.StreamResult;*
- *O trecho abaixo imprime o documento XML contido em document na saída padrão (System.out)*

```
TransformerFactory tFactory = TransformerFactory.newInstance();
Transformer transformer = tFactory.newTransformer();

DOMSource source = new DOMSource(document);

StreamResult result = new StreamResult(System.out);
transformer.transform(source, result);
```



Exemplo de DOM com .NET (C#)

- *Use a biblioteca System.XML*

```
using System.XML;
```

- *Para criar um **Document** vazio (e obter a referência para o seu elemento raiz):*

```
XmlDocument document = new XmlDocument();
```

- *Para processar um documento existente*

```
XmlDocument document = new XmlDocument();
```

```
document.loadXml("<peessoa><nome>...</peessoa>");
```

- *Exemplo de uso de DOM com C#*

```
Element elemento = document.getElementById("secao");
```

```
elemento.appendChild(document.createElement("p"));
```

- *Para gravar em texto em arquivo*

```
document.Save("c:\\resultado.xml");
```



Exemplo de DOM em PHP 5

- Para obter o **Document** de uma árvore nova:

```
$document = new DomDocument;
```

- Para processar um documento XML existente e obter seu **Document**:

```
$document = new DomDocument;
```

```
$document->load('arquivo.xml');
```

- Exemplo de uso de DOM em PHP

```
$elemento = $document->getElementById("secao");
```

```
$elemento->appendChild($document->createElement("p"));
```

- Para gravar a árvore novamente em XML (imprimindo na saída padrão)

```
print $document->saveXML();
```



Exemplo em Objective-C (Mac OS)

- *Para criar um Document novo*

```
NSXMLElement *raiz =  
    (NSXMLElement *) [NSXMLNode elementWithName:@"pessoa"];  
NSXMLDocument *document =  
    [[NSXMLDocument alloc] initWithRootElement:raiz];
```

- *Para obter o Document de um documento existente*

```
NSURL *furl = [NSURL fileURLWithPath:@"arquivo.xml"];  
NSXMLDocument *document = [[NSXMLDocument alloc]  
    initWithContentsOfURL:furl  
    options:NSXMLNodePreserveWhitespace  
    error:&err];
```

- *Exemplo de uso (não usa interfaces DOM padrão)*

```
NSXMLElement *elemento = (NSXMLElement *) [[document  
    nodesForXPath:@"*[@id='secao']" error: &err) objectAtIndex:0];  
[elemento addChild:[NSXMLElement elementWithName:@"p"]];
```

- *Gravação de XML*

```
NSData *xmlData = [document  
    XMLDataWithOptions:NSXMLNodePrettyPrint];  
[xmlData writeToFile:@"resultado.xml" atomically:YES];
```



DOM padrão: usar o não usar

- *Em linguagens de script, que rodam no browser, use sempre DOM padrão W3C*
 - *Usar soluções proprietárias em aplicações que rodam no cliente não é recomendado*
- *Em aplicações no servidor ou standalone, escolha a solução mais adequada à sua aplicação*
 - *Use as soluções nativas se existirem*
 - *Escolha entre soluções de acordo com recursos desejados, eficiência, etc (ex: várias APIs DOM diferentes para Mac OS e iPhone)*
 - *Java oferece APIs mais fáceis de usar que são alternativas ao DOM padrão com JDOM e DOM4J*

