



# XML

## EXtensible Markup Language

Prof.: Fabio Perez Marzullo



# Tópicos

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

- XML x HTML
- A sintaxe XML
- DTD / XML Schema
- XML e JAVA



[fabio@mz-empresarial.com.br](mailto:fabio@mz-empresarial.com.br)



# XML x HTML

- HTML tem um conjunto fixo de tags e não interpreta o conteúdo.
- XML - interpreta o conteúdo do documento.
  - Usuário define suas próprias tags para criar estrutura.
  - Um documento XML não tem nenhuma instrução para apresentação.



# XML x HTML

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

**<html>**

**<head><head/>**

**<body>**

**<h1> Bibliografia </h1>**

**<p><i> Soa na Prática </i>**

**<br> Novatec, 2009**

**<p> <i> ERP </i>**

**<br> Novatec, 2010**

**</body>**

**</html>**



# XML x HTML

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

**<bibliografia>**

**<livros>**

**<livro>**

**<nome>SOA na Prática</nome>**

**<editora>Novatec</editora>**

**<ano>2009</ano>**

**</livro>**

**<livro>**

**<nome>ERP</nome>**

**<editora>Novatec</editora>**

**<ano>2010</ano>**

**</livro>**

**<livros>**

**</biblioteca>**



[fabio@mz-empresarial.com.br](mailto:fabio@mz-empresarial.com.br)



# A Sintaxe XML

- Documentos XML são compostos de elementos XML.
- Todo elemento XML é delimitado por tags.
- Elementos podem ser simples, vazios ou conter outros elementos formando uma árvore.
- Cada **nó** da árvore corresponde a uma tag e as **folhas** são tags associadas a texto.
- Só existe um **nó** raiz.



# A Sintaxe XML

- Declaração da versão da página XML.
  - `<?xml version = "1.0"?>`
- Todo documento XML tem que ter um elemento raiz.
  - `<alunos>`  
... ..  
`</alunos>`
- As tags de fechamento são obrigatórias
  - `<name> teste </name>`
- Os elementos devem ser aninhados corretamente.
- Comentários `<!-- comentario XML -->`



# A Sintaxe XML

- Maiúsculas e minúsculas são diferentes (Case sensitive).
  - `<name> Tiger </name>`
  - `<Name>Tiger <Name/>`
- Os valores dos atributos devem vir entre aspas.
  - `<picture filename="tiger.jpg"/>`
- As referências a entidades devem ser declaradas.
- Tags vazias são finalizadas por `/>`
  - `<book type="tech book" type="hard cover"/>`
- Atributos não podem ser repetidos na mesma tag
  - `<book type="tech book" type="hard cover">... </book>`





- Namespaces

- Permite a convivência de vários vocabulários, distinguindo elementos com nomes semelhantes;
- É um ***super-rótulo*** que distingue um elemento que tem mesmo nome em documentos diferentes.
- Declarando um namespace :

```
<livraria xmlns:book='http://ecommerce.org/product'>  
  <livro>  
    <titulo> Desvendando XML </titulo>  
  </livro>  
</livraria>
```

*O prefixo “book” está vinculado a URL  
<http://ecommerce.org/product> para o elemento “livraria” e  
conteúdos*



# A Sintaxe XML

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

- Validação
  - Existem para manter a consistência dos dados.
  - As duas propostas mais importantes são:
    - XML DTD e XML SCHEMA ambas do W3C.



# A Sintaxe XML

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

- Definição do Tipo de Documento (DTD)
  - Um DTD contém as regras que descrevem as estruturas de um tipo de documento.
  - Especifica elementos, a estrutura desses elementos, e valores possíveis para seus atributos.
  - Existem diversos blocos de construção em um DTD.
    - Elementos, atributos, entidades, #CDATA, #PCDATA



# A Sintaxe XML

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

- DTDs
  - Um DTD pode ser interno ou externo.
  - Para declarar um DTD de um arquivo XML, devemos colocar o header:
    - `<!DOCTYPE bibliografia SYSTEM "bibliografia.dtd">`
      - Identificação Externa



# A Sintaxe XML

- DTDs internas : incluídas no próprio documento XML

- `<?xml version = "1.0"?> <?xml version = "1.0"?>`

```
<!DOCTYPE bibliografia [  
  <!ELEMENT bibliografia (livro*)>  
  <!ELEMENT livro(titulo,autor+,ano,editora)>  
    <!ELEMENT titulo(#PCDATA)>  
    <!ELEMENT autor(#PCDATA)>  
    <!ELEMENT ano(#PCDATA)>  
    <!ELEMENT editora(#PCDATA)>
```

```
  ]>
```

- `<bibliografia>`  
 `<livro>`  
 `<titulo> SOA na Prática </titulo>`  
 `<autor> Fabio Marzullo </autor>`  
 `<ano> 2009 </ano>`  
 `<editora> Novatec </editora>`  
 `</livro>`  
`</bibliografia>`



# A Sintaxe XML

- Elementos
  - Para declarar um elemento dentro de um DTD, fazemos da seguinte forma:
    - <!ELEMENT nome-do-elemento (conteúdo do-elemento)>
  - Como descrever um elemento
    - Elemento vazio:
      - <!ELEMENT nome-do-elemento (EMPTY)>
    - O elemento pode conter qualquer coisa em seu interior:
      - <!ELEMENT nome-do-elemento (ANY)>
    - Há caracteres expansíveis no interior da tag:
      - <!ELEMENT nome-do-elemento (#PCDATA)>
    - Especifica que as informações no interior são texto:
      - <!ELEMENT nome-do-elemento (#CDATA)>



# A Sintaxe XML

- Atributos
  - provêm informações adicionais sobre elementos.
  - Atributos são inseridos dentro da tag de início dos elementos.
  - Atributos são representados pelo par nome/valor.
  - A sintaxe para declarar um atributo em uma DTD é a seguinte:
  - *<!ATTLIST nome-do-elemento nome-do-atributo tipo-do-atributo valor-default >*
  - Os valores default de um atributo são os seguintes:

Valor	Explicação
#DEFAULT <b>valor</b>	O atributo tem um valor default. A tag Default pode ser omitida.
#REQUIRED	O atributo tem que ser colocado obrigatoriamente.
#IMPLIED	O atributo pode ou não ser colocado.
#FIXED <b>valor</b>	O atributo tem um valor fixo determinado por <b>valor</b> .



# A Sintaxe XML

- Exemplo de um DTD que descreve uma edição de um jornal.
  - <!DOCTYPE JORNAL [
    - <!ELEMENT JORNAL (ARTIGO+)>
    - <!ELEMENT ARTIGO (MANCHETE, NOTÍCIA, NOTAS)>
    - <!ELEMENT MANCHETE(#PCDATA)>
    - <!ELEMENT NOTÍCIA(#PCDATA)>
    - <!ELEMENT NOTAS(#PCDATA)>
      - <ATTLIST ARTIGO AUTOR CDATA #REQUIRED>
      - <ATTLIST ARTIGO EDITOR CDATA #IMPLIED>
      - <ATTLIST ARTIGO DATA CDATA #IMPLIED>
      - <ATTLIST ARTIGO EDIÇÃO CDATA #IMPLIED>
  - ]>
  - Os atributos AUTOR, EDITOR, DATA e EDIÇÃO são todos do elemento ARTIGO
  - O atributo AUTOR é o único obrigatório.





# A Sintaxe XML

- PCDATA (Parsed Character Data)
  - São dados interpretados. Eles são, por exemplo, o texto que aparece entre as tags.
  - Se houver alguma tag no interior de um PCDATA, ela será interpretado.
  - Exemplo:
    - `<!ELEMENT nome (#PCDATA, sobrenome)>`
      - `<nome>Roberto`  
`<sobrenome>Nascimento</sobrenome>`  
`</nome>`



# A Sintaxe XML

- CDATA (Character Data)
  - Representa informação estática.
  - Se houverem tags dentro de um bloco CDATA, ela não será interpretado.
  - Exemplo:
    - `<!ATTLIST nome sobrenome CDATA #REQUIRED>`
    - `<nome sobrenome="Nascimento">Rodrigo</nome>`



# A Sintaxe XML

- Entidades

- São blocos de texto pré-definido.
- Um exemplo conhecido de entidade é o &nbsp; do HTML, que serve para inserir espaços em branco no texto
- As entidades são expandidas quando um documento é interpretado por um XML parser.
- As entidades podem ser divididas em dois grupos:
  - Gerais ou Internas (Atalho de texto)
    - `<!ENTITY nome-da-entidade "valor-da-entidade">`
  - Parâmetro ou externa (declaradas na DTD).
  - `<!ENTITY nome-da-entidade SYSTEM "nome-documento">`



# A Sintaxe XML

- Exemplo

- `<?xml version="1.0" ?> ?>`
- `<!DOCTYPE universidade [  
    <!ENTITY uUFRJ "Universidade Federal do Rio de Janeiro">  
    <!ENTITY uPUC "Pontificia Universidade Catolica">  
]>`
- `<universidade>`
  - `<primeira>&uUFRJ;</primeira>`
  - `<segunda>&uPUC;</segunda>`
- `</universidade>`



# A Sintaxe XML

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

- **XML SCHEMA**

- Desenvolvida pelo W3C.
- Escrita em XML .
- Contém um sistema de datatypes (tipos de dados).
- Permite definir elementos globais ou locais.
- Permite mais controle sobre o documento XML.
- Exemplo:
  - `<xsd:element name= “peso” type=“xsd:string”/>`
  - `<xsd:element name=“populacao” type=“xsd:integer”/>`



# XML e Java

Universidade Federal  
do Rio de Janeiro  
Escola Politécnica

- Principais APIS
- DOM (DOCUMENT OBJECT MODEL)
  - Transforma um documento XML em um estrutura em árvore na memória
    - Uso extensivo da memória.
- SAX (SIMPLE API FOR XML)
  - Identifica eventos ao efetuar a análise do documento (parsing) e notifica a aplicação, como por exemplo para informar que um “tag foi aberto sem o “/tag” correspondente.
- JAXP (JAVA API FOR XML PARSING)
  - API da SUN com suporte a XSLT, SAX 2.0 , DOM e XML Schema.
- JDOM
  - É uma API Open Source para leitura, escrita e manipulação de XML no JAVA





## ● O pacote org.jdom

- As classes abaixo representam um documento XML:
- Attribute
- CDATA
- Comment
- DocType
- Document
- Element
- EntityRef
- Namespace e
- ProcessingInstruction
- (PartialList)
- (Verifier)
- (Assorted Exceptions)



# XML e Java

- O pacote org.jdom
- A classe *Document*
  - Documentos são representados pela classe org.jdom.Document

Exemplo:

```
Document doc =  
    new Document( new Element("rootElement") );
```





- O pacote `org.jdom.input`
- Classes para leitura para leitura XML:
  - DOMBuilder: `org.jdom.input.DOMBuilder` é útil para leitura de uma árvore DOM.
  - SAXBuilder : `org.jdom.input.SAXBuilder` é mais rápida e recomendada.
- O documento pode ser construído de um arquivo, stream ou URL:
- Exemplo:  
`SAXBuilder builder = new SAXBuilder();`  
`Document doc = builder.build(url);`



- Leitura do arquivo exemplo.xml
  - ```
public static Document readDocument() {  
    try {  
        SAXBuilder builder = new SAXBuilder();  
        Document anotherDocument = builder.build(  
            new File("xml//exemplo.xml"));  
        return anotherDocument;  
    } catch (JDOMException e) {  
        e.printStackTrace();  
    } catch (NullPointerException e) {  
        e.printStackTrace();  
    }  
    return null;  
}
```



- O pacote `org.jdom.output`

- Classes para escrita XML em várias formas de saída:
- `DOMOutputter` `DOMOutputter` - `org.jdom.output.DOMOutputter` cria um documento DOM.
- `SAXOutputter` `SAXOutputter` - `org.jdom.output.SAXOutputter` gera eventos SAX.
- `XMLOutputter` `XMLOutputter` - `org.jdom.output.XMLOutputter` escreve os documentos como XML.
- Exemplo:  

```
XMLOutputter outputter = new XMLOutputter();  
outputter.output(doc, System.out);
```



# XML e Java

- O pacote org.jdom.output
- Criando o arquivo exemplo.xml
  - `// Criando o elemento root`  
`Element carElement = new Element("carro");`  
  
`//criando o documento`  
`Document myDocument = new Document(carElement);`  
  
`//adicionando um atributo no elemento root`  
`carElement.addAttribute (new Attribute("combustivel", "gasolina"));`  
  
`//adicionando um comentario //adicionando um comentario`  
`carElement.addContent(new Comment("Descrição de um Carro"));`



# XML e Java

- O pacote org.jdom.output
  - // Adicionado o elemento pneus  
Element pneus = new Element("pneus");  
pneus.addContent("P6000");  
carElement.addContent (pneus);  
  
//adicionado mais elementos  
carElement.addContent (new  
Element("modelo").addContent("Corsa"));  
  
...



- O pacote org.jdom.output
  - Resultado do código anterior (exemplo.xml)
  - `<?xml version="1.0" encoding="UTF-8?>`
  - `<carro combustivel="gasolina">`
    - `<!-- Descrição do Carro-->`
    - `<pneus>P6000</pneus>`
    - `<modelo>Corsa</modelo>`
    - `<ano>1997</ano>`
    - `<cor>verde</cor>`
  - `</carro>`

# Dúvidas?

