Desenvolvimento de Software para Persistência

XML: Extensible Markup Language

Prof. Regis Pires Magalhães
regismagalhaes@ufc.br

UFC

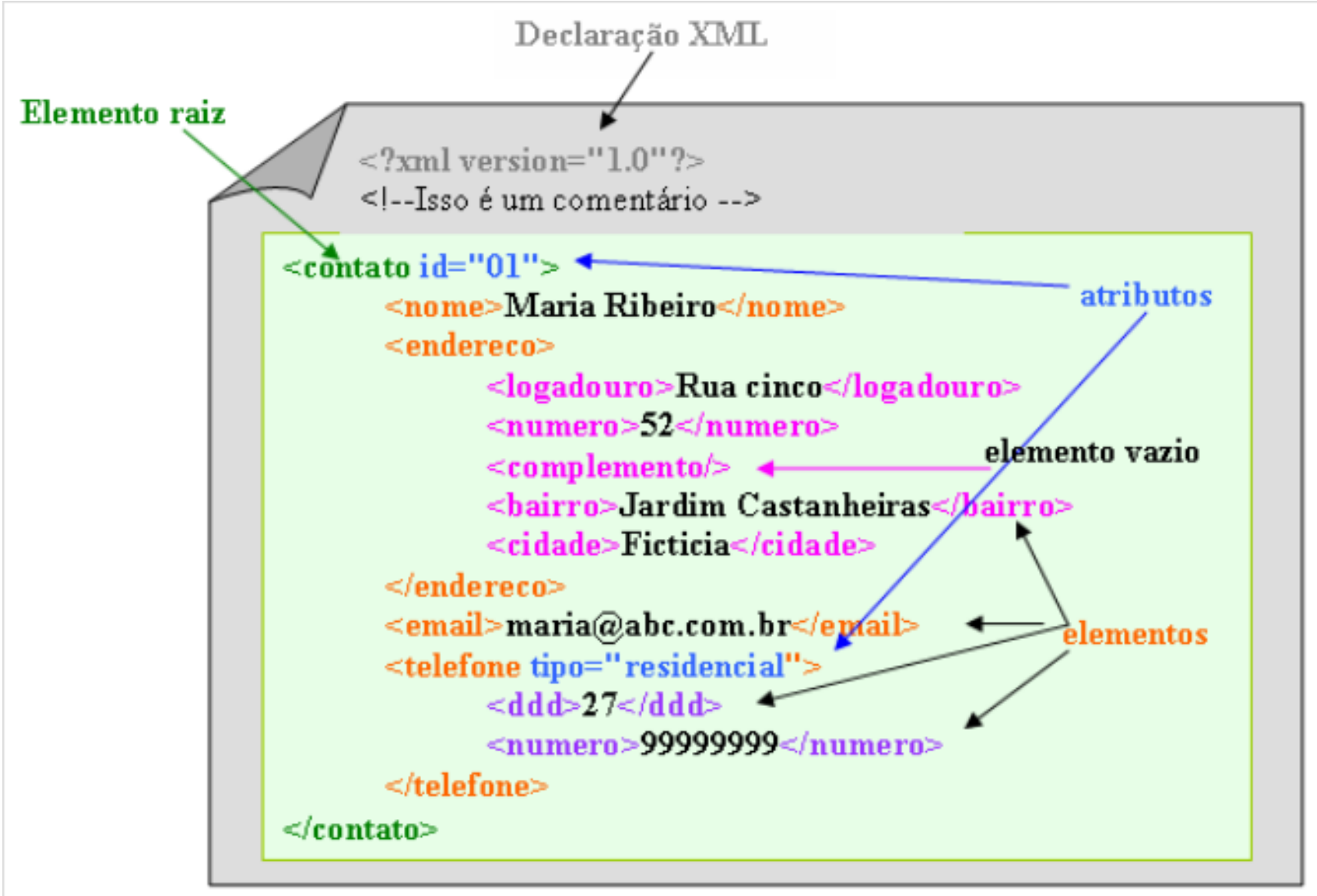
Motivação

- À medida que o comércio eletrônico e outras aplicações da Internet se tornam cada vez mais automatizadas, torna-se essencial a capacidade de trocar documentos Web entre diversos sites de computador e interpretar seu conteúdo de maneira automática.
- Essa necessidade foi um dos motivos que levaram ao desenvolvimento da XML.

XML - Extensible Markup Language

- Surgiu como padrão para estruturação e troca de dados pela Web.

Um Exemplo


Declaração XML
Elemento raiz
<?xml version="1.0"?>
<!--Isso é um comentário -->
<contato id="01">
<nome>Maria Ribeiro</nome>
<endereco>
<logadouro>Rua cinco</logadouro>
<numero>52</numero>
<complemento/>
<bairro>Jardim Castanheiras</bairro>
<cidade>Ficticia</cidade>
</endereco>
<email>maria@abc.com.br</email>
<telefone tipo="residencial">
<ddd>27</ddd>
<numero>99999999</numero>
</telefone>
</contato>
atributos
elemento vazio
elementos

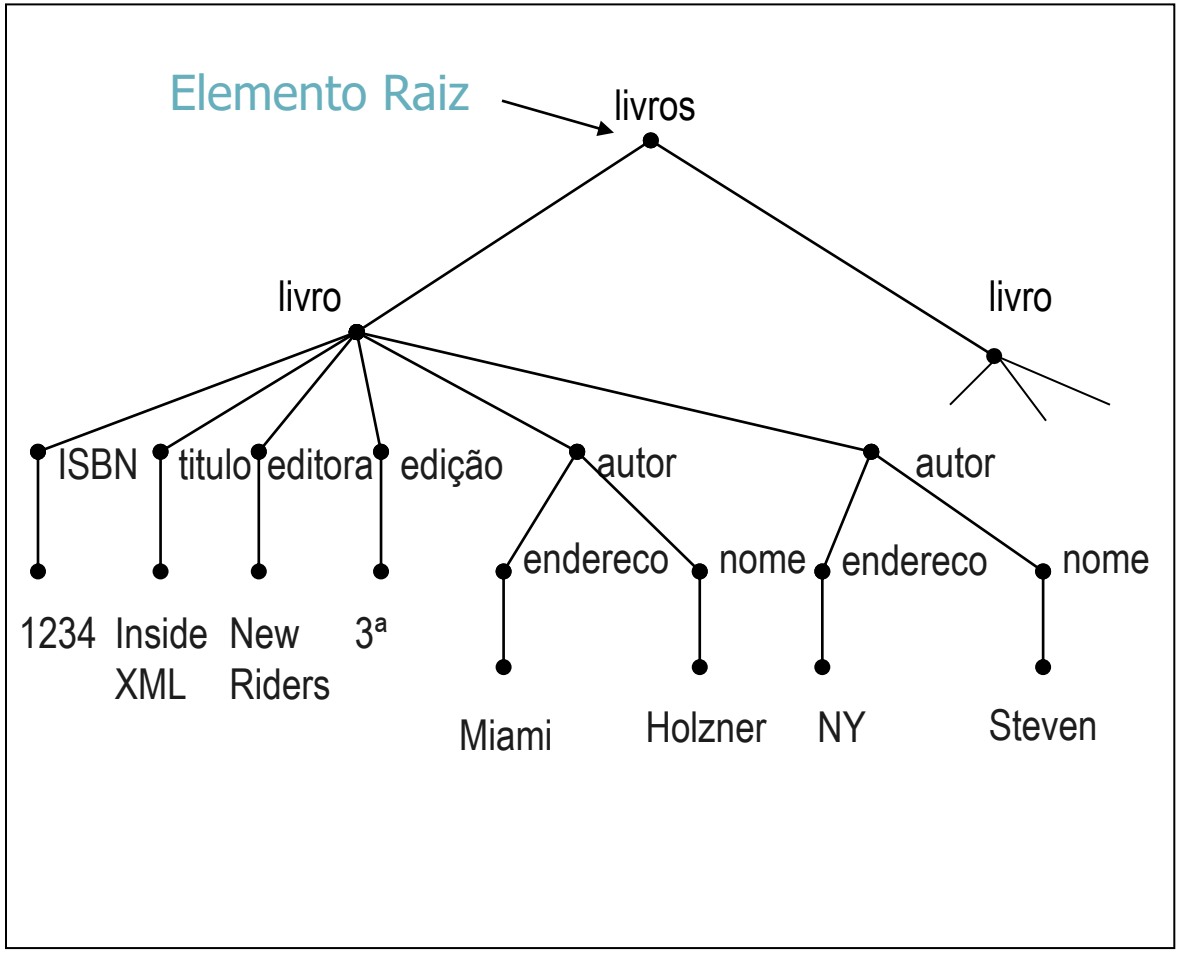
O que é XML?

- XML (e**X**tensible **M**arkup **L**anguage)
- Lingagem de marcação proposta pelo W3C
  - W3C (World Wide Web Consortium) – Orgão responsável pela recomendação de padrões e protocolos para a web.
- Padrão para representação e troca de dados na Web.
- Descreve os dados, dando semântica a unidades de informação
- Soluciona as limitações de HTML

O que é XML?

Exemplo Documento XML

```
<?xml version="1.0" >
<livros>
 <livro>
  <ISBN>1234</ISBN>
  <titulo>Inside XML</titulo>
  <editora>New Riders</editora>
  <edição>3ª</edição>
  <autor>
   <nome>Steven</nome>
   <endereco>NY</endereco>
  </autor>
  <autor>
   <nome>Holzner</nome>
   <endereco>Miami</endereco>
  </autor>
 </livro>
 <livro>  ...  </livro>
</livros>
```



O que é XML?

- Linguagem de Marcação-Descreve o conteúdo de um documento através de marcas .

Documento sem Marcas

Horário Aula XML
Wed, Jan 29, 2003 4:18 PM
vvidal@lia.ufc.br
eti-l@lia.ufc.b

A aula começará as 19:45

Documento com Marcas

```
<e-mail>
  <head>
    <subject> Horário Aula XML </subject>
    <data> Wed, Jan 29, 2003 4:18 PM</data>
    <from> vvidal@lia.ufc.br </from>
    <to> eti-l@lia.ufc.br </to>
  </head>
  <body> A aula começará as 19:45 <body>
<e-mail>
```

O que é XML?

- XML não é apenas um outra linguagem de marcadores
- A maioria das linguagens provê um conjunto fixo de marcadores. XML é extensível.

```
<livro>
      <titulo>Inside XML</titulo>
      <autor>Steven Holzner</autor>
      <preco>R$ 150,00 </preco>
 </livro>
```

Modelo de dados hierárquico da XML

- O objeto básico em XML é o documento XML.
- Conceitos de estruturação principais:
  - Elementos
    - Os elementos são identificados em um documento por sua tag de início e tag de fim.
    - Os nomes de tag são delimitados por sinais < ... >.
    - As tags de fim são identificadas ainda por uma barra, </ ... >.
  - Atributos.
    - Os atributos em XML oferecem informações adicionais que descrevem elementos.
  - Existem conceitos adicionais na XML, como entidades, identificadores e referências.

Modelo de dados hierárquico da XML

```
<?xml version= "1.0" standalone="yes"?>
   <Projetos>
      <Projeto>
         <Nome>ProdutoX</Nome>
         <Numero>1</Numero>
         <Localizacao>Santo_Andre</Localizacao>
         <Dept_nr>5</Dept_nr>
         <Trabalhador>
               <Cpf>12345678966</Cpf>
               <Ultimo_nome>Silva</Ultimo_nome>
               <Horas>32,5</Horas>
         </Trabalhador>
         <Trabalhador>
               <Cpf>45345345376</Cpf>
               <Primeiro_nome>Joice</Primeiro_nome>
               <Horas>20,0</Horas>
         </Trabalhador>
      </Projeto>
      <Projeto>
         <Nome>ProdutoY</Nome>
         <Numero>2</Numero>
         <Localizacao>Itu</Localizacao>
         <Dept_nr>5</Dept_nr>
         <Trabalhador>
               <Cpf>12345678966</Cpf>
               <Horas>7,5</Horas>
         </Trabalhador>
         <Trabalhador>
               <Cpf>45345345376</Cpf>
               <Horas>20,0</Horas>
         </Trabalhador>
         <Trabalhador>
               <Cpf>33344555587</Cpf>
               <Horas>10,0</Horas>
         </Trabalhador>
      </Projeto>
   ...
   </Projetos>
```

Elementos simples e complexos

- Elementos simples
  - contêm valores de dados.
- Elementos complexos
  - construídos hierarquicamente com base em outros elementos.

Diferenças e Similaridades

HTML:

- É uma linguagem de marcação
- Tags limitados e predefinidos
- Apresenta os dados

XML:

- É uma linguagem de marcação
- Tags definidos a formação, mas não limitados ➔ expandida
- Descreve os dados
- Os dados são auto descritos.
- É uma recomendação W3C

XML: Onde encontrar?

- Intercâmbio de dados: NF-e
- Configurações do eclipse
- Na arquitetura Orientada a Serviços – SOAP, nos Web Service.

Representação de documentos XML

- É possível representar um documento XML usando:
  - Representação textual
  - Estrutura de árvore.
    - Nós internos representam elementos complexos, enquanto os nós de folha representam elementos simples.
    - É por isso que o modelo XML é conhecido como um **modelo de árvore** ou um **modelo hierárquico**.
    - Em geral, não existe limite sobre os níveis de aninhamento dos elementos.

Documentos com ou sem esquema

- Documentos XML que não seguem um esquema predefinido de nomes de elemento e estrutura de árvore correspondente são conhecidos como documentos XML sem esquema.

- Um documento XML pode obedecer a um esquema XML predefinido ou DTD.

Documentos XML bem formados e válidos e XML DTD

- Um documento XML é **bem formado** se seguir algumas condições.
  - Precisa começar com uma **declaração XML** para indicar a versão da linguagem que está sendo usada, bem como quaisquer atributos relevantes.

```
<?xml version= "1.0" standalone="yes"?>
```

  - Seguir as diretrizes sintáticas do modelo de dados de árvore.
    - Deve haver um único elemento raiz.
    - Cada elemento precisa incluir um par correspondente de tags de início e de fim dentro das tags de início e fim do elemento pai.

Documentos XML bem formados e válidos e XML DTD

- Um documento XML bem formado é sintaticamente correto.
- Isso permite que ele seja processado por processadores genéricos, que percorrem o documento e criam uma representação de árvore interna.

Documento bem formado sem esquema

- Um documento XML bem formado pode não ter esquema, ou seja, ele pode ter quaisquer nomes de tag para os elementos do documento.

Documento válido

- O documento deverá ser bem formado e seguir um esquema específico.
  - Os nomes de elemento usados nos pares de tag de início e de fim devem seguir a estrutura especificada em um arquivo XML DTD (Document Type Definition) separado, ou arquivo de esquema XML (XML Schema).

Documento XML - Atributos

- Um elemento pode conter informação adicional sobre seu conteúdo armazenados em atributos.
  - Cada atributo é um par (*nome*, *valor*)
  - Valores dos atributos devem estar entre aspas

```
<livro isbn = "85.241.0591-9" >
    <titulo>Inside XML</titulo>
    <autor>Steven Holzner</autor>
    <preco>R$ 150,00 </preco>
</livro>
```

Estrutura de um documento XML

- Regras
  - Cada elemento possui um único *pai* (Exceção: O elemento raiz)
  - Cada elemento possui um número arbitrário de *irmãos* e *filhos*
    - Um elemento sem filho é denominado *folha*
  - Todo documento XML deve possuir uma raiz
  - Todas as tags devem ser fechadas
  - As tags devem estar bem aninhadas
  - As tags XML são “case sensitive”
  - Um elemento pode ter conteúdo vazio
    - <fone/>
    - <nada> </nada>

Erros Comuns

- Esquecer tags finais

```
<livraria>
   <livro isbn="1" editora="Addison Wesley">
         <titulo>Inside XML</titulo>
         <autor>Steven Holzner</autor>
</livraria>
```

Erros Comuns

- Esquecer que XML diferencia maiúsculas de minúsculas (*case sensitive*)

```
<livraria>
  <livro isbn="1" editora="Addison Wesley">
      <titulo>Inside XML</titulo>
      <autor>Steven Holzner</autor>
  </Livro>
</livraria>
```

Erros Comuns

- Incluir espaços no nome do elemento

```
<catálogo de livros>
  <livro isbn="1" editora="Addison Wesley">
      <titulo>Inside XML</titulo>
      <autor>Steven Holzner</autor>
  </livro>
</catálogo de livros>
```

Erros Comuns

- Esquecer as aspas no valor do atributo

```
<livraria>
  <livro isbn="1" editora=Addison Wesley>
      <titulo>Inside XML</titulo>
      <autor>Steven Holzner</autor>
  </livro>
</livraria>
```

Erros Comuns

- Fechar um subelemento após o fechamento de um elemento que o contém.

```
<livraria>
<livro isbn="1" editora="Addison Wesley">
    <titulo>Inside XML</titulo>
    <autor>Steven Holzner</autor>
</livraria>
</livro>
```

Erros Comuns

- Esquecer que deve existir apenas um elemento raiz

```
<livro isbn="1" editora="Addison Wesley">
   <titulo>Inside XML</titulo>
   <autor>Steven Holzner</autor>
</livro>
<livro isbn="2" editora="Addison Wesley">
   <titulo>Fundamental..</titulo>
   <autor>Navathe</autor>
</livro>
```

Namespace

- **Uma *namespace XML é uma coleção de* nomes identificada por uma URI**
  - Coleção de nomes = vocabulário de marcação

- **Servem para identificar a qual vocabulário de marcação a que um determinado elemento pertence.**

- **Evitam duplicação de nomes de elementos e atributos.**

Namespace

- **Utiliza uma URI para identificar cada *namespace***
  - É somente uma *string de caracteres*.

```
<meudoc xmlns:h ="http://www.w3.org/TR/html4"
        xmlns:f = "http://www.meudominio.com/moveis">
    <h:table>
        <h:tr>
        <h:td>Maçã</h:td>
                <h:td>Bananas</h:td>
        </h:tr>
    </h:table>
    <f:table>
        <f:name>Mesa de café</f:name>
        <f:width>80</f:width>
        <f:length>120</f:length>
    </f:table>
</meudoc>
```

Usando Namespace

- **São definidas através do atributo *"xmlns:" o qual declara o namespace e* o prefixo para elementos e atributos.**

```
...
<X:html xmlns:X="http://www.w3.org/TR/REC-html40">
    ...
    <X:p> Um paragrafo HTML.</X:p>
    ...
</X:html>
...
```

Usando Namespace

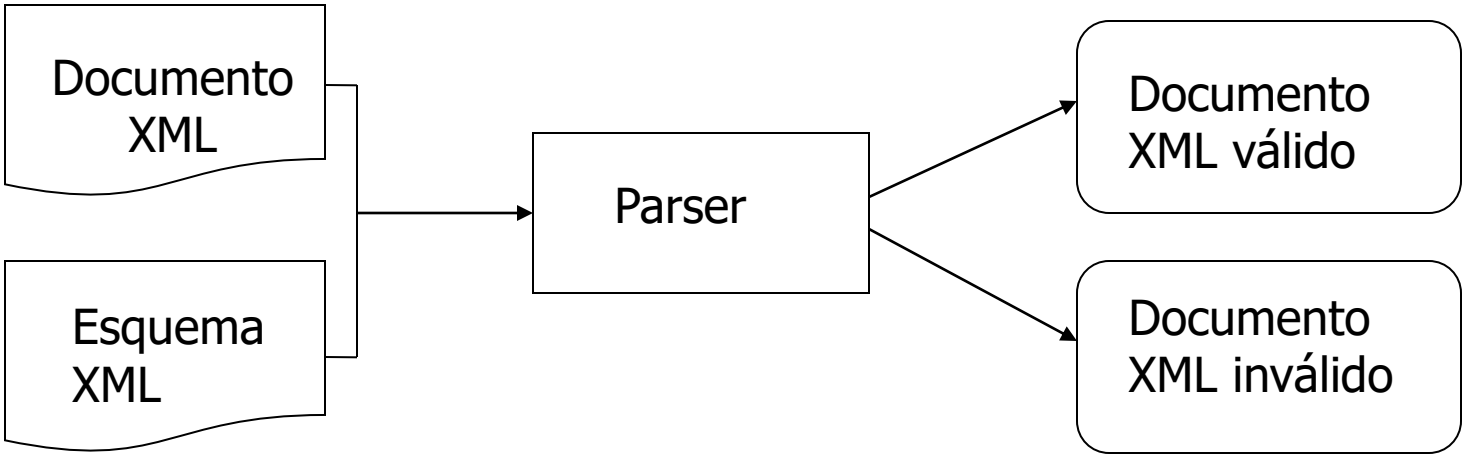
- **Um elemento pode declarar mais de um *namespace*.**

```
<Q:livro xmlns:Q="file:/DTDs/livro.dtd"
         xmlns:X="http://www.w3.org/TR/REC-html40">
   ...
   <Q:p>Um paragrafo do livro</Q:p>
   ...
   <X:p>Um paragrafo de código HTML</X:td>
   ...
</Q:book>
```

Validando Documentos XML

- Como em banco de dados, XML pode ter uma espécie de “esquema” o qual consiste de um conjunto de regras que definem a estrutura do documento.

- Um documento *é válido* com relação a um dado esquema XML se obdece este esquema



Validando Documentos XML

- Linguagens de Esquemas XML
  - DTD – Document Type Definition
  - XML Schema

Exemplo de documento DTD

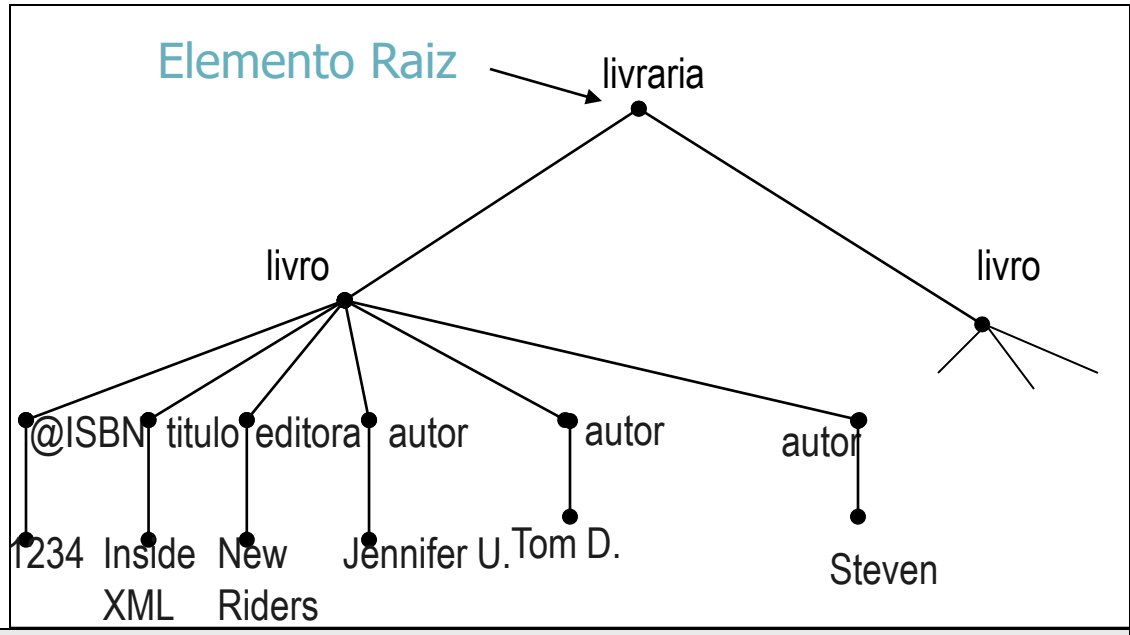
```
<!DOCTYPE Projects [
        <!ELEMENT Projetos (Projeto+)>
        <!ELEMENT Projeto (Nome, Numero, Localizacao, Dept_nr?, Trabalhadores)
            <!ATTLIST Projeto
                ProjId ID #REQUIRED>
            >
            <!ELEMENT Nome (#PCDATA)>
        <!ELEMENT Numero (#PCDATA)
        <!ELEMENT Localizacao (#PCDATA)>
        <!ELEMENT Dept_nr (#PCDATA)>
        <!ELEMENT Trabalhadores (Trabalhador*)>
        <!ELEMENT Trabalhador (Cpf, Ultimo_nome?, Primeiro_nome?, Horas)>
        <!ELEMENT Cpf (#PCDATA)>
        <!ELEMENT Ultimo_nome (#PCDATA)>
        <!ELEMENT Primeiro_nome (#PCDATA)>
        <!ELEMENT Horas (#PCDATA)>
    ]>
```

Documento DTD

- Ao especificar elementos, a notação a seguir é usada:
  - Um * após o nome do elemento significa que ele pode ser repetido zero ou mais vezes – elemento multivalorado (repetitivo) opcional.
  - Um + após o nome do elemento significa que ele pode ser repetido uma ou mais vezes – elemento multivalorado (repetitivo) obrigatório.
  - Um ? após o nome do elemento significa que ele pode ser repetido zero ou uma vez – elemento de único valor (não repetitivo) opcional.
  - Um elemento sem qualquer um dos três símbolos anteriores precisa aparecer exatamente uma vez no documento – elemento de único valor (não repetitivo) obrigatório.

DTD - Document Type Definition

Definição da DTD:
livraria.dtd




```
<!ELEMENT livraria (livro)+>
<!ELEMENT livro (titulo,editora, autor+)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT editora (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ATTLIST livro ISBN CDATA]>
```

XML Schema

- **Da mesma forma que a DTD.**
  - Utilizado para descrever a estrutura de um documento XML.
- **Utiliza sintaxe XML.**
- **Sintaxe simples: fácil compreensão humana.**
- **Introduz tipos de dados.**
  - data, string, números, etc.

XML Schema

- **Como em DTD, define:**
  - Elementos e atributos que aparecem em um documento.
  - Aninhamento de elementos.
  - Ordem dos elementos.
  - Número de elementos (“cardinalidade”).
  - Se um elemento é vazio ou pode incluir texto.
- **Novidades:**
  - Define tipos de dados (*data types) para elementos e* atributos.
  - Define valores *default e fixos para atributos e elementos*.
  - Suporta *namespaces*.
  - Escrito em XML.

Declaração de chamada

Chamada da DTD

```
<?xml version="1.0"?>
<!DOCTYPE artigo SYSTEM "artigo.dtd">
<artigo>
  <titulo> ... ... </titulo>
  <autor>
    <nome> ... ... </nome>
  </autor>
</artigo>
```

Chamada do XML Schema

```
<?xml version="1.0"?>
<artigo xmlns="http://www.my.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="c:/artigo.xsd">
 <titulo> ... ... </titulo>
  <autor>
    <nome> ... ... </nome>
  </autor>
<artigo>
```

Estrutura de um arquivo XMLSchema

- **Elemento root:**

```
<xs:schema>

  <!-- declaração de tipos, elementos e
  atributos -->

</xs:schema>
```

Extensao do arquivo: .xsd

Cardinalidade

```
<xs:complexType name="tProduto">
   <xs:sequence>
        <xs:element name="nome" type="xs:string"/>
        <xs:element name="quantidade" type="xs:integer"
                    minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="classificacao" type="tClassific
                    minOccurs="0" />
   </xs:sequence>
</xs:complexType>

<xs:complexType name="tClassific">
   <xs:sequence>
        <xs:element name="nome" type="xs:string"/>
        <xs:element name="email" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

Sequence – Exemplo

- **No XML Schema:**

```
<xs:complexType name="tPessoa">
   <xs:sequence>
        <xs:element name="nome" type="xs:string"/>
        <xs:element name="fone" type="xs:integer"/>
        <xs:element name="email" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
<xs:element name="Funcionario" type="tPessoa"/>
```

- Na instancia XML:

```
<Funcionario>
   <nome>Osvaldo da Silva</nome>
   <fone>1212121</fone>
   <email>OdaS@abc.def.br</email>
</Funcionario>
```

Choice - Exemplo

- **No XMLSchema:**

```
<xs:complexType name="tPublic">
  <xs:sequence>
     <xs:element name="nome" type="xs:string"/>
     <xs:choice>
       <xs:element name="ISBN" type="xs:integer"/>
       <xs:element name="volume" type="xs:integer"/>
     </xs:choice>
 </xs:sequence>
</xs:complexType>
<xs:element name="publicacao" type="tPublic"/>
```

- Na instancia XML:

```
<publicacao>
   <nome>Projeto de Banco de dados</nome>
   <ISBN>989898989</ISBN>
</publicacao>
```

```
<publicacao>
     <nome>SQL Magazine</nome>
     <volume>9</volume>
</publicacao>
```

All - Exemplo

- **No XMLSchema:**

```
<xs:complexType name="tAut">
  <xs:all>
     <xs:element name="nome" type="xs:string"/>
     <xs:element name="email" type="xs:integer"/>
     <xs:element name="instituicao" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:element name="autor" type="tAut"/>
```

- Na instancia XML:

```
<autor>
   <nome>Ana Clara</nome>
   <email>ana@server.domain</email>
   <instituicao>Universidade XYZ</instituicao>
</autor>
```

- Todos juntos ou nada
- Sem restrição de ordem

Restrições

- **A restrição "enumeration"**
  - Limita um tipo simples a um conjunto de valores distintos.

```
<xs:simpleType name="TipoFigura">
  <xs:restriction base="xs:string">
    <xs:enumeration value = "jpeg"/>
    <xs:enumeration value = "gif"/>
    <xs:enumeration value = "bmp"/>
    <xs:enumeration value = "tiff"/>
    <xs:enumeration value = "wmf"/>
  </xs:restriction>
</xs:simpleType>
<xs:attribute name="tipo" type="TipoFigura"/>
```

Restrições

- **Comparando o "enumeration" no XMLSchema com a DTD:**

**** XMLSchema**

```
<xs:simpleType name="TipoFigura">
  <xs:restriction base="xs:string">
    <xs:enumeration value = "jpeg"/>
    <xs:enumeration value = "gif"/>
    <xs:enumeration value = "bmp"/>
    <xs:enumeration value = "tiff"/>
    <xs:enumeration value = "wmf"/>
  </xs:restriction>
</xs:simpleType>
<xs:attribute name="tipo" type="TipoFigura"/>
```

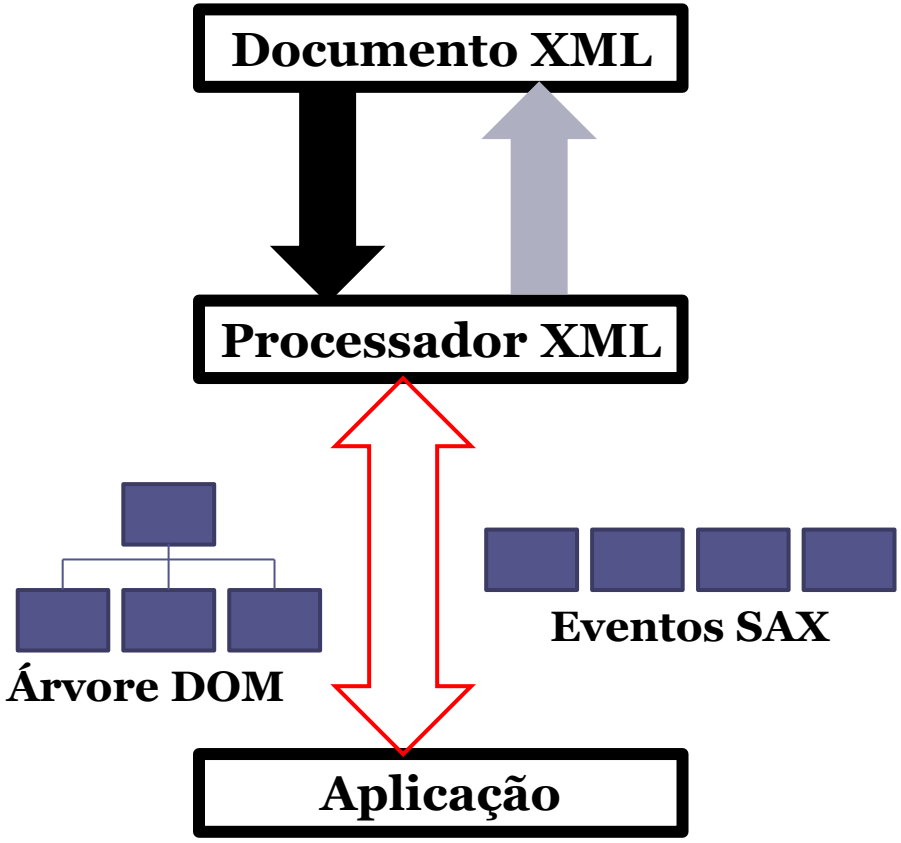
**** DTD (só é possível delimitar valores para atributos)**

```
<!ELEMENT figura ANY>
<!ATTLIST figura
    tipo NOTATION (jpeg|gif|bmp|tiff|wmf)>
```

Tipos de dados primitivos

- xs:string
- xs:boolean
- xs:decimal
- xs:float
- xs:double
- xs:duration
- xs:dateTime
- xs:time
- xs:date
- xs:gYearMonth
- xs:gYear
- xs:gMonthDay
- xs:gDay
- xs:gMonth
- xs:hexBinary
- xs:base64Binary
- xs:anyURI
- xs:QName
- xs:NOTATION

Manipulações



APIs para processamento de XML

- DOM (Document Object Model)
  - Modelo-padrão com um conjunto associado de funções de API (Application Programming Interface).
  - Permite que os programas manipulem a representação de árvore resultante correspondente a um documento XML bem formado.
  - No entanto, o documento inteiro precisa ser analisado de antemão quando se usa DOM, para converter o documento para a representação na estrutura de dados interna DOM padrão.

DOM – Quando usar...

- Quando for necessário modificar o documento.
- Quando for necessário avançar e retroceder no documento.
- Quando o arquivo XML for pequeno.

APIs para processamento de XML

- SAX (Simple API for XML)
  - Permite o processamento de documentos XML no ato ao notificar o programa de processamento por meio de chamadas de eventos sempre que uma tag de início ou fim for encontrada.
  - Facilita o processamento de grandes documentos e permite o uso dos chamados documentos **XML de streaming**, em que o programa de processamento pode processar as tags à medida que forem encontradas.
  - Isso também é conhecido como **processamento baseado em evento**.

SAX

- Não carrega o documento XML para a memória.
- Não cria uma representação do documento XML.
- Usa métodos callback para informar a estrutura do documento XML.
- É mais rápido e usa menos memória que parser DOM.
- Provê acesso sequencial ao conteúdo de um documento XML.
- Processamento em fluxo. Ideal para leituras contínuas em disco ou recebimento através da rede.

SAX – Quando usar...

- Quando não for necessário fazer modificações no XML.
- Quando for necessário acessar arquivos XML muito grandes.

Linguagens de consulta

- Houve várias propostas para linguagens de consulta XML, e dois padrões para linguagens de consulta XML se destacaram.
- O primeiro é o **XPath**, que oferece construções da linguagem para especificar expressões de caminho a fim de identificar certos nós (elementos) ou atributos em um documento XML que combina padrões específicos.
- O segundo é o **XQuery**, que é uma linguagem de consulta mais geral.
  - Usa expressões XPath, mas tem construções adicionais.

Exemplos de expressões XPath

1. /empresa
2. /empresa/departamento
3. //funcionario [salarioFuncionario gt 70.000]/nomeFuncionario
4. /empresa/funcionario [salarioFuncionario gt 70.000]/nomeFuncionario
5. /empresa/projeto/trabalhadorProjeto [horas ge 20,0]

XPath

- Exemplo(1)
  - Obtenha todas as informações do livro cujo título é “Inside XML”.

document(“liv.xml”) / livraria / livro[titulo =“Inside XML”]

XQuery

- A XPath nos permite escrever expressões que selecionam itens de um documento XML estruturado em árvore.
- A XQuery possibilita a especificação de consultas mais gerais sobre um ou mais documentos XML.

XQuery

- A forma típica de uma consulta em XQuery é conhecida como **expressão FLWR**, que indica as quatro cláusulas principais da XQuery e tem a seguinte forma:
  - FOR <vínculos de variável para nós (elementos) individuais>
  - LET <vínculos de variável para coleções de nós (elementos)>
  - WHERE <condições qualificadoras>
  - RETURN <especificação de resultado da consulta>

XQuery

FOR <vínculos de variável para nós (elementos) individuais>

LET <vínculos de variável para coleções de nós (elementos)>

WHERE <condições qualificadoras>

RETURN <especificação de resultado da consulta>

XQuery

```
LET $d := doc(www.empresa.com/info.xml)
FOR $x IN $d/empresa/projeto[NumeroProjeto = 5]/
      trabalhadorProjeto, $y IN $d/empresa/funcio-
      nario
WHERE $x/horas gt 20.0 AND $y.cpf = $x.cpf
RETURN <res> $y/nomeFuncionario/primeiroNome,
                $y/nomeFuncionario/ultimoNome,
                $x/horas </res>
```

XQuery

- XML Query Language
- Proposta pela W3C e agrega características de diversas outras linguagens de consulta para XML, bem como SQL e OQL
- É uma linguagem funcional na qual a consulta é representada como uma expressão
- Utiliza o conceito de expressões de caminho para navegar em árvores
- Xquery é flexível o suficiente para consultar vários tipos de fontes de informação XML incluindo bancos de dados e documentos

Expressões FLWR

Consiste de uma seqüência de um ou mais cláusulas FOR e/ou LET, seguidas por um WHERE opcional e terminada por um RETURN

FLWR = FOR-LET-WHERE-RETURN

- FOR e LET: associam valores a uma ou mais variáveis
- WHERE: expressão condicional
- RESULT: retorna o resultado

XQuery

Uma consulta em XQuery é uma expressão que:

- Lê um número de documentos XML ou fragmentos de documentos XML
- Retorna uma seqüência de fragmentos XML bem-formados

Expressões XQuery podem ser:

- Expressões de caminho
- Construtores de elementos
- Expressões FLWR
- Expressões condicionais
- Expressões com quantificadores

Outras linguagens e protocolos relacionados a XML

- WSDL (Web Services Description Language)
  - Permite a descrição de Web Services em XML.
  - Isso torna o Web Service disponível para usuários e programas pela Web.
- SOAP (Simple Object Access Protocol)
  - Protocolo independente de plataforma e de linguagem de programação para transmissão de mensagens e chamadas de procedimento remoto.

Referências

- Elsmari, R., Navathe, Shamkant B. “Sistemas de Banco de Dados”. 6ª Edição, Pearson Brasil, 2011.
- Apostila FJ22 da Caelum – Cap. 4 - Trabalhando com XML
- Java XML Tutorial
  - https://www.mkyong.com/tutorials/java-xml-tutorials/
- XML Tutorial – W3 Schools
  - http://www.w3schools.com/xml/
- XLST Tutorial – W3 Schools
  - http://www.w3schools.com/xsl/
- www.xml.org
- www.xml.xom
- www.msdn.microsoft.com/xml
- www.xmlsoftware.com
- www.w3c.org

Obrigado!

Dúvidas, comentários, sugestões?

Regis Pires Magalhães

regismagalhaes@ufc.br