

Módulo III

Introdução a XML

Prof. Ismael H F Santos

Ementa

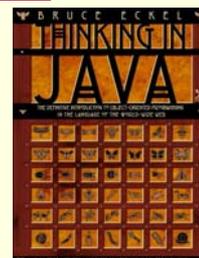
- **Modulo III – XML**
 - SGML - Standard Generalized Markup Language
 - XML - Extensible Markup Language
 - XML Validação
 - DTD x XML/Schema
 - XML Processing - XSLT

Bibliografia

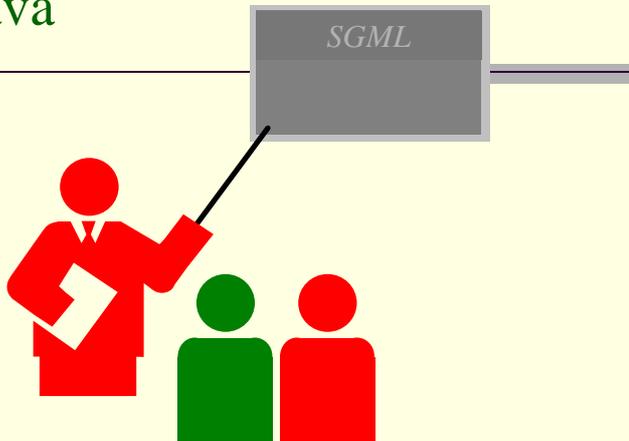
- *Linguagem de Programação JAVA*
 - Ismael H. F. Santos, Apostila UniverCidade, 2002
- *The Java Tutorial: A practical guide for programmers*
 - Tutorial on-line: <http://java.sun.com/docs/books/tutorial>
- *Java in a Nutshell*
 - David Flanagan, O'Reilly & Associates
- *Just Java 2*
 - Mark C. Chan, Steven W. Griffith e Anthony F. Iasi, Makron Books.
- *Java 1.2*
 - Laura Lemay & Rogers Cadenhead, Editora Campos

Livros

- **Core Java 2**, Cay S. Horstmann, Gary Cornell
 - Volume 1 (Fundamentos)
 - Volume 2 (Características Avançadas)
- **Java: Como Programar**, Deitel & Deitel
- **Thinking in Patterns with JAVA**, Bruce Eckel
 - **Gratuito.** <http://www.mindview.net/Books/TIJ/>



POO-Java



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

5

SGML - Standard Generalized Markup Language

- **ISO 8879 definido em 1986**
- **Conceitos básicos:**
 - **separação entre conteúdo e apresentação de documentos**
 - **conteúdo estruturado logicamente**
 - **informação específica à apresentação**
 - **objetivo principal do SGML é estruturar a informação, lidando com conteúdo e estrutura**
 - **apresentação é feita usando outros mecanismos (*style sheets*)**

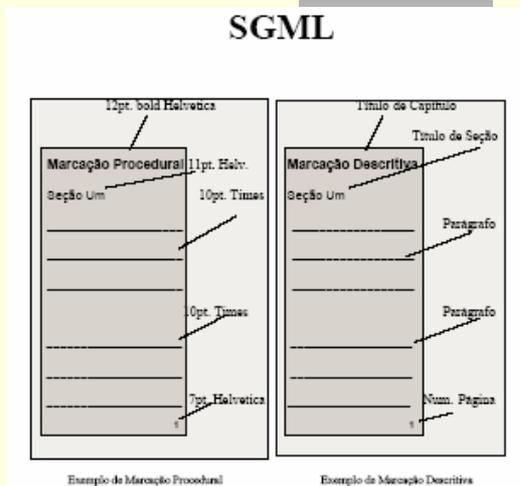
April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

6

SGML - Standard Generalized Markup Language

- **Marcação descritiva**
(documento pode ser processado por diferentes programas)
- **Documento tipado**
(método padrão para descrever a estrutura do documento)
- **Independência de sistema**
para representar o *script* no qual o texto é escrito



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

7

SGML - Standard Generalized Markup Language

- **Uso em larga escala:**
 - HTML 2.0: especificado como aplicação SGML (1994)
- **Conteúdo:**
 - elementos lógicos estruturados hierarquicamente => **Árvore do Documento (Document Tree)**
- **Processamento automatizado do documento**
 - índice, lista de figuras, tabelas



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

8

SGML

■ Classe de documentos especificada através de uma DTD - Document Type Definition

- os elementos de uma classe de documentos e seus atributos
- as regras para combinar esses elementos, especificando o conteúdo permitido para cada elemento

```

<antologia>
<poema><titulo>São demais os perigos desta
vida</titulo>
<estrofe>
<linha>São demais os perigos desta vida</linha>
<linha>pra quem tem paixão</linha>
<linha>Principalmente quando uma lua chega de
repente</linha>
<linha>e se deixa no céu como esquecida</linha>
</estrofe>
<estrofe>
<estrofe>
<linha>E se ao luar que atua desvaído</linha>
<linha>vem se unir uma música qualquer</linha>
<linha>Aí, então, é preciso ter cuidado</linha>
<linha>porque deve andar perto uma mulher</linha>
</estrofe>
<estrofe>
<linha>Uma mulher que é feita</linha>
<linha>de música, luar e sentimento</linha>
<linha>E que a vida não quer de tão perfeita</linha>
</estrofe>
<estrofe>
<estrofe>
<linha>Uma mulher que é como a própria
lua</linha>
<linha>Tão linda que só espalha sofrimento</linha>
<linha>Tão cheia de pudor que vive nua</linha>
</estrofe>
</poema>
<!-- mais poemas -->
</antologia>
    
```

April 05

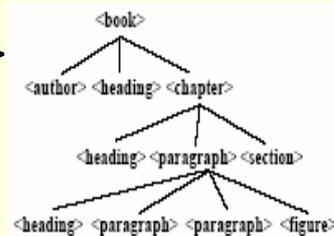
Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

9

SGML

■ DTD

- <!ELEMENT antologia (poema+)>
- <!ELEMENT poema (titulo?,estrofe+)>
- <!ELEMENT titulo (#PCDATA)>
- <!ELEMENT estrofe (linha+)>
- <!ELEMENT linha (#PCDATA)>



■ Exemplo de DTD de um livro:

- <!ELEMENT book (author, heading, chapter+)>
- <!ELEMENT chapter (heading, (paragraph|figure)*, section*)>
- <!ELEMENT section (heading, (paragraph|figure)*)>
- <!ELEMENT (author|heading) (#PCDATA)>
- <!ELEMENT paragraph ((#PCDATA|reference)*)>
- <!ELEMENT reference EMPTY>

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

10

SGML

- **Especificando atributos dos elementos**
 - `<!ATTLIST (chapter|section) id CDATA #IMPLIED>`
 - `<!ATTLIST reference id CDATA #REQUIRED type (section|page) section>`
- **Usando**
 - `<chapter id="sgml">`
 - `<heading>Standard Generalized Markup Language</heading>`
 - `<paragraph> ... </paragraph>`
 - `<section id="content">`
 - `<heading>Content and Presentation</heading>`
 - `<paragraph> ... </paragraph>`
 - ... in section `<reference type="section" id="content"></reference>` on page `<reference type="page" id="content"></reference>` bla bla bla ...

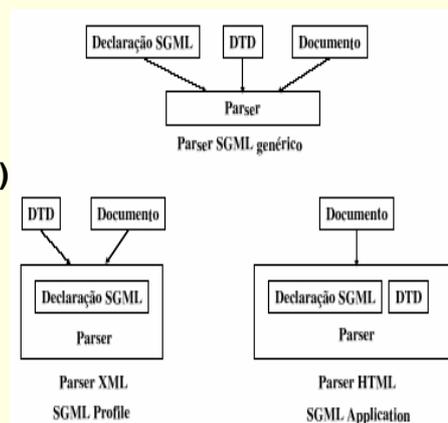
April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

11

Parser SGML

- **Declaração SGML**
 - delimitadores de marcação
 - nomes reservados (ELEMENT, ATTLIST, ...)
 - tamanho máximo dos nomes dos elementos
 - se delimitadores de início e fim podem ser opcionais
- • DTD
- • Documento

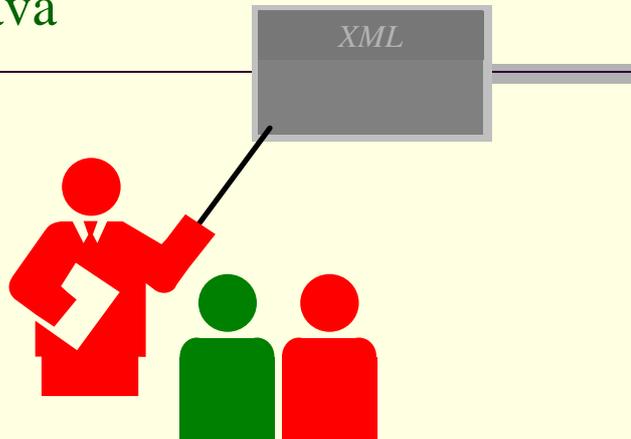


April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

12

POO-Java



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

13

XML – Extensible Markup Language

- XML é um documento de Texto
- Dois tipos de elementos
 - Marcação – Guarda a estrutura do documento
 - Dados – Informação propriamente dita
- Uma maneira de representar informação
 - não é uma linguagem específica
 - não define vocabulário de comandos
 - não define uma gramática, apenas regras mínimas
- Exemplo:

```
usuario_33.xml
<contato_codigo="33">
  <nome>Severino Severovitch</nome>
  <email>bill@norte.com.br</email>
  <telefone tipo="celular">
    <area>11</area>
    <numero>9999 4321</numero>
  </telefone>
</contato>
```

elemento

atributo

"nó" de texto

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

14

XML x HTML

- **HTML mostra como apresentar**

```
<h1>Severino Severovitch</h1>
<h2>bill@norte.com.br</h2>
<p>
  <b>11</b>
  <i>9999 4321</i>
</p>
```

- **XML mostra o que significa**

```
<nome>Severino Severovitch</nome>
<email>bill@norte.com.br</email>
<telefone>
  <ddd>11</ddd>
  <numero>9999 4321</numero>
</telefone>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

15

Anatomia de um documento XML

- Documentos XML são documentos de texto Unicode

- É uma hierarquia de **elementos** a partir de uma **raiz**
- Menor documento tem um elemento (vazio ou não):

```
<nome>Северино Северович </nome>
```

Elemento raiz

- Menor documento contendo elemento vazio

```
<nome></nome> = <nome/>
```

- Menor documento contendo elemento e conteúdo texto

```
<nome>Северино Северович</nome>
```

↑ Etiqueta inicial ↑ Conteúdo do Elemento ↑ Etiqueta final

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

16

Componentes de um documento XML

- **Árvore XML**
 - nós,
 - raiz,
 - galhos e
 - folhas
- **Prólogo**
- **Comentários**
- **Instruções de processamento**
- **Elementos**
- **Atributos**
- **Nós de texto**
- **Entidades**

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

17

Partes de um documento XML

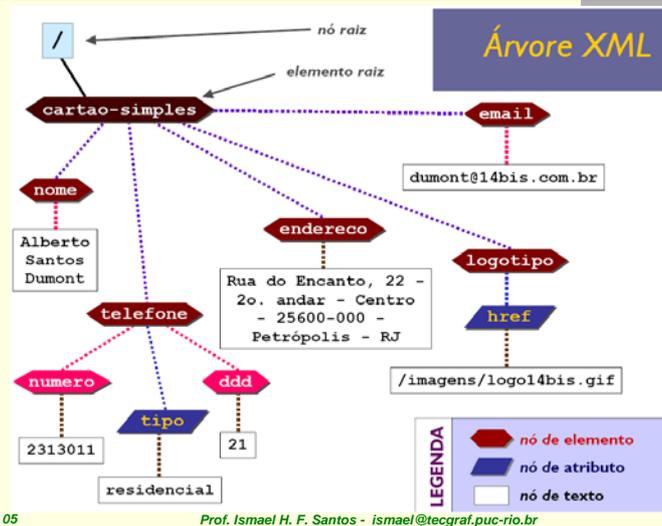


April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

18

Árvore XML



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

19

Estrutura XML

- Um documento XML pode ser representado como uma árvore. A estrutura é formada por vários nós.

```

<?xml version="1.0" encoding="iso-8859-1" ?>
<!-- Isto é um comentário -->
<cartao-simples>
  <logotipo href="/imagens/logo14bis.gif" />
  <nome>Alberto Santos Dumont</nome>
  <endereco>Rua do Encanto, 22 - 2o. andar -
  Centro - 25600-000 - Petrópolis - RJ</endereco>
  <email>dumont@14bis.com.br</email>
  <telefone tipo="residencial">
    <ddd>21</ddd>
    <numero>2313011</numero>
  </telefone>
</cartao-simples>
  
```

informações usadas pelo processador XML

um "nó" pode ser ...
 um elemento,
 um atributo,
 um bloco de texto,
 um comentário,
 uma instrução,
 uma declaração,
 uma entidade, ...

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

20

Prólogo XML

Prólogo

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-- Isto é um comentário -->
<?comando tipo="simples" parametro ?>
<!DOCTYPE cartao-simples SYSTEM "cartoes.dtd">
<cartao-simples>
  <logotipo href="/imagens/logo14bis.gif" />
  <nome>Alberto Santos Dumont</nome>
  <endereco>Rua do Encanto, 22 - 2o. andar -
  Centro - 25600-000 - Petrópolis - RJ</endereco>
  <email>dumont@14bis.com.br</email>
  <telefone tipo="residencial" >
    <ddd>21</ddd>
    <numero>2313011</numero>
  </telefone>
</cartao-simples>
```

Declarção XML
Comentário
Instrução de processamento
Declaração de tipo de documento

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

21

Nó raiz e elementos

elemento raiz nó raiz (/)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<cartao-simples>
  <logotipo href="/imagens/logo14bis.gif" />
  <nome>Alberto Santos Dumont</nome>
  <endereco>Rua do Encanto, 22 - 2o. andar -
  Centro - 25600-000 - Petrópolis - RJ</endereco>
  <email>dumont@14bis.com.br</email>
  <telefone tipo="residencial" >
    <ddd>21</ddd>
    <numero>2313011</numero>
  </telefone>
</cartao-simples>
```

elementos elementos

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

22

Atributos

- Só podem conter um descendente (só texto)

```
<?xml version="1.0" encoding="iso-8859-1" ?>

<cartao-simples>
  <logotipo href="/imagens/logol4bis.gif" />
  <nome>Alberto Santos Dumont</nome>
  <endereco>Rua do Encanto, 22 - 2o. andar -
  Centro - 25600-000 - Petrópolis - RJ</endereco>
  <email>dumont@14bis.com.br</email>
  <telefone tipo="residencial" >
    <ddd>21</ddd>
    <numero>2313011</numero>
  </telefone>
</cartao-simples>
```

atributos

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

23

Nós de Texto

- Não podem ter descendentes (são as folhas da árvore)

```
<?xml version="1.0" encoding="iso-8859-1" ?>

<cartao-simples>
  <logotipo href="/imagens/logol4bis.gif" />
  <nome>Alberto Santos Dumont</nome>
  <endereco>Rua do Encanto, 22 - 2o. andar -
  Centro - 25600-000 - Petrópolis - RJ</endereco>
  <email>dumont@14bis.com.br</email>
  <telefone tipo="residencial" >
    <ddd>21</ddd>
    <numero>2313011</numero>
  </telefone>
</cartao-simples>
```

nós de texto

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

24

Entidades

- São constantes associadas a um valor de texto
 - Podem aparecer em qualquer lugar do documento
- Substituídas **durante** o processamento do documento
- Sintaxe:
 - `&ENTIDADE;`
- Exemplo:
 - `&data_de_hoje;`
- Entidades pré-definidas:
 - `<` que corresponde a `<`
 - `>` que corresponde a `>`
 - `&` que corresponde a `&`
 - `"` que corresponde a `"`
 - `'` que corresponde a `'`

Entidades de caracteres

- Substituídas durante o processamento do documento
- Sintaxe:
 - `&#CÓDIGO_16b_decimal;`
 - `ÓDIGO_16b_hexadecimal;`
- Exemplo:
 - ` ` ou ` `
 - Um espaço em Unicode
 - Veja em www.unicode.org/charts/

Elementos e Atributos

- **Regras básicas**
 - Etiqueta inicial e final têm que ter o mesmo nome (considerando diferença de maiúscula e minúscula)
 - Não pode haver espaço depois do < nas etiquetas iniciais nem depois do </ nas finais
 - Atributos têm sempre a forma nome="valor" ou nome='valor':
 - aspas podem ser usadas entre apóstrofes e apóstrofes podem ser usados entre aspas
 - aspas e apóstrofes não podem ser neutralizados mas sempre podem ser representados pelas entidades ' e "
 - Não pode haver atributos na etiqueta final

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

27

Elementos e Atributos (2)

- **Elementos mal formados**

```
<Profissão>Arquiteto</profissão>
<TR><TD>item um</td></tr>
<ÄÄÍÄËË>139.00</äâíãèé>
```
- **Há várias maneiras de representar a mesma informação em XML**

```
<data>23/02/1998</data>
<data dia="23" mes="02" ano="1998" />
<data>
  <dia>23</dia>
  <mes>02</mes>
  <ano>1998</ano>
</data>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

28

Quando usar elementos/atributos

- **Questão de design**
 - *Elementos geralmente referem-se a coisas que têm atributos*
 - *Atributos geralmente são características dessas coisas que podem ser descritas com poucas palavras*
- **Questão de suporte tecnológico**
 - *Atributos não podem conter subelementos*
 - *Atributos são mais fáceis de serem validados num DTD*

Identificadores

- **Nomes de atributos e elementos**
- **Podem conter**
 - *qualquer caractere alfanumérico ou ideograma*
 - *. (ponto)*
 - *- (hífen)*
 - *_ (sublinhado)*
- **Não podem começar com**
 - *ponto,*
 - *hífen ou*
 - *número*

Identificadores (2)

- *Elementos bem formados*

```
<αριστοτελεσ>περι ποιητικησ</αριστοτελεσ>
<êíèää xml:lang='ru'>
  <íàçääíèà>Äääáíèé ííääèí</íàçääíèà>
  <ääóíð ðíæääíèå="1799"
    ñíáðóú="1837">Äèèñàíáð Nððáááåè÷ Íóæèèí</ääóíð>
</êíèää>
<_1_/>
<cdd:gen.inf cdd:cod="005">Introdução a XML</cdd:gen.inf>
```

- *Elementos mal formados*

```
<3-intro>Fundamentos</3-intro>
<cartão de crédito>1234567887654321</cartão de crédito>
```

Conteúdo misto

```
<trecho>
<secao>2</secao>
<paragrafo>A unidade de informação
dentro de um documento XML é o
<definicao>elemento</definicao>. Um
elemento é formado por duas
<definicao>etiquetas</definicao> que
atribuem algum significado ao conteúdo.
</paragrafo>
</trecho>
```

Seção CDATA (Character DATA)

- *Ignora efeitos especiais dos caracteres*

```
<titulo>Curso de XML</titulo>
<exemplo>Considere o seguinte trecho de
XML:
<![CDATA[
    <empresa>
        <nome>João & Maria S/A</nome>
    </empresa>
]]>
</exemplo>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

33

Instruções de processamento

- *Instruções dependentes do processador*
- *Funcionam como comentários para os processadores que não a conhecem*

```
<?nome-do-alvo área de dados ?>
<?query-sql select nome,
              email
              from agenda
              where id=25 ?>
```

- *Iguais aos comentários HTML*

```
<!-- Isto é um comentário -->
<!-- isto é um erro -- sério! -->
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

34

Declaração XML

- *Opcional (exceto quando o conjunto de caracteres usado for diferente de UTF-8)*

```
<?xml version="1.0"
      encoding="iso-8859-1"
      standalone="yes" ?>
```

Documento XML bem-formatado

- **Documento bem-formatado**
 - *ter um único elemento raiz*
 - *etiquetas iniciais e finais combinam (levando em conta que caracteres maiúsculos e minúsculos são diferentes)*
 - *elementos bem aninhados*
 - *valores de atributos entre aspas ou apóstrofes*
 - *atributos não repetidos*
 - *identificadores válidos para elementos e atributos*
 - *comentários não devem aparecer dentro das etiquetas*
 - *sinais < ou & nunca devem ocorrer dentro dos valores dos atributos ou nos nós de texto do documento.*

XML Namespaces

- Permite que elementos de mesmo nome de diferentes aplicações sejam misturados sem que haja conflitos
- Um namespace (universo de nomes) é declarado usando atributos reservados
 - `xmlns="identificador"` (namespace default)
 - associa o identificador com todos os elementos que não possuem prefixo. Ex: `<nome>`
 - `xmlns:prefixo="identificador"`
 - associa o identificador com os elementos e atributos cujo nome local é precedido do prefixo. Ex `<prefixo:nome>`
- O prefixo é arbitrário e só existe dentro do documento
- O identificador (geralmente uma URI) deve ser reconhecido pela aplicação

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

37

XML Namespaces

- **Limita o escopo de elementos**
 - Evita conflitos quando duas linguagens se cruzam no mesmo documento
- **Consiste da associação de um identificador a cada elemento/atributo da linguagem, que pode ser**
 - herdado através do escopo de uma sub-árvore
 - atribuído explicitamente através de um prefixo

Exemplo

```
<cadastro xmlns:firma="01.234.567/0001-99">  
  <nome>Severino Severovitch</nome>  
  <firma:nome>Sibéria Informática Ltda.</firma:nome>  
  <email>bill@norte.com.br</email>  
</cadastro>
```

Este elemento `<nome>` pertence a outro namespace

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

38

Outro Exemplo

Vale para todo o elemento <cartao>

Esta URI está associada a este prefixo

```
<ct:cartao xmlns:ct="01.234.567/0001-89/cartoes">
  <ct:nome>Alberto Santos Dumont</ct:nome>
  <ct:endereco>Rua do Encanto, 22 - Centro
  25600-000 - Petrópolis - RJ</ct:endereco>
  <ct:email>dumont@14bis.com.br</ct:email>
  <ct:telefone tipo="residencial">
    <ct:ddd>21</ct:ddd>
    <ct:numero>2313011</ct:numero>
  </ct:telefone>
</ct:cartao>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

39

Exemplo com 3 Namespaces

Namespace default

URI padrão XHTML

```
<departamento
  xmlns:ct="01.234.567/0001-89/cartoes"
  xmlns="01.234.567/0001-89/empresa"
  xmlns:html="http://www.w3.org/1999/xhtml">
  <ct:nome>Fulano de Tal</ct:nome>
  <nome>Contabilidade</nome>
  <endereco>Rua Projetada, 33</endereco>
  <html:a href="web.html">
    <html:strong>link negrito HTML</html:strong>
  </html:a>
  <urgencia><ct:numero>2313011</ct:numero></urgencia>
</departamento>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

40

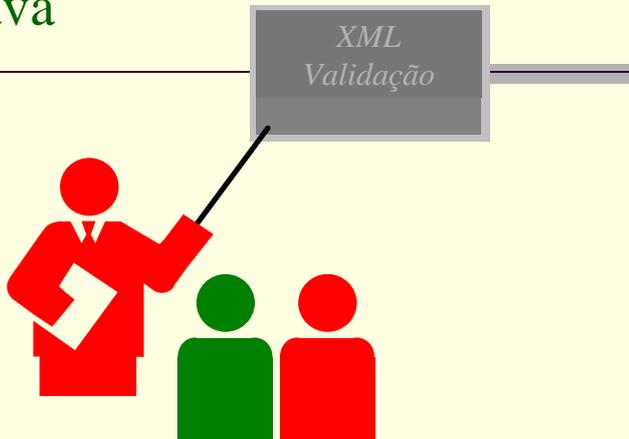
Por que usar XML para compartilhar dados?

- **Porque é um padrão aberto**
 - *Facilidade para converter para formatos proprietários*
- **Porque é texto**
 - *Fácil de ler, fácil de processar, menos incompatibilidades*
- **Porque promove a separação entre estrutura, conteúdo e apresentação**
 - *Facilita geração de dados para visualização dinâmica*
 - *Evita repetição de informação / simplifica manutenção*

Por que usar XML para compartilhar dados?

- **Porque permitirá semântica na Web**
 - *Elementos HTML não carregam significado, apenas dicas de formatação: mecanismos de busca ficam prejudicados*
 - *Solução com XML dependerá de suporte dos clientes*

POO-Java



Por que validar ?

- *Para a maior parte das aplicações, um XML bem formado é suficiente*
- *É possível, em documentos XML não válidos*
 - *Montar a árvore usando DOM*
 - *Extrair nós, acrescentar nós, alterar o conteúdo dos elementos usando SAX ou DOM*
 - *Transformar o documento em outro usando XSLT*
 - *Gerar um PDF ou um SVG com dados contidos no documento*
 - *Exibir o XML em um browser usando CSS*
- *Para que serve, então, o trabalho de desenvolver um DTD?*

Definindo um Esquema XML

- *Para se ter uma linguagem precisa-se de um **esquema***
 - *Linguagem implica comunicação.*
 - *Não há comunicação eficiente sem uma convenção entre as partes sobre vocabulários, regras de formação, etc.*
- *Documentos não válidos são "individualistas"*
 - *Um esquema representa um **conjunto** de documentos, que existem e que virão a existir*
 - *É possível fazer muitas coisas com **UM** documento não válido. É difícil automatizar os processos sem considerar uma **CLASSE** de documentos*
- *Um esquema é uma formalidade necessária*
 - *Se você tem uma grande coleção de documentos que foram construídos segundo determinadas regras, você já tem, **informalmente**, um esquema*
 - *Para validar documentos de acordo com suas convenções, é preciso ter um esquema*

April 05

45

Classes x Instâncias

- *Um esquema define uma classe de documentos*
 - *Os documentos que quiserem fazer parte dessa classe devem aderir ao esquema*
- *Um documento pode pertencer a várias classes*
 - *Um documento pode ser válido em vários esquemas*

April 05

Prof. Ismael H. F. Santos - ismael@tegraf.puc-rio.br

46

Documentos Válidos

- **Documentos válidos contém**
 - **Declaração de tipo de documento (para DTD), ou**
`<!DOCTYPE bilhete SYSTEM "bilhete.dtd">`
 - **Declaração de namespace e schema (para XML Schema)**
`<bilhete codigo="ZMIKT8"
xmlns="urn:123456789"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:123456789 bilhete.xsd">`
- **Para validar**
 - **Use um validador (com suporte a Schema)**

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

47

O que define um Esquema XML

- **Um vocabulário**
 - **Elementos, atributos**
- **Uma gramática**
 - **Relacionamentos**
- **Uma coleção de entidades**
 - **No caso dos DTDs**

```
< Pessoa >  
  < nome >  
    < prenome >Richard</ prenome >  
    < sobrenome >Feynman</ sobrenome >  
  </ nome >  
  < profissao >Fisico</ profissao >  
  < profissao >Matematico</ profissao >  
  < profissao >Arrombador de cofres</ profissao >  
</ Pessoa >
```

DTD

```
<!ELEMENT Pessoa (nome, profissao*)>  
<!ELEMENT nome (prenome, sobrenome)>  
<!ELEMENT prenome (#PCDATA)>  
<!ELEMENT sobrenome (#PCDATA)>  
<!ELEMENT profissao (#PCDATA)>
```

```
<!DOCTYPE Pessoa SYSTEM  
  "http://pessoas.com/pessoa.dtd">  
< Pessoa >  
  < nome >  
    < prenome >Richard</ prenome >  
    < sobrenome >Feynman</ sobrenome >  
  </ nome >  
</ Pessoa >
```

Documento Válido

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

48

Documentos não-válidos

- Diga porque os documentos são não-válidos ?

```
<!DOCTYPE pessoa SYSTEM
      "http://pessoas.com/pessoa.dtd">
<pessoa>
  <nome>
    <prenome>Richard</prenome>
  </nome>
  <profissão>Arrombador de cofres</profissão>
</pessoa>
```

```
<!DOCTYPE pessoa SYSTEM
      "http://pessoas.com/pessoa.dtd">
<pessoa>
  <profissão>Arrombador de cofres</profissão>
  <nome>
    <prenome>Richard</prenome>
    <sobrenome>Feynman</sobrenome>
  </nome>
  <profissão>Fisico</profissão>
</pessoa>
```

DTD Externo SYSTEM

- Como vincular

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE cartao-simples SYSTEM "cartao.dtd">
<cartao-simples>
  <nome> (... )
```

nome do elemento
raiz do documento

onde buscar validação:
SYSTEM ou PUBLIC

URI ou
identificador

- Elementos:

- **<!DOCTYPE>** vincula o DTD a um documento
- **<!ELEMENT>** define um elemento
- **<!ATTLIST>** define os atributos de um elemento
- **<!ENTITY>** define uma entidade (ex: **&nome;**)
- **<!NOTATION>** define uma notação interna para uma URI

DTD Publico

- Comum em linguagens consolidadas
- Não precisa de URI, mas deve ser fornecida por segurança
- FPI: formal public identifier
 - arquivo chamado de "catalog" guardado com a aplicação verifica o FPI
 - se não encontrar, busca o DTD na URI fornecida

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"  
"http://www.w3.org/TR/REC-html40/strict.dtd">
```

DTD Interno

```
<?xml version="1.0" encoding="iso-8859-1" ?>  
<!DOCTYPE pessoa [  
  <!ELEMENT pessoa (nome, profissao*)>  
  <!ELEMENT nome (prenome, sobrenome)>  
  <!ELEMENT prenome (#PCDATA)>  
  <!ELEMENT sobrenome (#PCDATA)>  
  <!ELEMENT profissao (#PCDATA)>  
<pessoa>  
  <nome>  
    <prenome>Richard</prenome>  
    </sobrenome>Feynman</sobrenome>  
  </nome>  
</pessoa>
```

DTD Incompleto

- O DTD foi importado, mas está incompleto. Parte dele é definido localmente

```
<!DOCTYPE pessoa SYSTEM "pessoa.dtd" [  
  <!ELEMENT nome (#PCDATA)>  
  <!ENTITY dtd "Document Type Definition">  
>
```

- Elementos, atributos e entidades definidos no documento têm prioridade sobre declarações importadas
 - Processador lê primeiro elementos locais, depois externos
 - A primeira declaração é usada. Declarações adicionais para o mesmo elemento/atributo/entidade são desconsideradas

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

53

DTD exemplo

DTD: exemplo

```
<!ELEMENT cartao-simples ( logotipo?,  
  nome, endereco,  
  email*,  
  telefone+ ) >
```

zero ou uma ocorrência (pointing to `logotipo?`)
exatamente uma ocorrência cada (pointing to `nome, endereco, email*, telefone+`)

definição de elemento (pointing to the whole DTD line)

```
<!ELEMENT logotipo EMPTY >  
<!ATTLIST logotipo  
  href CDATA #REQUIRED >  
<!ELEMENT nome ( #PCDATA ) >  
<!ELEMENT endereco ( #PCDATA ) >  
<!ELEMENT email ( #PCDATA ) >  
<!ELEMENT telefone ( ddd, numero ) >  
<!ATTLIST telefone  
  tipo ( residencial | comercial ) "residencial"  
  codigo-pais NMTOKEN #IMPLIED >  
<!ELEMENT ddd ( #PCDATA ) >  
<!ELEMENT numero ( #PCDATA ) >
```

zero ou mais ocorrências (pointing to `nome`)
uma ou mais ocorrências (pointing to `email*`)
uma ou mais ocorrências (pointing to `telefone+`)

não pode conter outros nós (exceto atributos) (pointing to `EMPTY`)

atributo obrigatório contém "Character DATA" (pointing to `#REQUIRED`)

um ou o outro (pointing to `residencial | comercial`)

valor default (pointing to `"residencial"`)

lista de atributos (pointing to the `ATTLIST` block)

atributo opcional contém nome ou número (pointing to `#IMPLIED`)

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

54

<!ELEMENT> e #PCDATA

- **Sintaxe**

```
<!ELEMENT nome_do_elemento conteudo >
```

- **O conteúdo pode ser**

- (1) (#PCDATA),
- (2) (uma seqüência),
- (3) (uma lista para escolha),
- (4) (conteúdo misto),
- (5) a palavra **EMPTY** ou
- (6) a palavra **ANY**.

- **PCDATA - Parsed Character Data**

- Elemento pode conter texto
- Não pode conter elementos

Seqüências de Elementos-filho

- **O conteúdo deve ter uma lista de elementos separados por *vírgula***

- **Podem ter sufixos indicando multiplicação**

- * zero ou mais
- + um ou mais
- ? zero ou um

- **Exemplos**

```
<!ELEMENT cartao (nome, telefone+,  
                  email*, website?) >  
<!ELEMENT trem (locomotiva, vagao*) >
```

Seleção de um Elemento-filho

- *Lista de elementos separada por | (barra vertical)*
- *Exemplos*

```
<!ELEMENT trecho (ferroviário | aéreo |
                  rodoviário | fluvial)* >
<!ELEMENT circulo (centro, (raio | diametro)) >
<!ELEMENT ponto ((x, y) | (r, θ)) >
<!ELEMENT nome (prenome | sobrenome |
                (prenome, inicial*, sobrenome))>
```

Conteúdo Misto

- **(#PCDATA | elem1 | ... | elemn)***
 - *#PCDATA tem que ser o primeiro elemento*
 - *O * no final é obrigatório*
 - *Não é possível controlar o número ou a estrutura dos elementos*
- *Exemplo: a seguinte declaração em DTD:*

```
<!ELEMENT frase (#PCDATA | enfase)*>
```

permite o seguinte conteúdo

```
<frase>A frase que <enfase>ele</enfase>
leu não foi a que <enfase>ela</enfase>
ouvIU.</frase>
```

Elementos Vazios e Any

- Elementos que não podem conter nada devem ser declarados como **EMPTY**

```
<!ELEMENT nome EMPTY>
```

- Elementos **EMPTY** podem conter atributos

```
<!ATTLIST nome prenome CDATA #REQUIRED>
```

- A declaração acima permite elementos como

```
<nome prenome="José" />
```

- **ANY**: para elementos que podem conter qualquer coisa

```
<!ELEMENT todos ANY>
```

<!ATTLIST>

- *Sintaxe*

```
<!ATTLIST elemento  
          atributo1 tipo valor_default>
```

```
<!ATTLIST elemento  
          atributo2 tipo valor_default>
```

```
...
```

ou

```
<!ATTLIST elemento  
          atributo1 tipo valor_default  
          atributo2 tipo valor_default  
          atributo3 tipo valor_default  
          atributo4 tipo valor_default
```

```
... >
```

Exemplos

```
<!ATTLIST voo de NMTOKEN #IMPLIED >
<!ATTLIST voo numero CDATA #FIXED "12/32-1">
<!ATTLIST voo para (REC|CGH|GRU|SDU) #REQUIRED >
<!ATTLIST voo transportador (RG | JH) "RG" >

<!ATTLIST voo de NMTOKEN #IMPLIED
          numero CDATA #FIXED "12/32-1"
          para (REC|CGH|GRU|SDU) #REQUIRED
          transportador (RG | JH) "RG" >
```

Tipos de Dados XML

CDATA	IDREF
NMTOKEN	IDREFS
NMTOKENS	ENTITY
<i>Seleção</i>	ENTITIES
ID	NOTATION

CDATA e NMTOKEN

- **CDATA** (*character data*) representa qualquer texto arbitrário
 - pode conter espaços, pontuação, etc.
- **NMTOKEN** (*name token*). É semelhante a um nome de elemento ou atributo
 - caracteres alfanuméricos
 - não pode conter espaços
- **NMTOKENS** representa um ou mais NMTOKEN separados por espaços.

```
<elemento atr="um dois três 56 21">
```

Seleção

- Uma Seleção é uma lista de NMTOKENS de onde pode-se escolher o valor do atributo. As escolhas são separadas por |:

```
<!ATTLIST voo para (REC|CGH|SDU) #REQUIRED>  
<!ATTLIST curso periodo (dia|noite) "noite">
```

- Elementos não podem conter espaços ou outros caracteres não-alfanuméricos
 - Tipo NMTOKEN!

ID

- **Atributos do tipo ID tem que conter um nome (e não NMTOKEN) que seja unívoco no documento.**
 - Nome tem mesmas regras que identificador XML (caracteres alfanumérico, não começa com número, etc.)
- **Não se pode usar um número como ID.**
 - A solução é colocar um prefixo antes do número que seja ou uma letra ou um sublinhado.
- **Elementos só podem ter um tipo ID.**
 - Não pode haver outro elemento na mesma página com mesmo ID
- **Exemplos**

```
<!ATTLIST voo codigo ID #REQUIRED>
<!ATTLIST piloto numero ID #REQUIRED>
```

IDREF

- **IDREF é referência para um ID. Exemplo:**

```
<!ATTLIST piloto alocado IDREF #REQUIRED>
```
- **Aplicação (codigo e numero são IDs)**

```
<aeroporto>
  <voo codigo="RG123"> ... </voo>
  <voo codigo="RG456"> ... </voo>
  <piloto numero="S1" alocado="RG123">
    ... </piloto>
  <piloto numero="S2" alocado="RG456">
    ... </piloto>
</aeroporto>
```

IDREFS

- *Lista de elementos IDREF. Exemplo:*

```
<!ATTLIST piloto alocado IDREFS #REQUIRED>
```

- *Aplicação (codigo e numero são IDs)*

```
<aeroporto>  
  <voo codigo="RG123"> ... </voo>  
  <voo codigo="RG456"> ... </voo>  
  <piloto numero="S2"  
    alocado="RG456 RG123">  
    ... </piloto>  
</aeroporto>
```

Valores Default

- **#REQUIRED:** força o autor do documento a definir um valor explícito ao atributo.
- **#IMPLIED:** o atributo é opcional.
- **#FIXED:** o atributo tem um valor fixo, constante
 - Valor não pode ser mudado
 - Autor não precisa definir atributo e, se o fizer, não pode mudar o valor. Exemplo:

```
<!ATTLIST valor moeda CDATA #FIXED "US$">
```

- **Valor inicial, entre aspas**

- Autor não precisa definir atributo, mas pode mudá-lo

```
<!ATTLIST voo companhia (RG | JH) "RG" >
```

```
<!ATTLIST endereco pais CDATA "Brasil" >
```

<!NOTATION> e tipo NOTATION

- **Associa URI a um nome**
 - Usado frequentemente para associar valores CDATA a NMTOKEN (mesmo onde não há URIs)
- **Exemplos**

```
<!NOTATION amazon SYSTEM "http://www.amazon.com">
<!NOTATION barnes SYSTEM "http://www.bn.com">
```
- **Tipo NOTATION de <!ATTLIST> é útil em situações onde não se pode usar CDATA**

```
<!ATTLIST book
  store NOTATION (amazon | barnes) #REQUIRED>
```
- **Assim pode-se limitar melhor valores dos atributos**

```
<book store="amazon" />
```

<!ENTITY>

- **Vários tipos de <!ENTITY>**
 - Constantes usadas nos DTDs e documentos XML
- **De parâmetro (%nome; - só no DTD)**
 - **internas** - incluem texto definido localmente
 - **externas** - incluem sub-DTDs de arquivos externos
- **Gerais (&nome; - só no XML)**
 - **internas** (texto local - sempre processadas)
 - **caracter** - ō 㪴
 - **globais** - < &
 - **definidas pelo usuário**
 - **externas** (carregam arquivos externos)
 - **processadas** - incluem texto no XML
 - **não-processadas** - incluem formatos binários (como imagens)

Entidades gerais internas

```
<!ENTITY nome "valor">
```

- *Exemplos:*

```
<!ENTITY empresa "ACME Indústria &amp;
Comércio de Coisas S.A.">
<!ENTITY copyright "<table>
<tr>
<td>Copyright &#x00A9 2000</td>
</tr>
</table>">
```

- *Uso: &nome; no documento XML*

```
<texto> Visite a &empresa; ainda hoje!.
</texto>
<div> &copyright; </div>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

71

Entidades gerais externas

- *Carregam texto de arquivos externos*

```
<!ENTITY nome SYSTEM "uri">
```

```
<!ENTITY nome PUBLIC "fpi" "uri">
```

- *Exemplo*

```
<!ENTITY menu_sup SYSTEM "/sec/menu.xml">
```

- *Uso*

```
<elemento>
&menu_sup;
</elemento>
```

- *Conteúdo de menu.xml:*

```
<menu>
  Texto
</menu>
```

- *Resultado*

```
<elemento>
  <menu>
    Texto
  </menu>
</elemento>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

72

Entidades externas não processadas

- Usadas para carregar dados que não podem ser processados (que não são texto) através de atributos
`<!ENTITY nome SYSTEM "uri" NDATA notação>`
- Depende de uma declaração NOTATION
 - Neste caso típico usada para informar tipo de dados
- Exemplo de uso no DTD
`<!NOTATION gif SYSTEM "image/gif">`
`<!ENTITY logo SYSTEM "logo.gif" NDATA gif>`
- Atributos podem declarar receber tipo ENTITY
`<!ATTLIST imagem fonte ENTITY #REQUIRED>`
- Uso no XML:
`<imagem fonte="logo">`

Entidades de parâmetro

```
<!ENTITY % nome "valor">
```

- Variáveis para uso dentro do DTD
- Exemplos. Em vez de repetir ...
`<!ATTLIST voo de (REC|CGH|GRU|GIG|SDU)>`
`<!ATTLIST voo para (REC|CGH|GRU|GIG|SDU)>`
- Declare as entidades
`<!ENTITY % aerosp "CGH|GRU">`
`<!ENTITY % aerorio "GIG|SDU">`
- Use as entidades no DTD
`<!ENTITY % aeros "REC|%aerorio;|%aerosp;">`
`<!ATTLIST voo de (%aeros;) #REQUIRED >`
`<!ATTLIST voo para (%aeros;) #REQUIRED >`

Entidades de parâmetro externas

- Carregam trechos de DTD de outros arquivos
`<!ENTITY % nome SYSTEM "uri">`
 - Exemplo
`<!ENTITY % tabela SYSTEM "voos.dtd">`
 - É preciso chamar a entidade dentro do DTD.
 - Uso
 - Resultado
- | | |
|-----------------------------------|--|
| <code><!ENTITY ...></code> | <code><!ENTITY ...></code> |
| <code>%tabela;</code> | <code><!ENTITY % sp "CGH GRU"></code> |
| <code><!ELEMENT ...></code> | <code><!ENTITY % rio "GIG SDU"></code> |
| <code>...</code> | <code><!ELEMENT ...></code> |
- Conteúdo de voos.dtd:
`<!ENTITY % sp "CGH|GRU">`
`<!ENTITY % rio "GIG|SDU">`

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

75

Condicionais

- Servem para construir DTDs configuráveis
 - Ignora o conteúdo
- ```
<![IGNORE [
 <!ELEMENT detalhes (#PCDATA)>
]]>
```
- Declara que o conteúdo deve ser interpretado
- ```
<![INCLUDE [  
  <!ELEMENT detalhes (#PCDATA)>  
]]>
```
- Parecem inúteis, não?

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

76

Utilidades dos Condicionais

- *Um DTD com declarações "desligadas"*

```
<!ENTITY % define.mod.um "IGNORE">
<!ENTITY % define.mod.dois "IGNORE">
<![%define.mod.um [
  <!ELEMENT coisas (#PCDATA)>
]]>
<![%define.mod.dois [
  <!ELEMENT maiscoisas (#PCDATA)>
]]>
```

- *Na utilização, pode-se ligar as declarações:*

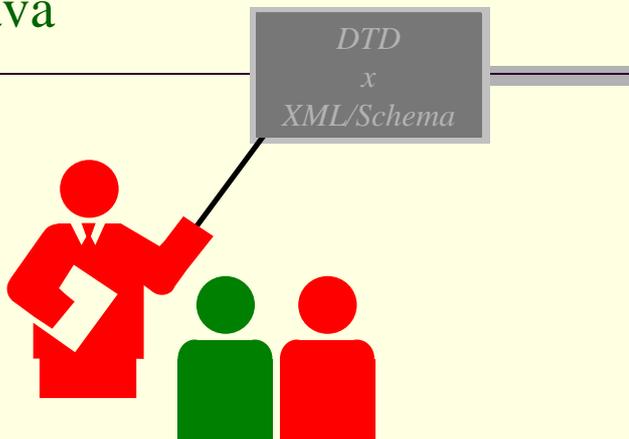
```
<!DOCTYPE raiz SYSTEM "coisas.dtd" [
  <!ENTITY % define.mod.um "INCLUDE">
]>
<raiz>
  <coisas> ... </coisas>
```

Namespaces e XML Schema

- *DTDs validam "documentos", não "linguagens"*
 - *DTDs supõem que documentos contém apenas uma linguagem*
- *DTDs não suportam namespaces. Para declarar um elemento é preciso usar o nome qualificado:*

```
<!ELEMENT prefixo:elemento (#PCDATA) >
```
- *Para validar namespaces em vez de documentos, use XML Schema*
 - *Cada namespace pode ser associado a um esquema diferente (um documento pode ter vários esquemas)*
 - *DTDs ainda podem ser usados, para declarar, por exemplo, entidades.*

POO-Java



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

79

XLink

- **Namespace:** `http://www.w3.org/1999/xlink`
- **Sete atributos:**
 - `type` = `simple` | `extended` | `locator` | `arc` | `title` | `resource`
 - `href` = *a URI destino do vínculo*
 - `show` = `new` | `replace` | `embed` | `other` | `none`
 - `actuate` = `onLoad` | `onRequest` | `other` | `none`
 - `title` = *descrição detalhada*
 - `role` = *papel / contexto do vínculo*
 - `label` = *descrição sucinta (para exibição)*
- **Exemplo de uso típico**

```
<elemento xlink:href="http://lugar.com" />
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

80

Exemplos de XLink

```
<p xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="simple"
  xlink:href="#end"
  xlink:show="replace"
  xlink:actuate="onRequest">link</p>
```

Conteúdo será substituído...
.. quando o usuário solicitar.

```
<img xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="simple"
  xlink:href="12345.jpg"
  xlink:show="embed"
  xlink:actuate="onLoad" />
```

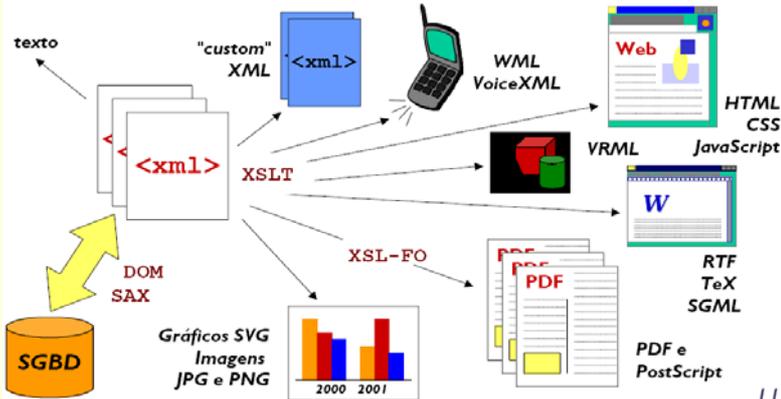
Conteúdo será embutido no documento...
.. quando o documento for carregado.

XPointer

- *Função que recebe uma expressão XPath*
 - *Serve para apontar para um recurso ou parte do documento*
- *Para apontar para um recurso descoberto por uma expressão XPath*
xpointer (expressão)
- *Quando há campos marcados com ID no documento destino, pode-se usar um ponteiro*
#ponteiro
xpointer (expressão) #ponteiro

Onde usar XML?

- *Dados armazenados em XML podem ser facilmente transformados em outros formatos*



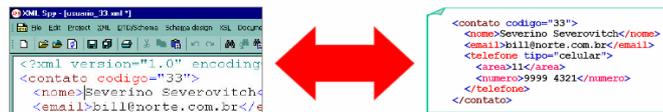
April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

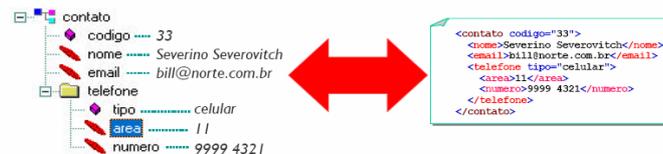
83

Como produzir XML

- **Criando** um documento de texto Unicode a partir de qualquer editor de textos



- **Gerando** um documento a partir de uma árvore montada dinamicamente



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

84

Documentos XML bem formados

- **Para que possa ser manipulado como uma árvore, um documento XML precisa ser bem formado**
 - Documentos que não são bem formados não são documentos XML
- **Documentos bem-formados obedecem as regras de construção de documentos XML genéricos**
- **Regras incluem**
 - Ter um, e apenas um, elemento raiz
 - Valores dos atributos estejam entre aspas ou apóstrofes
 - Atributos não se repetirem
 - Todos os elementos terem etiqueta de fechamento
 - Elementos estejam corretamente aninhados

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

85

XML válido

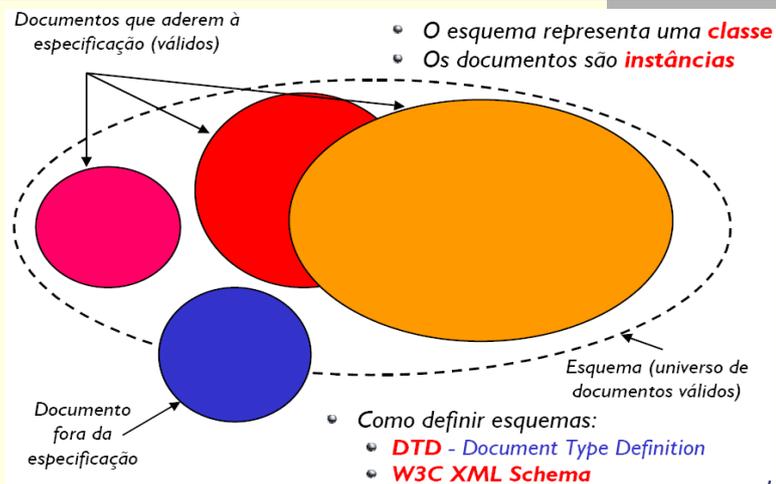
- **Um XML bem construído pode não ser válido em determinada aplicação**
- **Aplicação típica pode esperar que**
 - elementos façam parte de um **vocabulário limitado**,
 - certos atributos tenham **valores** e **tipos** definidos,
 - elementos sejam organizados de acordo com uma determinada estrutura **hierárquica**, etc.
- **É preciso especificar a linguagem!**
 - **Esquema**: modelo que descreve todos os elementos, atributos, entidades, suas relações e tipos de dados
- **Um documento XML é considerado válido em relação a um esquema se obedecer todas as suas regras**

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

86

Esquemas XML



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

87

DTD vs. XML Schema

- Um esquema é essencial para que haja **comunicação usando XML**
 - Pode ser estabelecido "informalmente" (via software)
 - Uso formal permite validação usando ferramentas genéricas de manipulação de XML
- Soluções

DTD

```
<!ELEMENT contato
(nome, email, telefone)>
<!ATTLIST contato
codigo NMTOKEN #REQUIRED>
```

Simple mas não é XML
Não suporta namespaces
Limitado quando a tipos de dados

XSchema

```
<xsd:schema
xmlns:xsd=".../XMLSchema">
<xsd:element name="contato">
<xsd:complexType>
<xsd:attribute name="codigo"
use="required">
```

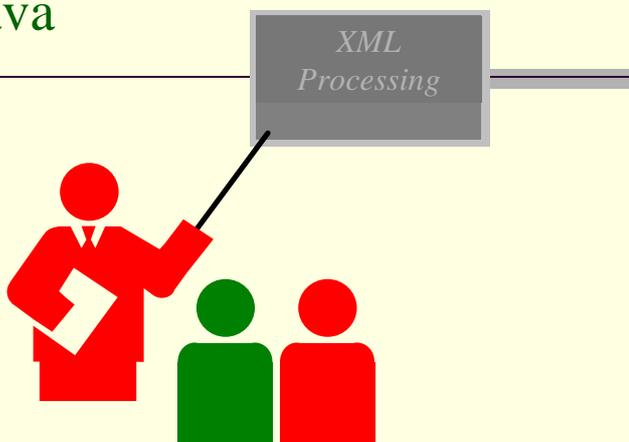
É XML, porém mais complexo
Suporta namespaces
Permite definição de tipos

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

88

POO-Java



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

89

Visualização em um browser

- **Folha de estilo:** conjunto de regras para formatar ou transformar as informações de um documento XML
- **CSS - Cascading Style Sheets**
 - Transformação visando apresentação visual
 - Aplicação do estilo em tempo de execução no cliente

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

90

Visualização em um browser

- **XSLT - eXtensible Stylesheet Language**
 - Transformação em texto, HTML ou outro formato
 - Aplicação em tempo real ou prévia (no servidor)
- **Se não estiver associado a uma folha de estilo, o documento XML não tem uma "aparência" definida**
 - Internet Explorer e outros mostram a árvore-fonte XML
 - Netscape mostra apenas os nós de texto

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

91

Como manipular XML?

- **Há duas APIs padrão para manipular (interpretar, gerar, extrair dados e tratar eventos) arquivos XML:**
 - W3C Document Object Model (**W3C DOM**)
 - Simple API for XML (**SAX**)
- **Servem a finalidades diferentes**
- **Implementações disponíveis em várias linguagens**
- **SAX oferece métodos que respondem a eventos produzidos durante a leitura do documento**
 - notifica quando um elemento abre, quando fecha, etc.
- **DOM monta uma árvore, que permite a navegação na estrutura do documento**
 - propriedades dos objetos podem ser manipuladas

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

92

Leitura de XML com SAX

- Se um processador SAX receber o documento ...

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

- ... ele irá disparar a seguinte seqüência de 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

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

93

Criação de documentos com DOM (1)

- Criação dos elementos

<code>/</code>	Document	Obter objeto do tipo Document (raiz) (dependente de processador): <code>doc</code>
<code><carta></code>	Element	<code>carta = doc.createElement("carta")</code>
<code><mensagem></code>	Element	<code>mens = doc.createElement("mensagem")</code>
<code>Bom dia!</code>	String	<code>texto = doc.createTextNode("Bom dia!")</code>

- Atributos

<code><mensagem id="1"></code>		<code>mens.setAttribute("id", "1")</code>
--------------------------------------	--	---

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

94

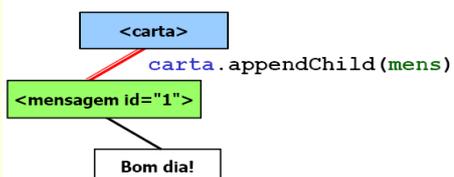
Criação de documentos com DOM (2)

Montagem da árvore passo-a-passo

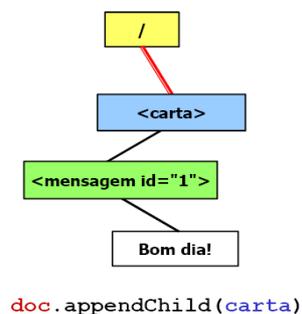
1. Sub-árvore <mensagem>



2. Sub-árvore <carta>



3. Árvore completa



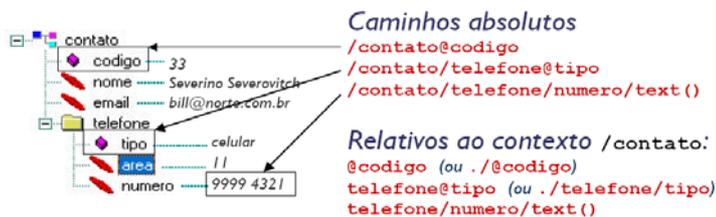
April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

95

XPath

- Linguagem usada para **navegar** na árvore XML
 - Uma **expressão XPath** é um caminho* na árvore que resulta em um valor (número, texto, booleano), objeto (elemento, atributo, nó de texto) ou conjunto de objetos



- Expressões XPath são usadas dentro de **atributos** XML
 - Usadas em XSLT, XLink, XQuery e XPointer

* pode também ser padrão de busca

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

96

XSLT

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="livro/titulo">
<td><xsl:value-of select="." /></td>
```

• XSL Transformations

- Linguagem (XML) para criação de documentos que contêm regras de transformação para documentos XML
- Documentos escritos em XSLT são chamados de **folhas de estilo** e contêm
 - Elementos XSLT: <template>, <if>, <foreach>, ...
 - Expressões XPath para localizar nós da árvore-fonte
 - Texto ou XML a ser gerado no documento-resultado
- Processador XSLT

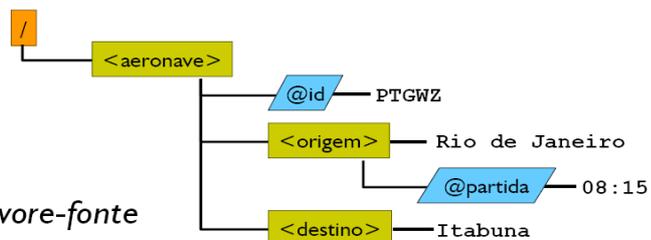


XSLT: documento-fonte (1)

• Considere o seguinte documento-fonte:

```
<aeronave id="PTGWZ">
  <origem partida="08:15">Rio de
    Janeiro</origem>
  <destino>Itabuna</destino>
</aeronave>
```

• Árvore-fonte



XSLT: folha de estilos (2)

- O seguinte **template** (parte de uma folha de estilos XSLT) pode extrair os dados do documento-fonte

```
<xsl:template match="aeronave">
  <p>A aeronave de prefixo
  <xsl:value-of select="@id" /> decolou
  de <xsl:value-of select="origem" /> às
  <xsl:value-of select="origem@partida" />
  tendo como destino o aeroporto de
  <xsl:value-of select="destino" />.</p>
</xsl:template>
```

documento-fonte (em azul - XPath)

documento resultado (em preto)

elementos XSLT (em vermelho, com prefixo xsl)

- Elementos XSLT geralmente são usados com um **prefixo** associado ao seu **namespace**: `<xsl:elemento>` para evitar conflitos com o documento-resultado.

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

99

XSLT: documento-resultado (3)

- Após a transformação, o resultado será

```
<p>A aeronave de prefixo
PTGWZ decolou
de Rio de Janeiro às
8:15
tendo como destino o aeroporto de
Itabuna.</p>
```

- Para obter outros resultados e gerar outros formatos com os mesmos dados, deve-se criar folhas de estilo adicionais

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

100

XLink, XPointer e XQuery

- **XLink**: é uma especificação W3C que permite definir vínculos entre documentos XML
 - Funcionalidade mínima é igual ao `<a href>` do HTML
 - Funcionalidade estendida permite vínculos bidirecionais, arcos, vários níveis de semântica, etc.
 - É uma coleção de atributos, com namespace próprio, que podem ser usados em elementos de qualquer linguagem XML.
- **XPointer**: aponta para partes de documentos XML
 - Identificador (ID) colocado no destino, acessível através de fragmento de URL: `xlink:href="#identificador"`
 - Caminho resultante de expressão XPath: `xpointer(/livro/id)`
- **XQuery**: linguagem para pesquisar documentos XML
 - Exemplo:

```
FOR $b IN document("usuario_33.xml")/contato
WHERE nome="Severino Severovitch"
RETURN $b
```

April 05

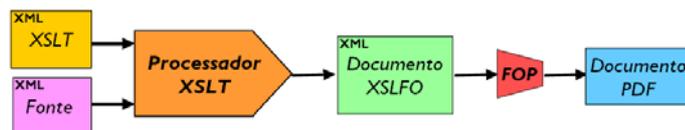
Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

101

XSL-FO

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="p1">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
</fo:root>
```

- **XSL Formatting Objects**
 - Linguagem XML de **descrição de página** com os mesmos recursos que PostScript ou PDF
 - Descreve o **layout preciso** de texto e imagens
 - Possui centenas de elementos, atributos e propriedades (que são semelhantes às propriedades do CSS)
 - Páginas são facilmente convertidas para PDF e PostScript
 - Ideal para gerar documentos para impressão (livros, etc.)
- Normalmente **gerada** via XSLT



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

102

XSL-FO: menor documento

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="p1">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-name="p1">
    <fo:flow flow-name="xsl-region-body">
      <fo:block color="blue" font-size="20pt">
        Hello PDF!
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Este é o "<head>" do XSL-FO

Ligação entre as regras de layout e o conteúdo afetado

Este é o "<body>" do XSL-FO

29

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

103

XHTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>Página XHTML</title></head>
  <body>
    <h1>Página XHTML</h1>
  </body>
</html>
```

• eXtensible HTML

- Linguagem XML de **descrição de página Web**
- Mesmos elementos do HTML 4.0 Strict
- Elementos descrevem **somente a estrutura** dos componentes da página.
 - **A forma precisa ser especificada usando CSS**: não há elementos/atributos para mudar cor, alinhamento, etc.
- Pode ser misturada (estendida) com outras linguagens XML (MathML, SVG, linguagens proprietárias)

• Normalmente **gerada** via XSLT



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

104

SVG

```
<svg>
<circle style="fill: red" cx="3cm" cy="3cm" r="2.5cm" />
<rect style="fill: blue" x="6cm" y="6cm"
height="2.5cm" width="1.5cm" />
</svg>
```

• W3C Scalable Vector Graphics

- Gráficos vetoriais em XML
- Plug-ins para principais browsers: concorre com Flash
- Suporta animações, links, JavaScript, CSS
- Produzido por ferramentas como Adobe Illustrator
- Pode ser embutido no código XHTML e XSL-FO



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

105

Exemplo SVG

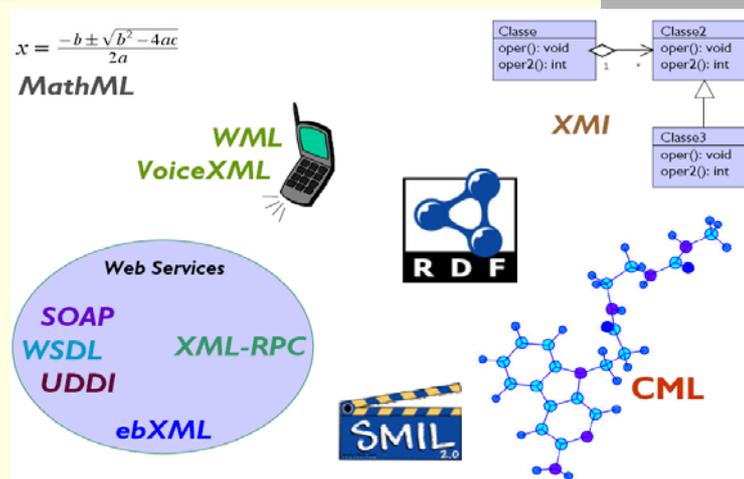
```
<svg width="10cm" height="10cm">
<g onclick="alert('Clicou no grupo 1!')">
<circle style="fill: red" cx="3cm" cy="3cm" r="2.5cm" />
<rect style="fill: blue" x="6cm" y="6cm"
height="2.5cm" width="1.5cm" /></g>
<g onclick="alert('Clicou no grupo 2!')">
<circle style="fill: green; opacity: 0.5"
cx="5cm" cy="5cm" r="2cm" /></g>
<a xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://www.w3.org/Graphics/SVG">
<text style="color: black; font-family: tahoma;
font-size: 12pt" x="3cm" y="8cm">
SVG é XML</text></a>
</svg>
```

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

106

Algumas outras linguagens XML



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

107

Ferramentas XML

- **Para programação**
 - **Parsers-validadores:** Xerces, Crimson, MSXML 4.0
 - **Validadores:** MSV (Sun)
 - **Transformadores XSL:** TrAX, Xalan, Xt, Saxon
 - **APIs:** JDOM, JAX (P, B, M, R, RPC), SAX e DOM
 - **Veja mais em** xml.apache.org e www.alphaworks.ibm.com
- **Para edição (de XML genérico)**
 - XML Spy Suite
 - Framemaker / ArborText
 - JEdit com plug-ins para XML, XSLT e XPath
 - **Veja mais em** www.w3.org/XML/

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

108

Conclusões

- XML é uma ótima solução para **compartilhar** dados
- Para **implementar** soluções em gestão de informações usando XML, pode-se usar
 - **DTD** ou **XSchema** para especificar o modelo de dados e validar as informações
 - As APIs **DOM** ou **SAX** para extrair dados dos documentos, gerar documentos, ler e gravar em bancos de dados
 - **XSLT** e **XPath** para transformar os dados em outros formatos
 - **XLink**, **XPointer** e **XQuery** para criar vínculos lógicos entre os documentos e localizar seus componentes
 - **XSL-FO** ou **XHTML** para formatar os dados para impressão ou visualização na tela (PDF, Word ou Web)
 - **SVG** para gerar informações em forma de gráfico vetorial