# eXtensible Markup Language

## © 2001–2005 Devendra Tewari

## Versão 1.1

## Última Atualização – 09-FEV-2005

# Conteúdo

### Tópico 1 – XML bem Formada – Sintaxe XML

### Tópico 2 – XML Válida – DTD e Esquema

### Tópico 3 – Acesso Programático – DOM e SAX

### Tópico 4 – Transformando XML – CSS e XSL

# Bibliografia

## Livros

#### Professional XML – Wrox e Ciência Moderna – Várias Autores.

#### XML Conceitos e Aplicações – Benoît Marchal – QUE e Berkeley.

#### Aprendendo XML – O'Reilly e Campus – Erik T. Ray.

## Sites

#### Site do W3C – http://www.w3.org.

#### Oasis – http://www.oasis-open.org.

# Tópico 1

## Sintaxe da XML – XML Bem Formada

# O que é XML?

## XML - eXtensible Markup Language

## Baseada em marcadores (tags) como HTML

## Por Exemplo:

## <livros>

## <livro nome="Aprenda Java"></livro>

## <livro nome="Aprenda XML"></livro>

## </livros>

# Porque XML?

## XML é uma HTML melhorada?

## Nenhum marcador predefinido

#### Extensível

## Sintaxe mas rígida

#### Fácil validar

# Quem controla XML?

## É um padrão aberto

## Padronizada pelo World Wide Web Consortium (W3C)

## Evoluiu da SGML (padrão da ISO - ISO8879)

## Recomendação do W3C desde Fevereiro de 1998

# Uso de XML

## Armazenagem de Documentos

## Intercâmbio de Dados

#### Entre base de dados e aplicação

#### Entre aplicações

## Armazenagem de Dados

#### Base de dados

#### Arquivos de configuração

# Outros Padrões Relacionados a XML

## Espaços identificadores (namespaces)

## XPATH

## Folhas de Estilo (style-sheets)

#### CSS

#### XSL

## DOM e SAX

## XLink e XPointer

# Trabalhando com XML

## Browser XML

## Editor XML

## Parser XML

## Processador de XSL

# Estrutura de um Documento XML

## A declaração “xml”

## Elementos

## Atributos

## Comentários

# Entidades XML

## Usadas para criar dados predefinidos

## A ocorrência de uma entidade no documento XML é substituída pelo seu valor pelo parser

## A ocorrência começa com “&” e termina com “;”

## A XML utiliza entidades para representar alguns caracteres reservados pela linguagem, como “>” (&gt;) e “<” (&lt;)

## Novas Entidades podem ser criadas usando uma DTD

# Atributos Especiais

## xml:space

#### Preservar ou não espaços em branco duplicados.

## xml:lang

#### Linguagem natural do valor texto de um elemento.

# Instruções de Processamento

## Podem ser usadas para associar arquivos CSS e XSL

## Iniciam com “<?” e terminam com “?>”

## Depois de “<?” aparece a aplicação destinatária

## Depois da aplicação aparece o texto da instrução

## Exemplo:

## <?xml-stylesheet href="teste.xsl" type="text/xsl"?>

# Seções CDATA

## CDATA significa dados caractere

## É usada para colocar texto qualquer como scripts de código (como VBScript ou JScript) dentro da XML

## É delimitada pelo “<![CDATA[” e “]]>”

## Uma seção CDATA não pode ser aninhada dentro da outra

# XML Bem Formada

## Condições:

### Apenas um elemento raiz

### Elementos sem sobreposição

### Início e fim de elementos corretamente demarcado

### Nome de elementos e atributos corretamente formado

### Valor de atributo demarcado com aspas duplas

### Texto caractere sem caracteres reservados ou inválidos

# Projetando Documentos XML

## Identificar nomes para coisas e conceitos

## Identificar hierarquia das coisas

### Identificar os relacionamentos entre coisas

## Definir propriedades das coisas

## Isso é modelagem orientado objeto

### Documentos XML podem ser gerados exportando uma hierarquia de objetos

# Atributos ou Elementos Filhos

# Exercício 1

## Como utilizar XMLSpy?

## Crie um catálogo de livros usando XML

#### Sugira um modelo para o documento

#### Crie o documento usando o modelo

# Tópico 2

## XML Válida – DTD e Esquemas

# Porque Usar DTD?

## DTD significa Definição do Tipo de Documento

## XML bem formada não basta

#### Precisa haver uma forma de definir o vocabulário do documento XML com precisão

#### Parsers XML podem usar a DTD para validação de uma instância do documento XML

#### Editores de XML (como XMLSpy) podem usar a DTD para fazer valer as regras contidas nele na hora de editar um documento XML

# Associando DTD com Documento XML

## Usando a instrução de processamento DOCTYPE

## Duas formas

### DTD Interna

#### DTD fica dentro do documento XML

### DTD Externa

#### DTD fica num arquivo separado

#### Facilmente aplicada a vários documentos XML

# Construtos de DTD

## Sintaxe da DTD não utiliza XML

## Uma DTD contém os construtos a seguir:

#### ELEMENT

#### ATTLIST

#### ENTITY

#### NOTATION

# Declaração ELEMENT

## Declarar um novo tipo de elemento XML

#### “<!ELEMENT” seguido pelo nome do elemento

#### Nome tem que ser identificador XML válido

#### Nome seguido pela definição de conteúdo

## Exemplos:

### 1. <!ELEMENT livro (#PCDATA)>

### 2. <!ELEMENT autor (titulo, nome)>

### 3. <!ELEMENT autor ANY>

# Declaração ELEMENT – Modelo de Conteúdo

## Para especificar conteúdo dentro de um elemento

## Contém uma combinação de:

#### Texto – indicado usando #PCDATA

#### Elementos filhos

#### Operadores

## Exemplos:

## 1. <!ELEMENT livro (#PCDATA | autor)\*>

## 2. <!ELEMENT livro (autor+, custo?)>

## 3. <!ELEMENT livro (custo, (autor | editora)+)>

# Declaração ATTLIST

## Atributos são propriedades de um elemento

## Declarados usando a sintaxe

### <!ATTLIST elemento atributo tipo uso>

## Exemplo

### <!ATTLIST autor

### primeiro CDATA #REQUIRED

### ultimo CDATA #REQUIRED

### conveniado CDATA #FIXED "sim"

### >

# Atributos tipo ID, IDREF e IDREFS

## ID é usado para definir um nome único no documento

#### Uso de um ID é sempre #REQUIRED

## IDREF refere-se a um ID declarado no documento

#### Usado para criar relacionamentos um para um

## IDREFS refere-se a vários ID ao mesmo tempo

#### Usado para criar relacionamentos um para vários

## Exemplo

## <!ATTLIST livro nome ID #REQUIRED>

## <!ATTLIST livro tipo IDREF #REQUIRED>

# Atributos tipo ENTITY e ENTITIES

## Apontam para Entidades definidas no mesmo documento usando declaração ENTITY

## ENTITIES é uma lista de ENTITY separadas por espaços

## Exemplo

### <!ENTITY textoCopyright "(c) Livraria Cataloga">

### <!ATTLIST livro

### copyright ENTITY #IMPLIED

### >

### O documento XML vai conter

### <livro copyright="textoCopyright">

# Atributos tipo NMTOKEN e NMTOKENS

## NMTOKEN = Fichas Nomeadas

## Os atributos desse tipo são atribuídos fichas válidas

## O processamento e checagem de validade das fichas fica por conta da aplicação

## Exemplo

## <!ELEMENT livro EMPTY>

## <!ATTLIST livro

## nome CDATA #REQUIRED

## tipo IDREF #REQUIRED

## autores IDREFS #REQUIRED

## copyright ENTITY #IMPLIED

## editora NMTOKEN #REQUIRED

## >

# Atributo tipo NOTATION

## NOTATION significa Notação

## O atributo desse tipo é usado em conjunto com a declaração NOTATION (notação)

## Uma notação declara um formato e associa um aplicativo externo para processar o formato

## Exemplo

## <!NOTATION jpg SYSTEM "jpgviewer.exe">

## <!NOTATION gif SYSTEM "gifviewer.exe">

## <!ELEMENT livro (imagem)>

## <!ELEMENT imagem (#PCDATA)>

## <!ATTLIST imagem

## tipo NOTATION (gif | jpg) "gif"

## >

# DTD Interna

## Elementos declarados na DTD interna sobrepõem elementos da DTD externa

## DTD interna é pouca utilizada porque:

#### Ocupa espaço em cada documento

#### Difícil mudar por estar em vários documentos XML

# Exemplo Escola

## Criar DTD para representar dados de uma escola

## Coisas

#### Escola

#### Estudante

#### Professor

#### Aula

# Limitação de DTD

## Difícil de escrever e entender

## Difícil processar programaticamente

## Difícil de estender

## Sem suporte para espaços identificadores (namespace)

## Nenhum suporte para tipos de dados

## Nenhum suporte para herança

# Exercício 2

## Crie uma DTD para o documento XML de catálogo de livros criado no Exercício 1

# O que é Esquema XML?

## Um esquema XML contém

#### Definição de tipos

#### Declaração de elementos

## Permite

#### Validar elementos e atributos XML e seus valores

#### Estender um esquema existente

## É uma recomendação do W3C

# Componentes de Esquema – Componentes Primários

## Definição de Tipos Simples

## Definição de Tipos Complexos

## Declaração de Atributos

## Declaração de Elementos

# Componentes de Esquema – Componentes Secundários

## Declaração de Notações

## Definição de Grupos de Atributos

## Definição de Restrições de Identidade

## Definição de Grupos de Modelos

# Componentes de Esquema – Componentes Auxiliares

## Anotações

## Grupos de Modelos

## Partículas

## Curingas (wildcards)

## Uso de Atributos

# Estrutura do Arquivo de Esquema

## O arquivo tem a extensão xsd

## Contém um elemento raiz chamado “schema”

## Elemento “schema” pode conter atributos:

#### attributeFormDefualt

#### elementFormDefault

#### id

#### targetNamespace

#### version

#### xmlns (mais de uma ocorrência)

# Associando Esquema a um Documento XML

## Um documento XML pode ser considerado como uma instância de um esquema

## O elemento raiz do documento XML deve declarar o espaço identificador ao qual ele pertence e isso deve corresponder ao espaço identificador especificado pelo atributo “targetNamespace” no esquema

## O elemento raiz deve declarar o espaço identificador “http://www.w3.org/2000/10/XMLSchema-instance” e atribuir para o atributo “schemaLocation” deste espaço identificador o nome do arquivo contendo o esquema

# Espaço Identificadores (namespaces)

## Evitam ambigüidade e colisão de nomes quando um elemento XML utilizar dois esquemas diferentes

## O elemento declara a utilização de um esquema usando o atributo “xmlns” ou a forma “xmlns:prefixo”

## O prefixo é utilizado no nome de todos os elementos declarados no esquema

## O espaço identificador é identificado utilizando um URI (identificador universal de recursos)

## Geralmente o URI é um URL

# Definição de Tipo Simples

## Pode ser usada na declaração de:

#### Atributos

#### Elementos simples contendo apenas texto

## Por Exemplo

## <xs:simpleType name="tipoTemperaturaAgua">

## <xs:restriction base="xs:number">

## <xs:minExclusive value="0.00"/>

## <xs:maxExclusive value="100.00"/>

## </xs:restriction>

## </xs:simpleType>

# Restringindo Valores Usando Expressões Regular

## Podemos restringir valores utilizando expressões regular (regular expressions)

## Uma expressão regular é especificada utilizando o elemento “xs:pattern” dentro do elemento “xs:restriction”

## Demonstrado embaixo utilizando o valor do CPF como exemplo

### Observe o uso do atributo “memberTypes” dentro do elemento “xs:union”

# Definição de Tipo Complexo

## Usada para declarar elementos contendo:

#### Atributos

#### Tipo do Conteúdo (elementos filhos)

## Tipos complexos podem estender outros tipos simples ou complexos

# Declaração de Atributos

## Uma associação entre um nome e um tipo simples

## Pode conter um valor padrão

# Declaração de Elementos

## É uma associação entre um nome e tipos simples ou complexos

## Pode está contida dentro de uma definição de tipo complexo

## Equivalente as declarações ELEMENT e ATTLIST do DTD

# Declaração de Notações

## É uma associação entre um nome e um identificador de notação

## Uma notação serve para identificar conteúdo não XML e o programa que processa o conteúdo

# Definição de Grupos de Atributo

## É usada para reutilizar grupos de atributos em várias definições de tipos complexos

# Definição de Restrições de Identidade

## É uma associação entre um nome e qualquer tipo de restrição como chave primária, chave única ou chave estrangeira

## Utiliza a especificação XPATH para indicar os registros aos quais se aplica a restrição

# Definição de Grupos de Modelos

## É uma associação entre um nome e um grupo de modelo

## Facilita a reutilização do mesmo grupo de modelo em vários tipos complexos

# Anotações

## Contém informação sobre o modelo

## Usadas por leitores humanos ou máquinas

# Grupos de Modelos

## Compostos por uma lista de

#### Elementos

#### Curingas e

#### Grupos de Modelos

## Temos três tipos de grupos de modelos

#### Seqüência

#### Conjunção

#### Disjunção

# Curingas

## O elemento “xs:any” fornece um mecanismo para introduzir um curinga para o conteúdo dos elementos

## O elemento “xs:anyAttribute” é um curinga para qualquer atributo do espaço identificador especificado no atributo “namespace” do elemento

# Uso de Atributos

## Usado para especificar se a utilização de qualquer atributo é:

#### Requerido

#### Opcional

#### Proibido

## O atributo pode ter um valor fixo (constante)

# Tipos de Dados Internos

# Tipos Derivados Internos

# Exemplo Escola

## Mesmo exemplo como no caso da DTD

## Esquema permite maior flexibilidade:

### Definição das chaves:

#### Única

#### Primária

#### Estrangeira

### Definição e extensão de tipos simples e complexos

# Exercício 3

## Crie um esquema XML para a DTD de catálogo de livros criada no Exercício 2

## Estude tópicos avançados:

### Expressões Regular

### Tipos abstratos

### Redefinir esquema - elemento xs:redefine

### Importando esquemas - elemento xs:include

### Derivando tipos - elementos xs:extension e xs:restriction

### Restringindo derivação de tipos - atributos final e block do elemento xs:complexType

### Formas de restringir valores de tipos simples (facets)

# Tópico 3

## Acesso Programático - DOM e SAX

# DOM – Document Object Model

## Representa todo conteúdo do documento XML como uma árvore

## Cada elemento, texto do elemento, atributo, comentário, instrução de processamento etc. é representado como um nó nesta árvore

## É uma especificação recomendada pelo W3C

# Objetos do DOM

## DOM composto de vários tipos de objetos para representar documento XML, elemento, atributo etc

## Existem objetos para representar e agregar nó:

#### Node: Um único nó do documento

#### NodeList: Apresenta uma lista de nó

#### NamedNodeMap: Fornece acesso aos nós de atributos pelo nome

# Objeto Node

## Objeto Node representa um nó

## Cada objeto (Element, Attribute, etc.) herda características básicas do objeto Node

## O objeto Node disponibiliza várias propriedades para navegar a árvore de elementos

# Propriedade nodeType

## Contém valor numérico identificando o tipo do nó

# Métodos do Objeto Node

## Disponibilizados pelo objeto Node para inserir, duplicar, remover e substituir outros objeto Node

# Objetos NodeList e NamedNodeMap

## Objeto NodeList

#### Fornece uma lista de nós

#### É uma coleção indexada de nós começando com o índice 0

## Objeto NamedNodeMap

#### Fornece uma coleção de nós

#### Os nós podem ser acessados usando nomes

#### Os nós também podem ser acessados usando um índice numérico

# Objeto Attr

## Representa atributos

## Objeto Attr herda do objeto Node mas não faz parte da árvore de documento

#### Por isso atributos parentNode, previousSibling e nextSibling do objeto Node tem valor nulo

## Em MSXML 3.0 a interface é chamada de IXMLDOMAttribute

# Objeto Element

## Representa um elemento XML

## Herda do objeto Node

#### Tem todas as características e operações do objeto Node

## Contém métodos para recuperar objetos do tipo Attr pelo nome ou pelo índice

## Propriedades

#### tagName: Nome do elemento

# Objetos CharacterData e Text

## Objeto CharacterData

#### Representa dados caractere usando Unicode (UTF-16)

#### Estende o objeto Node

## Objeto Text

#### Representa dados texto dentro de um elemento.

#### Herda do Objeto CharacterData

# Objeto Document

## Representa um documento XML

## Herda do objeto Node

## Acrescenta algumas propriedades a mais

#### documentElement: Elemento raiz

#### doctype: Tipo do documento

## Acrescenta alguns métodos

## Métodos específicos do parser MSXML 3.0:

#### loadXML: Carregar texto XML

#### load: Carregar arquivo XML

# Usando DOM para criar Documentos XML

## Usaremos MSXML nos exemplos

## Nosso exemplo:

#### Utiliza o objeto DOMDocument para criar um novo documento XML

###### Insere uma nova instrução de processamento

###### Cria nó do tipo “NODE\_ELEMENT” e do tipo “NODE\_ATTRIBUTE”

# Usando DOM para ler Documentos XML

## Usaremos MSXML neste exemplo

## O objeto DOMDocument é usado para ler XML

#### O método load carrega um arquivo XML

## No exemplo ilustramos

#### Como tratar erros usando atributo parseError

#### Como ler todos os nós de um documento XML usando um método recursivo

# Resultado da Leitura

## Aqui mostramos a saída gerada pela leitura do documento XML

## O documento XML utilizado foi:

### <catalogo>

#### <livro nome="Programação Java">

##### <autor nome="Paulo Junior"/>

#### </livro>

### </catalogo>

# Exercício 4

## Use DOM para ler o catálogo de livros criado anteriormente e mostrar nome de todos os livros dentro de um controle como ListBox

# Simple API for XML

## Está baseada em eventos

## Não é um padrão do W3C

## É um padrão “de fato”

## A versão preliminar chamado de SAX

## A versão atual é chamado SAX2

### Oficialmente disponível para Java

### Tem suporte para espaços identificadores

# Lendo um Documento XML Usando SAX

## Utilizamos parser SAX do pacote MSXML

## Implementamos a interface “ContentHandler” para receber e tratar callbacks (eventos) SAX

## Usamos os eventos:

### startElement

### endElement

## Para executar o exemplo podemos chamar o método

### saxParse(nome do arquivo XML)

# Lendo um Documento XML Usando SAX

## Resultado da execução

### catalogo

### autores

### autor

### primeiroNome

### sobreNome

### ...

### livros

### livro

### autor

### primeiroNome

### sobreNome

# XML e Java

## O mundo Java tem diversos APIs e parsers XML

## APIs

#### JAXP da SUN

#### JDOM

## Parsers DOM

#### XML4J da IBM

#### Xerces do Apache

## Parsers SAX

#### Xerces do Apache

# Usando JAXP - DOM

## JAXP gera uma árvore DOM usando um parser SAX

## Pacotes JAXP utilizados no exemplo:

#### javax.xml.parsers

###### DocumentBuilder, DocumentBuilderFactory, ParserConfigurationException

#### org.w3c.dom

###### Document, Node, NamedNodeMap, NodeList

#### org.xml.sax

###### SAXException

# Usando JAXP – SAX

## JAXP vem com um parser SAX

#### Isso não impede a utilização de outro parser SAX qualquer

## Pacotes JAXP utilizados no exemplo:

#### javax.xml.parsers

###### SAXParser, SAXParserFactory, ParserConfigurationException

#### org.xml.sax

###### SAXException

#### org.xml.sax.helpers

###### DefaultHandler

# Exercício 5

## Utilize SAX para ler o catálogo de livros criado anteriormente e mostrar o nome de todos os livros dentro de um controle como ListBox

# JDOM

## Limitações do DOM:

#### Independente da Linguagem

#### Hierarquias estritas de classes

#### Baseado em Interface

## Vantagens da JDOM:

#### Para Java

#### Hierarquias de classes inexistentes

## Ainda na versão beta (consulte www.jdom.org)

# Método Pull em Microsoft .NET

## A API SAX utiliza callbacks para avisar quando elementos são encontrados

## O método Pull utiliza chamadas explícitas a um método que recupera um elemento após outro

## Implementada pela classe XmlTextReader do espaço identificador System.Xml

## O exemplo utiliza a linguagem C# da plataforma .NET

# Tópico 4

## Transformando XML – CSS e XSL

# Cascading Style Sheets (CSS)

## CSS significa Folhas de Estilo Em Cascata

## É utilizada com HTML

## Pode ser utilizada com XML

## Determina a formatação visual dos elementos

## Qualquer navegador com suporte a CSS pode formatar os elementos:

#### IE 5.x ou melhor

#### Netscape 6.x ou melhor

## Padrão atualmente no nível 2 (CSS2)

# Exemplo da CSS com XML

## Vamos abrir este documento XML em IE 5.0

### <?xml version="1.0" encoding="ISO-8859-1"?>

### <?XML:stylesheet type="text/css" href="artigo.css"?>

### <artigo>

### <titulo>Frederico o Grande encontra Bach</titulo>

### <autor>Johann Nikolaus Forkel</autor>

### <paragrafo>

### Uma noite, quando ele estava acabando de preparar a

### <instrumento>flauta</instrumento> e seus músicos estavam

### reunidos, um oficial trouxe a lista de convidados.

### </paragrafo>

### </artigo>

# Seletores

## Uma folha de estilo contém um conjunto de regras compostas de:

#### Seletores

#### Blocos de declarações

## Um seletor especifica um ou mais elemento XML

## Exemplo:

### artigo, titulo, autor, paragrafo { display: block }

#### Artigo, titulo, autor e paragrafo são seletores

#### A declaração CSS começa depois de { e termina com }

# Tipos de Seletores

## Seletor universal

## Seletor de tipo

## Seletor de descendente

## Seletor de filho (elemento filho)

## Seletor de irmão (elementos adjacentes)

## Seletor de atributo

## Seletor de classe

## Seletor de ID

# Pseudo-elementos e Pseudo-Classes

## Pseudo-classes

#### :first-child

#### :link e :visited

#### :hover, :active, e :focus

#### :lang

## Pseudo-elementos

#### :first-line

#### :first-letter

#### :before e :after

# Valor das Propriedades

## Propriedades estão contidas dentro de um bloco de declaração

## Basicamente têm os seguinte tipos de valores:

### Comprimento

### Porcentagem

### Cor

### URL

# Propriedades da Caixa

## Os elementos são pintados dentro de uma caixa

## Uma caixa padrão é um retângulo

## As propriedades da caixa são divididas nas seguinte categorias:

#### Display

#### Margem

#### Padding

#### Borda

# Propriedades de Texto e Fonte

## Propriedades de fonte atribuam:

#### Nome da fonte

#### Comprimento da fonte

#### Estilo e peso

#### Alinhamento do Texto

#### Endentação e Altura de Linha

# Propriedades da Cor e Fundo

## Podemos especificar propriedades para:

#### Cor do primeiro plano

#### Cor de fundo

#### Cor da borda

#### Imagem de fundo

# Importando uma Folha de Estilo

## Uma folha de estilo pode ser importada dentro da outra

## Regras especificadas no arquivo atual sobrepõem as regras do arquivo importado

## Utilizamos @import para fazer a importação

# Exercício 6

## Crie um documento CSS para mostrar o catálogo de livros dentro do browser

# eXtended Stylesheet Language

## CSS é muito básico

## Precisamos transformar

#### XML para XML

#### XML para HTML

#### Agrupar, filtrar, calcular e classificar

## Não estamos falando em abolir CSS

# Conceito de XSL

## O processo de usar XSLT:

### Ler documento XML

### Aplicar Transformações

### Gravar documento resultante

# Exemplo de XSLT

## Vamos aplicar transformações neste documento:

### <?xml version="1.0" encoding="UTF-8"?>

### <?xml-stylesheet type="text/xsl" href="catalogo.xsl"?>

### <catalogo>

### <livros>

### <livro nome="Professional XML"/>

### <livro nome="XML By Example"/>

### </livros>

### </catalogo>

# Execução do Exemplo

# XPATH

## Padrão base para XSLT

## Usado para selecionar elementos e atributos

## Trata documento como sistema de arquivos (elementos = pastas)

## Um caminho pode ser relativo ou absoluto

## Permite usar expressões lógicas e funções

# Controlando a Saída

## Podemos controlar a saída do processador XSLT usando o elemento xsl:output

## Podemos escolher a saída atribuindo os seguinte valores para o atributo method:

#### xml

#### text

#### html

## O elemento xsl:output deve aparecer logo após do marcador inicial do elemento xsl:stylesheet

# Definindo Templates

## Documento XSL contém pelo menos um elemento xsl:template

## O atributo match desse elemento seleciona o nó que será processado por esse elemento

## Ao atributo match deve ser atribuído uma expressão XPATH válida

## Dentro do elemento xsl:template podemos ter um elemento xsl:apply-templates para pedir o processador XSLT para aplicar os demais templates do documento

# Variáveis e Parâmetros

## Usando elementos xsl:variable e xsl:param

## Elemento xsl:variable

#### Associa um valor a um nome

#### O tipo do valor pode ser qualquer texto ou nó selecionado usando expressões XPATH ou constante

## Elemento xsl:param

#### Usado para passar parâmetros para uma folha de estilo ou template

#### Parâmetros podem ser passados usando o elemento xsl:with-param

# Lendo valores e Copiando Fragmentos do Documento

## Elemento xsl:value-of

#### Recupera valor de um nó ou variável

#### Atributo select usado para selecionar o nó

#### O valor recuperado é escrito para a saída

## Elemento xsl:copy

#### Copia nó atual para a saída

#### Não copia atributos e sub nós

## Elemento xsl:copy-of

#### Copia um nó, seus atributos e sub nós para a saída

# Processamento Condicional

## Elemento xsl:if

#### Expressão lógica atribuída ao atributo test

#### Não tem “Se Não”

## Elemento xsl:choose

#### Seleciona uma entre várias possibilidades

#### Possibilidades especificadas usando atributo test do elemento xsl:when

#### Elemento xsl:otherwise especifica o processamento caso nenhuma das possibilidades for válida

# Repetição

## Usando elemento xsl:for-each

## Atributo select é atribuído uma expressão XPATH que deverá retornar uma lista de nós

## Dentro do elemento xsl:for-each deverá ser definido o template a ser aplicado nos nós selecionados

# Classificação

## Usando elemento xsl:sort

## Pode aparecer dentro dos elementos xsl:apply-templates ou xsl:for-each

## Pode aparecer uma série de elementos xsl:sort

#### Isso permite classificar por várias colunas

## O valor do atributo select deve ser uma expressão XPATH apontando para um objeto válido

#### O valor padrão do atributo select é “.”

## O valor do atributo data-type pode ser “text” ou “number” indicando se o valor usado para classificação será string ou numérico

# Formatação de Números

## O elemento xsl:number pode ser usado para inserir um número formatado na saída

## O atributo value deve conter uma expressão

## O atributo format especifica o formato do número convertido para string

# Importando Folhas de Estilo XSL

## Elemento xsl:import

#### Importa o arquivo especificado no atributo href dentro do arquivo XSL atual

#### Os templates do arquivo importado têm uma precedência menor

#### Só pode aparecer dentro do elemento xsl:stylesheet e antes de qualquer outro elemento

## Elemento xsl:include

#### Igual ao elemento xsl:import

#### Templates do arquivo XSL importado têm a mesma precedência dos templates locais

# Chamando Templates Nomeados

## Usando elemento xsl:call-template

## O template chamado tem que ter o mesmo nome especificado no atributo nome do elemento

# Criando Elementos

## Usando elemento xsl:element

## O nome do elemento criado é especificado usando atributo nome

## O espaço identificador pode ser especificado usando atributo namespace

## Um conjunto de atributos pode ser especificado usando o atributo use-attribute-sets

# Criando Atributos

## Elemento xsl:attribute

#### Atributo name especifica o nome

#### Atributo namespace especifica o espaço identificador

## Elemento xsl:attribute-set para criar conjuntos

#### Contém uma série de elementos xsl:attribute

#### Atributo name especifica o nome do conjunto

#### Nome de um conjunto pode ser atribuído a atributo use-attribute-sets dos elementos xsl:element, xsl:copy ou outro xsl:attribute-set

# Criando Texto

## Usando elemento xsl:text

## Usado para colocar texto na saída

## Atributo disable-output-escaping especifica se o tratamento de saída deve ser desabilitado

# Criando Comentários e Instruções de Processamento

## Elemento xsl:comment

#### Para criar comentário

## Elemento xsl:processing-instruction

#### Para criar instrução de processamento

#### Pode ser usado para atribuir uma CSS ao documento de saída

# Exemplo – Criação de Menu

## Vamos colocar os dados representando o menu num arquivo XML como mostrado a seguir

## <?xml version="1.0" encoding="UTF-8"?>

## <?xml-stylesheet href="menu.xsl" type="text/xsl"?>

## <menu>

## <link url="1.asp" nome="1"/>

## <link url="2.asp" nome="2"/>

## <titulo nome="t1">

## <link url="3.asp" nome="3"/>

## </titulo>

## <titulo nome="t2">

## <link url="4.asp" nome="4"/>

## </titulo>

## </menu>

# Exemplo – Criação de Menu

## Resultado

## 1 2 t1 3 t2 4

## Melhorias

### Aninhar os itens do menu com títulos

### Permitir criar títulos dentro de títulos

### Gerar DHTML para fechar e abrir títulos

# Transformação usando JAXP

## Utilizando JAXP para transformar XML via programa

## Pacotes utilizados

#### javax.xml.parsers

#### org.xml.sax

#### org.w3c.dom

#### javax.xml.transform

#### javax.xml.transform.dom

#### javax.xml.transform.stream

#### java.io

# Transformação usando JAXP

## Classes utilizadas

### DocumentBuilderFactory e DocumentBuilder

#### Para ler o documento XML

### TransformerFactory, Transformer

#### Para transformar documento XML

### DOMSource, StreamSource, StreamResult

#### Entrada e saída do transformador

# Exercício 7

## Crie um documento XSL para gerar uma página HTML contendo uma tabela listando informações sobre os livros dentro do catálogo

# Estado de padrões do W3C

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1996 | 1997 | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 |
| CSS | 1.0 |  | 2.0 |  |  |  |  |  |  |
| DOM |  |  | Level-1 |  | Level-2 |  |  |  |  |
| HTML |  | 4.0 |  |  |  |  |  |  |  |
| MathML |  |  |  | 1.01 |  | 2.0 |  |  |  |
| NS |  |  |  | 1.0 |  |  |  |  |  |
| RDF |  |  |  | Parcial |  |  |  |  |  |
| SMIL |  |  | 1.0 |  |  | 2.0 |  |  |  |
| SOAP |  |  |  |  |  |  |  |  |  |
| SVG |  |  |  |  |  | 1.0 |  |  |  |
| XHTML |  |  |  |  | 1.0 |  |  |  |  |
| XLink |  |  |  |  |  | 1.0 |  |  |  |
| XML/DTD |  |  | 1.0 |  |  |  |  |  |  |
| XMLBase |  |  |  |  |  | 1.0 |  |  |  |
| XMLProtocol |  |  |  |  |  |  |  |  |  |
| XPATH |  |  |  | 1.0 |  |  |  |  |  |
| XPointer |  |  |  |  |  |  | 1.0 |  |  |
| XQuery |  |  |  |  |  |  |  |  |  |
| XS |  |  |  |  |  | 1.0 |  |  |  |
| XSL:FO |  |  |  |  |  | 1.0 |  |  |  |
| XSLT |  |  |  | 1.0 |  |  |  |  |  |