



XML – introdução XML – eXtensible Markup Language – é uma linguagem de marcadores como a HTML e foi desenhada para descrever dados , a sua grande vantagem é que ela é extensível , ou seja , você não está limitado a um certo número de tags , e pode criar as suas próprias tags, assim sendo ela é uma linguagem auto definível . Para descrever os dados a XML usa a DTD – Document Type Definition.

Antes que você fique confuso , a XML não é uma linguagem que veio para substituir a HTML pois XML foi criada com um objetivo diferente da HTML. Enquanto HTML foi criada para exibir dados e ela se preocupa como os dados serão exibidos, a XML foi criada para descrever dados e ela se preocupa com o que os dados são. HTML está relacionada com exibir dados enquanto que XML está relacionada em descrever dados.

As tags usadas em documentos HTML e a estrutura dos documentos HTML são predefinidas e o autor de um documento HTML pode usar somente as tags que estão definidas em HTML padrão. Já a XML permite que o autor defina as suas próprias tags e a própria estrutura do seu documento. A utilização de XML vem crescendo dia a dia , mas não devemos pensar que ela será a substituta da HTML , ela será um complemento a HTML e será usada para descrever e estruturar os dados enquanto HTML será usada para formatar e exibir os mesmos dados.

A sintaxe da XML

Vejamos um exemplo de documento XML (*vamos dar a ele o nome de aviso.xml*), e a seguir vamos analisar cada linha de código:

```
<?xml version="1.0"?>
<aviso>
<para>Janice data="01/04/2000"</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem prova de matematica</corpo>
</aviso>
```

Aviso.xml

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

Esta primeira linha do documento é uma declaração XML e deve sempre ser incluída pois define a versão XML do documento. Neste caso estamos especificando a versão 1.0 da XML. O Internet Explorer vem com

XML – introdução XML – eXtensible Markup Language – é uma linguagem de marcadores como a HTML e

um analisador XML que atende o padrão 1.0.

```
<aviso>
```

Esta linha define o primeiro elemento do documento – o elemento raiz.(nó raiz)

```
<para>Janice data="01/04/2000"</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem prova de matematica</corpo>
```

Estas quatro linhas definem 4 elementos filhos da raiz (*para, de, cabecalho e corpo*)

```
</aviso>
```

A última linha define o fim do elemento raiz.

Você de Ter estranhado as tags usadas no documento , pois é , elas foram criadas pelo autor; somente a primeira linha é obrigatória pois define a XML para o analisador do navegador.

Vejam as características principais de um documento XML:

- Perceba que cada tag inicial tem um tag final , ou seja , as tags são usadas sempre em pares. Você não pode usar: **<de>** Jefferson
- Todo documento XML possui um elemento raiz . Os demais elementos devem estar aninhados dentro de um elemento raiz.

```
<raiz>
  <filho1>
    <filho2>
    </filho2>
  </filho1>
</raiz>
```

- elemento raiz possui um atributo , a data de criação do documento; da mesma forma os elementos podem possuir atributos aos pares **nome/valor**, e , o atributo deve sempre vir entre aspas.
- As tags usadas em XML são *case sensitive* , ou seja , você deve escrever uma tag de fechamento da mesma maneira que escreveu uma tag de início. Assim **<para>** e **</Para>** irá ocasionar um erro no documento XML.

Usando atributos em XML

Os atributos em XML são usados para descrever os elementos XML ou para fornecer uma informação adicional sobre os elementos. Em HTML quando usávamos a seguinte linha :

**** tínhamos SRC como um atributo do elemento IMG que fornecia informação adicional sobre o elemento. Vejam o exemplo a seguir em XML:

```
Usando um atributo para o sexo:
<pessoa sexo="feminino">
  <nome>Ana</nome>
  <sobrenome>Rachel</sobrenome>
```

```
</pessoa>
```

Usando um elemento para o Sexo:

```
<pessoa>
  <sexo>feminino</sexo>
  <nome>Ana</nome>
  <sobrenome>Rachel</sobrenome>
</pessoa>
```

No primeiro exemplo **sexo** é um atributo ; no último exemplo sexo é um elemento. Não existe uma regra que indique quando você deve usar um ou outro, o que vale é o bom senso. Mas a utilização de atributos deve ser evitada se você pode descrever seus dados com elementos.

Vejamos outro exemplo para mostrar como os elementos podem ser mais efetivos que os atributos. Temos três exemplos de documentos XML que contém a mesma informação: No primeiro usamos um atributo para a data, no segundo usamos um elemento para a data, e no terceiro usamos um elemento expandido para data:

```
<?xml version="1.0"?>
<aviso date="12/11/99">
<para>Janice</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem prova de matematica</corpo>
</aviso>
```

```
<?xml version="1.0"?>
<aviso>
<data>12/11/99</data>
<para>Janice</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem prova de matematica</corpo>
</aviso>
```

```
<?xml version="1.0"?>
<aviso>
<data>
  <dia>12</dia>
  <mês>11</mês>
  <ano>99</ano>
</data>
<para>Janice</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem prova de matematica</corpo>
</aviso>
```

As principais desvantagens na utilização de atributos são:

- Atributos não podem conter múltiplos valores (os elementos podem)
- Atributos não são expansíveis
- Atributos não podem descrever estruturas
- Atributos são de difícil manutenção.

Criando um documento XML válido

Para que um documento XML seja um arquivo válido temos que usar o que chamamos de – Definição do Tipo do Documento – ou , originalmente – Document Type Definition – DTD.

XML – Introdução

O propósito da DTD é definir uma construção de blocos válidas para um documento XML, e ela define a estrutura do documento usando uma lista de elementos válidos. A DTD pode ser declarada dentro de um documento XML ou num arquivo à parte.

O DTD permite descrever cada marca e fornecer regras para interpretar cada informação usada em um arquivo XML. Quando usada em um arquivo XML a DTD aparece entre colchetes [e termina com um colchete , seguido de um sinal de maior (]>). Usando DTD em nosso arquivo aviso.xml temos o seguinte código:

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

<!DOCTYPE nota [

<!ELEMENT aviso (para,de,cabecalho,corpo)>

<!ELEMENT para (#PCDATA)>

<!ELEMENT de (#PCDATA)>

<!ELEMENT cabecalho (#PCDATA)>

<!ELEMENT corpo (#PCDATA)>

]>

<aviso>
<para>Janice data="01/04/2000"</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem prova de matematica</corpo>
</aviso>
```

Arquivo aviso.xml com definições DTD

Interpretando o código DTD usado temos:

<!DOCTYPE nota [– Declara um seção do documento com um DTD com o nome de nota

–**!ELEMENT aviso** – define o elemento "aviso" como tendo 4 elementos (para, de, cabecalho e corpo)

–**!ELEMENT para** – define o elemento "para" como sendo do tipo "CDATA"

–**!ELEMENT de** – define o elemento "de" como sendo do tipo "CDATA"

–**!ELEMENT cabecalho** – define o elemento "cabecalho" como sendo do tipo "CDATA"

–**!ELEMENT corpo** – define o elemento "corpo" como sendo do tipo "CDATA"

Obs: Um **ELEMENT** define a forma e os termos de uma marca XML usada.

Para usar a DTD em um arquivo externo basta fazer o seguinte:

1–Insira uma declaração informando o nome do arquivo externo onde estão as definições DTD

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

Criando um documento XML válido

```
<!DOCTYPE note SYSTEM "nota.dtd">

<aviso>
<para>Janice data="01/04/2000"</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>

<corpo>Amanha
```

Arquivo aviso.xml com declaração do arquivo DTD

2-) Crie o arquivo com as definições DTD e salve-o usando o nome declarado no arquivo XML . No nosso caso : nota.dtd

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

<!ELEMENT aviso (para,de,cabecalho,corpo)>

<!ELEMENT para (#PCDATA)>

<!ELEMENT de (#PCDATA)>

<!ELEMENT cabecalho (#PCDATA)>

<!ELEMENT corpo (#PCDATA)>
```

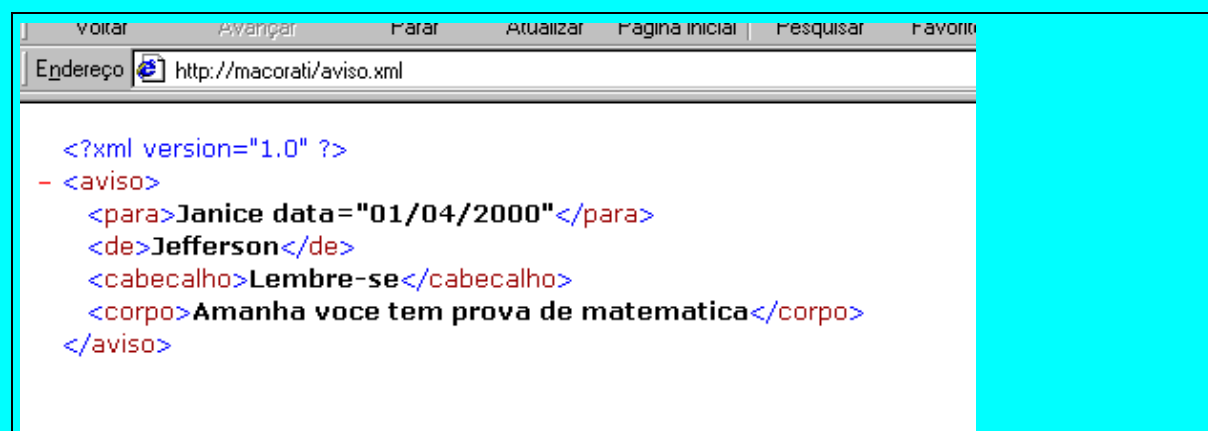
Nota.dtd

A DTD pode ser usada para compartilhar e intercambiar dados usando definições DTD comuns.

Exibindo o XML na WEB

Atualmente somente o Internet Explorer suporta o padrão XML 1.0 e o DOM . Esses padrões são definidos pelo consórcio W3C – *World Wide Web Consortium*.

Se você abrir o nosso primeiro arquivo exemplo (aviso.xml) usando o Internet Explorer irá obter o seguinte:



Nada amigável , não é mesmo ? Então como exibir os dados de um documento XML aos seus usuários na Web ? Bem , você pode fazer isto de duas maneiras:

- Convertendo XML usando o DOM

- Usando as folhas de estilo(XSL) do XML

1-) Exibindo XML usando DOM

O DOM é uma interface de programação para documentos HTML e XML e define a maneira como o documento pode ser acessado e manipulado.

Ao usar DOM você pode criar um documento , percorrer sua estrutura e incluir, modificar ou excluir elementos.

A DOM , que também é uma especificação do consórcio W3C, tem como objetivo fornecer uma interface de programação padrão que pode ser usada em diversos ambientes e aplicações.

Vamos exibir o arquivo boletim.xml , cujo código do documento é dado a seguir , usando DOM:

```
<?xml version="1.0"?>
<boletim data="20/04/2000">
  <aluno>
    <nome>JANICE</nome>
    <notas>
      <portugues>7</portugues>
      <matematica>8</matematica>
      <ciencias>6</ciencias>
    </notas>
  </aluno>
</aluno>
<aluno>
  <nome>JEFFERSON</nome>
  <notas>
    <portugues>5</portugues>
    <matematica>8</matematica>
    <ciencias>7</ciencias>
  </notas>
</aluno>
</boletim>
```

Boletim.xml

Para poder exibir os dados do documento boletim.xml iremos usar um arquivo de script ASP onde iremos usar a DOM. O código do arquivo boletim.asp é:

```
<%language = VBScript%>
<html>
<head>
<title> XML -> DOM </title>
</head>

<% dim XMLdoc, raiz, item , nome, strnotas
set XMLdoc = Server.CreateObject("Microsoft.XMLDOM")
XMLdoc.load(Server.MapPath("boletim.xml"))
set raiz = XMLdoc.DocumentElement

Response.Write "<h2> Exibindo XML com DOM - "
Response.Write raiz.Attributes.GetNamedItem("data").text & "</h1>" & vbcrLf
If raiz.hasChildNodes() then
  For each aviso in raiz.childNodes
    For each item in aviso.childNodes
      if "nome"= item.nodeName then
        response.write "<b>" & item.text & "</b>"
      elseif "notas" = item.NodeName then
        strnotas=""
        for i=0 to item.childNodes.length - 1
```

XML – Introdução

```
        strnotas = strnotas & item.childNodes(i).text & " | "  
    next  
    strnotas = left(strnotas, len(strnotas)-1) & "-" & "<br>" & vbcrLf  
    response.write " | " & strnotas  
end if  
next  
next  
end if  
%>  
</body>  
</html>
```

Vamos analisar o código do arquivo boletim.asp:

– **set XMLdoc = Server.CreateObject("Microsoft.XMLDOM")**

Criamos uma instância do DOM usando o objeto Microsoft.XMLDOM

XMLdoc.load(Server.MapPath("boletim.xml"))

Carregamos o arquivo boletim.xml

– **set raiz = XMLdoc.DocumentElement**

Recuperamos o primeiro nó do documento , o nó raiz , que é o elemento "boletim"

– **Response.Write raiz.Attributes.GetNamedItem("data").text & "</h1>" & vbcrLf**

Recuperamos o atributo "data" usado no elemento "boletim"

If raiz.hasChildNodes() then

Verificamos se há realmente elementos filhos (aluno) no documento

```
For each aviso in raiz.childNodes  
    For each item in aviso.childNodes  
  
        if "nome"= item.nodeName then  
            response.write "<b>" & item.text & "</b>"  
        elseif "notas" = item.NodeName then  
            strnotas=""  
            for i=0 to item.childNodes.length - 1  
                strnotas = strnotas & item.childNodes(i).text & " | "  
            next  
            strnotas = left(strnotas, len(strnotas)-1) & "-" & "<br>" & vbcrLf  
            response.write " | " & strnotas  
        end if  
    next  
next
```

O código acima faz o looping em todos os nós filhos (**aviso.ChildNodes**) , e, através de outro loop iteramos nos nós filhos "nome" e "notas" exibindo os dados ao usuário.

O resultado final do processamento é mostrado na figura abaixo:



2– Usando as folhas de estilo XSL

Nós já vimos que as tags usadas em um documento XML podem ser criadas pelo usuário, então como o Navegador irá interpretar essas tags, já que não existem tags padrão em XML? Para exibir documentos XML será necessário um mecanismo que descreva como o documento será exibido. Esse mecanismo chama-se **XSL – Extensible Stylesheet Language** – e, é usado para transformar XML em HTML.

A XSL pode ser encarada como uma linguagem que pode transformar XML em HTML, pode filtrar e ordenar dados em documentos XML e que pode formatar dados XML. Ela pode ser usada para definir em como um arquivo XML será exibido pela transformação em um arquivo reconhecido pelo navegador do usuário. Geralmente este serviço é feito pela transformação de cada elemento XML em um elemento HTML, sendo que a XSL pode incluir novos elementos, remover elementos, rearranjar e ordenar elementos e ainda testar e tomar decisões.

Vamos criar um arquivo XML chamado **catalago.xml** e a seguir usar a XSL e transformá-lo no arquivo **catalago.xsl**:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<CATALOGO>
  <CD>
    <TITULO>Minas</TITULO>
    <ARTISTA>Milton Nascimento</ARTISTA>
    <PAIS>BRAZIL</PAIS>
    <COMPANIA>Columbia</COMPANIA>
    <PRECO>15.90</PRECO>
    <ANO>1985</ANO>
  </CD>
</CATALOG>
```

Catalago.xml Um catálogo de CD como documento XML

O arquivo **catalago.xsl** correspondente tem o seguinte código:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Titulo</th>
        <th>Artista</th>
      </tr>
      <xsl:for-each select="CATALOGO/CD">
        <tr>
          <td><xsl:value-of select="TITULO"/></td>
          <td><xsl:value-of select="ARTISTA"/></td>
```



```

        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Catalogo.xsl

Vejamos a explicação do código usado no arquivo catalgo.xsl:

xsl:stylesheet Indica que o documento é um stylesheet – folha de estilo. **<xsl:template match="/">** informa que este é um modelo que corresponde a raiz (/) de um documento fonte XML – **<?xml version='1.0'?>**

Como um arquivo XSL é também um arquivo XML ele deve iniciar com uma declaração XML

```
<xsl:for-each select="CATALOGO/CD">
```

O elemento **xsl:for-each** localiza elementos no documento XML e repete o modelo para cada elemento. O atributo **select** descreve o elemento na fonte do documento

```

<td><xsl:value-of select="TITULO"/></td>
<td><xsl:value-of select="ARTISTA"/></td>

```

Nessas linhas o elemento **xsl:value-of** seleciona um nó filho na hierarquia e insere o conteúdo do nó filho no modelo.

Para transformar o arquivo catalogo.xml em um arquivo HTML basta incluir uma referência ao arquivo catalogo.xsl criado acima no arquivo catalogo.xml . Veja no exemplo a seguir:

```

<?xml version="1.0" encoding="ISO8859-1" ?>

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

<CATALOGO>
  <CD>
    <TITULO>Minas</TITULO>
    <ARTISTA>Milton Nascimento</ARTISTA>
    <PAIS>BRAZIL</PAIS>
    <COMPANIA>Columbia</COMPANIA>
    <PRECO>15.90</PRECO>
    <ANO>1985</ANO>
  </CD>
</CATALOG>

```

Para fazer esta transformação no cliente , usamos o código a seguir:

```

<html>
<body>
<script language="javascript">
// carrega XML
var xml = new ActiveXObject("Microsoft.XMLDOM")
xml.async = false
xml.load("catalogo.xml")

// carrega XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")

```

```
xsl.async = false
xsl.load("catalog.xsl")

// Transforma XML em HTML
document.write(xml.transformNode(xsl))
</script>

</body>
</html>
```

```
- var xml = new ActiveXObject("Microsoft.XMLDOM")
```

O primeiro bloco de código cria uma instância da XML usando o Microsoft.XMLDOM

```
- xml.load("catalogo.xml")
```

Carrega o documento XML na memória

```
- var xsl = new ActiveXObject("Microsoft.XMLDOM")
```

Esta linha de código cria uma instância XML

```
- xsl.load("catalog.xsl")
```

Carrega o documento XSL na memória

```
- document.write(xml.transformNode(xsl))
```

Transforma o documento XML usando o documento XSL e gera um documento HTML

Embora a XML esteja nos primórdios , dia a dia ela é cada vez mais usada . Com esta pequena introdução espero que XML e XSL deixem de ser apenas um par de siglas sem sentido para você e passem a orbitar os seus scripts ASP pois pode ter certeza, você ainda vai ouvir falar muito nessa dupla.