Bem-vindo ao XML

Bem-vindo a Extensible Markup Language, XML, a linguagem para tratamento de dados em uma forma compacta, fácil de gerenciar – sem mencionar o mais poderoso avanço visto na Internet nos últimos anos.

Diferente do HTML, a linguagem XML não se preocupa com a apresentação e sim com o empacotamento do dado para poder transportá-lo facilmente. A principal razão do XML ter ganho tanta popularidade é que os dados são armazenados como texto, o que possibilita que os documentos XML possam ser transferidos usando-se a tecnologia da Web existente, construída para transferência de documentos na forma de texto. O exemplo mais comum de uma linguagem de marcas é o HTML que pode ser visto na Listagem 1.1.

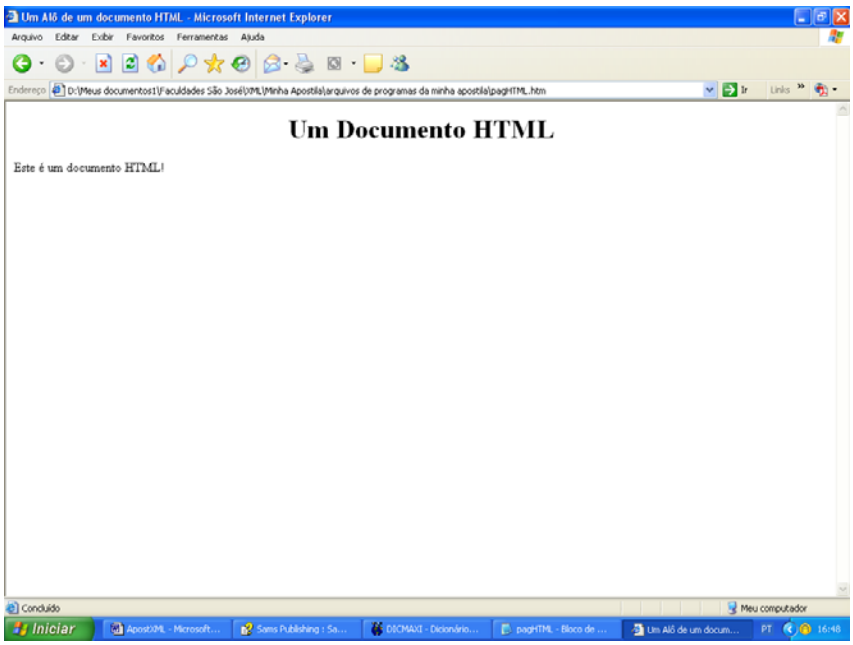
Listagem 1.1 Um exemplo de uma página HTML

```
<HTML>
  <HEAD>
    <TITLE> Um Alô de um documento HTML </TITLE>
  </HEAD>
  <BODY>
    <CENTER>
      <H1>
        Um Documento HTML
      </H1>
    </CENTER>
    Este é um documento HTML!
  </BODY>
</HTML>
```

As marcas neste documento HTML estão ali para informar ao navegador como interpretar os dados contidos no documento – o que é cabeçalho, o que é texto para o corpo do documento, e assim por diante. Estas marcas de HTML são constituídas por tags HTML do tipo <HEAD>, <BODY>, e outras mais, e essas tags

direcionam as atividades do navegador. A página da Listagem 1.1 pode ser vista na Figura 1.1. Observe que como as marcas HTML só estão no documento para direcionar as atividades do navegador, nenhuma delas é mostrada no documento quando aberto pelo navegador.

Figura 1.1. Uma página HTML em um navegador.



A linguagem HTML como uma linguagem de marcas é muito limitada. Ela é boa para a criação de páginas Web padrões, mas ela não pode ir muito além disso.

Por exemplo, HTML foi desenvolvida para criar páginas Web que mostrem texto padrão e algumas imagens, e as tags HTML do tipo <img>, <table>, e outras mais são ótimas para isso. Mas, à medida que as coisas tornam-se mais complexas, o HTML original não é mais suficiente.

Para atender as exigências atuais para linguagens de marcação, o HTML deveria ter centenas de tags adicionais. Mesmo assim, não teria como essas tags adicionais atenderem a todas as situações possíveis – por exemplo, como armazenar informações sobre os clientes de uma empresa? Não existem tags

HTML para <primeiro nome>, <último nome>, <telefone> etc. É aqui que o XML torna-se importante pois permite que o desenvolvedor crie suas próprias tags.

Entendendo XML

Extensible Markup Language, XML, é realmente importante ao permitir as marcas sejam criadas de acordo com a necessidade. Diferente do HTML, XML foi desenvolvida para o armazenamento de dados, e não com a preocupação de apresentá-los. XML provê uma forma de estruturação dos dados em um documento e foi aceito rapidamente pelos profissionais da área porque os documentos XML são do tipo texto, e isto possibilita que sejam transmitidos usando a tecnologia existente da Internet que foi desenvolvida para o HTML.

Pode-se empacotar listas de livros em uma biblioteca, tipos de peixes, produtos de uma empresa etc. em um documento XML, coisa que o HTML não permite. Uma vez que os dados estejam empacotados, podem ser transmitidos pela Internet e uma pessoa, ou mesmo um software dedicado, pode recebê-lo e tratá-lo.

XML é uma criação do World Wide Web Consortium (W3C) http://www.w3.org, que é o grupo responsável pelo HTML e por outras especificações. W3C publica suas especificações no seu site e o desenvolvedor que deseja trabalhar com XML e todas as especificações relacionadas precisa conhecê-las.

XML versão 1.1 é a recomendação mais atual e pode ser obtida em http://www.w3.org/TR/2004/REC-xml11-20040204, editada em 15 de Abril de 2004.

Mas qual é o formato de um documento XML? Pode-se ver um documento XML na Listagem1.2.

Listagem 1.2 Exemplo de um Documento XML (exempXML.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
      Este e um documento XML!
   </mensagem>
</documento>
```

Igual a todo documento XML, este também começa com uma declaração XML, <?xml version="1.0" encoding="UTF-8"?>. Esta declaração XML indica que o documento usa o XML versão 1.0 e a codificação de caracteres UTF-8, que significa que o documento usa a versão 8-bit condensada do Unicode.

Esta declaração, <?xml?>, usa dois atributos, version e encoding, para configurar a versão do XML e o conjunto de caracteres usado. Atributos XML são muito parecidos com os atributos HTML – eles são responsáveis por guardar algum tipo de informação adicional. Os atributos são configurados com um valor entre aspas duplas como mostrado aqui: version = "1.0". Diferente do HTML, se você estiver usando um atributo, você precisa sempre configurá-lo com um valor pois não há valores default como no HTML.

Observe ainda no Exemplo 1.2, exempXML.xml, que na próxima linha é criado um novo elemento XML chamado <documento>. Como no HTML, um elemento é a unidade fundamental que se utiliza para armazenar um dado - todos os dados em um documento XML precisam estar em um elemento. Elementos sempre iniciam com uma tag de abertura que, neste caso. é <documento>, termina com uma tag de fechamento que, neste caso, é </documento> (Note que é similar e não igual pois no HTML nem sempre é necessário usar uma tag de fechamento). As tags XML

sempre começam com < e terminam com >. Cria-se um elemento XML com pares de tags de abertura e fechamento como foi feito para criar o elemento <documento>.

```
<?xml version="1.0" encoding="UTF-8"?>
<documento>
   .
   .
   .
</documento>
```

Agora o desenvolvedor está livre para armazenar outros elementos no interior do elemento <documento>, ou dados de texto, como desejado. Pode-se criar elementos com qualquer nome no XML e isto é a grande vantagem do XML – a capacidade de criação de marcas com qualquer nome. Não é preciso chamar o elemento de <documento>; ele poderia ter sido nomeado como <data>, ou <record>, ou <people>, ou <registro>, ou <planetas>, ou qualquer outro nome. Um nome de elemento pode iniciar com uma letra ou um underscore, e os caracteres seguintes podem ser letras, dígitos, underscores, pontos ou hífens, pelo menos na Versão 1.0 do XML. Diferente do HTML, o fato das letras estarem em maiúsculo ou minúsculo faz diferença pois <DOCUMENTO> não é a mesma coisa que <documento>, por exemplo.

Entre uma tag de abertura e a de fechamento, pode-se inserir o conteúdo do elemento. Um conteúdo de elemento pode ser constituído de textos simples ou até outros elementos. Da mesma forma que as declarações XML, os elementos XML também podem conter atributos.
Quando se cria um documento XML tem-se que encapsular todos os elementos dentro de um elemento maior, chamado de elemento raiz, também chamado de elemento documento. O elemento raiz contém todos os outros elementos contidos em um documento XML e, no nosso caso, nomeamos o nosso elemento como <documento>. Os documentos XML sempre precisam de um elemento raiz, mesmo se eles não têm quaisquer outros elementos ou mesmo texto (ou seja, mesmo que o elemento raiz não tenha qualquer outro conteúdo).

Dentro do nosso elemento, adicionaremos um novo elemento ao nosso documento chamado <cabecalho>, como mostrado abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento>
   <cabecalho>
      .
      .
      .
   </ cabecalho >
   .
   .
   .
</documento>
```

Este novo elemento conterá dado na forma de texto – “Alo de um documento XML”;

```
<?xml version="1.0" encoding="UTF-8"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   .
   .
   .
</documento>
```

Nós também adicionaremos um outro elemento ao elemento raiz (não existe limite para o número de subelementos que um elemento pode conter), que chamaremos de <mensagem>, e que armazenará o texto " Este e um documento XML!":

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

```
  <cabecalho>
    Alo de um documento XML
  </cabecalho>
  <mensagem>
    Este e um documento XML!
  </mensagem>
</documento>
```

E isto completa o nosso primeiro documento XML. Neste caso, o elemento raiz, <documento>, contém dois elementos, <cabecalho> and <mensagem>, ambos contendo dados na forma de texto (embora eles também pudessem conter outros elementos).

Como você pode ver, este documento XML se parece com o documento HTML criado no início – os elementos que criamos estão encapsulados por tags, da mesma forma que no documento HTML. Contudo, nós nomeamos os elementos no documento XML como desejamos; nós não seguimos um conjunto pré-definido de nomes para tags. Essa capacidade de criar seus próprios elementos tem vantagens e desvantagens –você não está restrito a um conjunto pré-definido e limitado de tags, mas, por outro lado, um navegador padrão pode tratar tags HTML mas não saberá como tratar uma tag <mensagem> como a que criamos para o nosso exemplo.

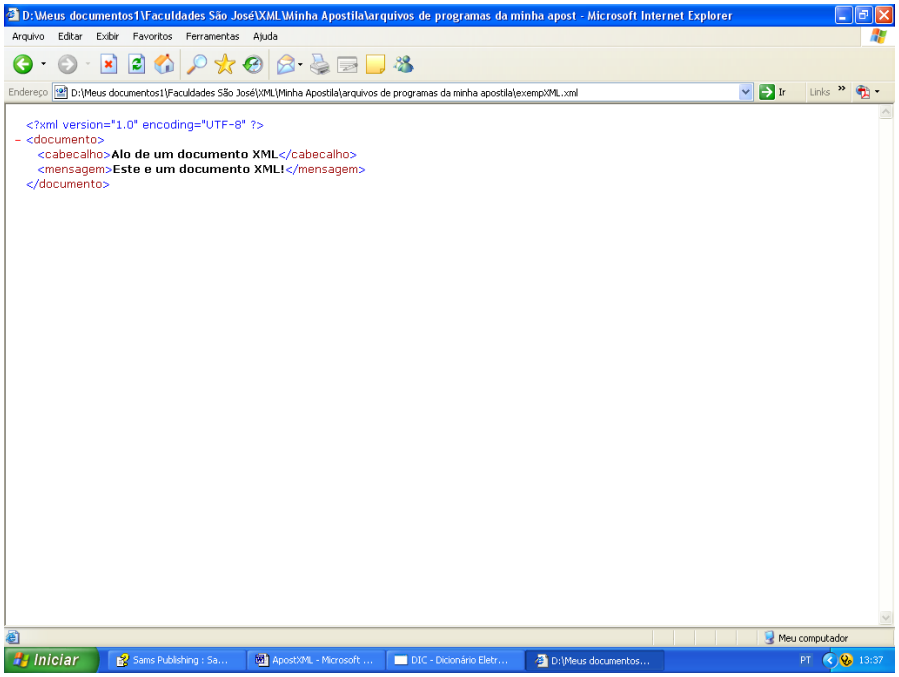
Nós já armazenamos nosso dado em um documento XML; para iniciar o tratamento desses dados, nós começaremos simplesmente fazendo com que o navegador abra o nosso documento.

Olhando o documento XML em um navegador

Alguns navegadores, tais como o Microsoft Internet Explorer versão 5 or os mais recentes, permitem que os documentos XML sejam apresentados diretamente.

Por exemplo, pode-se abrir o arquivo *exempXML.xml* com o Internet Explorer como mostrado na Figura 1.2.

Figura 1.2. Abrindo um documento XML em um navegador.



Como pode ser visto na figura, todo o conteúdo do documento XML que nós criamos é apresentado. Você pode até clicar no sinal – localizado na frente do elemento <documento> para fechar todo o conteúdo daquele elemento em uma única linha (que terá um sinal + em frente, indicando que aquela linha pode ser expandida). Neste caso, você pode mostrar o documento original no Internet explorer.

Observe, contudo, que o Internet Explorer não fez mais nada do que mostrar o conteúdo do documento, ou seja, ele não fez qualquer tipo de tratamento com o conteúdo do documento XML. Isto aconteceu desta forma porque os navegadores são feitos para apresentação de dados, não para interpretar tags XML.

De fato, se você está somente interessado na apresentação de seus dados, você pode usar tags XML para dizer ao navegador como apresentá-los usando folhas de estilo. Por exemplo, você poderia criar um elemento chamado <vermelho> para dizer ao navegador que todo texto contido no seu interior deveria ser apresentado em vermelho. Usando folhas de estilo, você pode fazer um navegador interpretar seu XML somente para fazê-lo apresentar seus dados.

Uma das razões mais populares para utilização de folhas de estilo com XML é que você pode armazenar seus dados no documento XML, e especificar como aqueles dados devem ser apresentados usando um arquivo separado, no caso, uma folha de estilo. Isto faz com que os dados fiquem separados dos detalhes de apresentação, diferente do HTML, onde as tags que especificam como apresentar os dados estão misturadas com os próprios dados.

Existe muito suporte para quem trabalha com documentos XML e folhas de estilo nos navegadores Internet Explorer e Netscape. Existem dois tipos de folhas de estilo que podem ser usadas com documentos XML – Cascading Style Sheets (CSS), que também é usada com documentos HTML, e Extensible Stylesheet Language (XSL), projetada para ser usada somente com documentos XML.

Como um exemplo, vamos usar CSS para formatar nosso exemplo de documento XML. Para usar CSS, precisaremos de uma instrução de processamento XML, <?xml-stylesheet?>, suportada tanto pelo Internet Explorer como pelo Netscape, que possibilita associar uma folha de estilo CSS com um documento XML.

Como você poderia deduzir, as instruções de processamento são instruções para o software que estiver processando o documento XML; todas as instruções de processamento como esta iniciam com <? e terminam com ?>. Instruções de processamento podem aparecer através de um documento XML e também podem ter atributos. Da mesma forma que acontece com os elementos XML, você está livre para criar suas próprias instruções de processamento —a instrução de

processamento <?xml-stylesheet?> não é uma instrução embutida no XML, ela foi criada e é suportada tanto pelo Netscape como pelo Internet Explorer.

Nesta caso, esta instrução de processamento terá seu atributo type configurado para "text/css" de modo a indicar que estaremos usando uma folha de estilo CSS, e seu atributo href será configurado com a localização da folha de estilo CSS (da mesma forma que o atributo href de um elemento HTML<a> especifica o target de um hiperlink), como pode ser verificado no arquivo exemp2XML.xml do Listagem 1.3.

Listagem 1.3 Documento XML usando Folha de Estilo (exemp2XML.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="exem1css.css"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
      Este e um documento XML!
   </mensagem>
</documento>
```

Neste caso, nós nomeamos a folha de estilo CSS como exem1css.css e o conteúdo deste arquivo pode ser visto na Listagem 1,4.

Listagem 1.4 Folha de Estilo (exem1css.css)

```
cabecalho {display: block; font-size: 24pt; color: #ff0000; text-align: center}
mensagem {display: block; font-size: 18pt; color: #0000ff; text-align: center}
```

No arquivo exem1css.css, estamos dizendo para o navegador como apresentar o conteúdo dos elementos XML. Em particular, estamos dizendo que o conteúdo do elemento <cabecalho> deve ser apresentado centralizado no navegador, com tamanho do fonte de 24 pontos e na cor vermelha, e o conteúdo do elemento

<mensagem> deve ser apresentado centralizado, fonte no tamanho 18 pontos e na cor azul.

Você pode ver o resultado no Internet Explorer, como mostrado na Figura 1.3.

Figura 1.3. Exibindo um documento XML em um navegador.

Desta forma, somos capazes de dizer ao navegador como queremos que nosso dado seja formatado, usando elementos XML, e como queremos que o navegador interprete esses elementos XML, usando uma folha de estilo.

Usar o XML para indicar como o dado deve ser apresentado é somente o início, pois pode-se também extrair dados do próprio documento XML. Por exemplo, pode-se usar uma linguagem scripting, tipo JavaScript, para dizer ao navegador como extrair dados dos elementos de um documento XML,

Trabalhando com Dados em um Documento XML

Imagine que o objetivo seja extrair os dados de um documento XML e tratar esses dados, em vez de somente exibi-los. Por exemplo, suponha que se queira extrair o texto contido no elemento <cabecalho>:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="exem1css.css"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
      Este e um documento XML!
   </mensagem>
</documento>
```

Um modo de ganhar acesso a dados em um navegador é usar JavaScript, que navegadores tipo Internet Explorer e Netscape suportam. Observe na Listagem 1.5 um exemplo de uma página HTML com código JavaScript embutido que extrai dados de um documento XML.

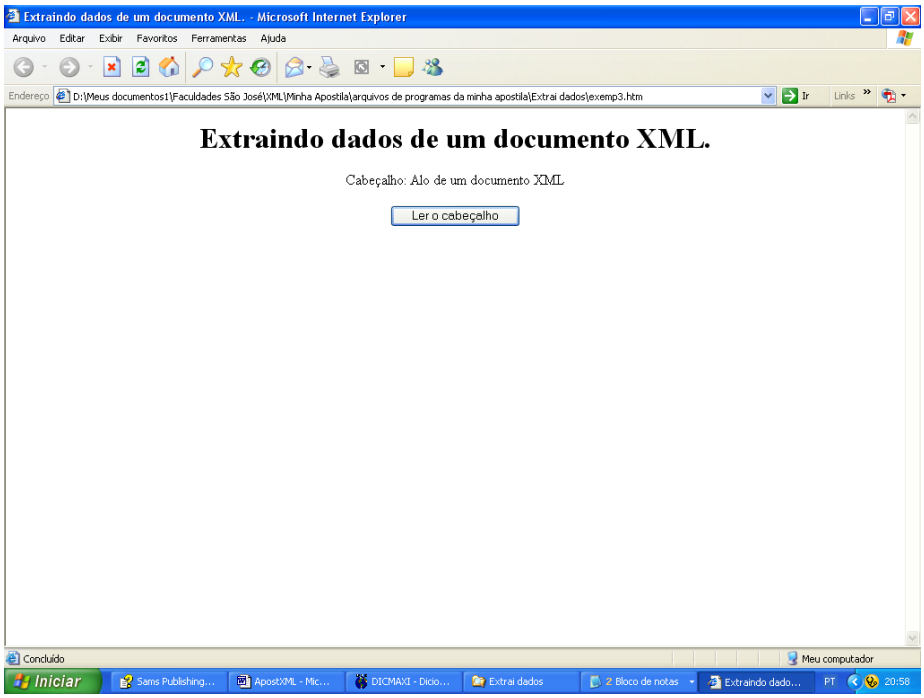
Listagem 1.5 Extraindo dados de um documento XML usando Javascript (exemp3.htm)

```
<HTML>
   <HEAD>
      <TITLE>
         Extraindo dados de um documento XML.
      </TITLE>

      <XML ID="firstXML" SRC="exemp2XML.xml"></XML>

      <SCRIPT LANGUAGE="JavaScript">
         function getData()
         {
            xmldoc= document.all("firstXML").XMLDocument;
```

```
            nodeDoc = xmldoc.documentElement;
            nodeHeading = nodeDoc.firstChild;

            outputMessage = "Cabeçalho: " +
                nodeHeading.firstChild.nodeValue;
            message.innerHTML=outputMessage;
        }
    </SCRIPT>
  </HEAD>

  <BODY>
    <CENTER>
      <H1>
          Extraindo dados de um documento XML.
      </H1>

      <DIV ID="message"></DIV>
      <P>
      <INPUT TYPE="BUTTON" VALUE="Ler o cabeçalho"
        ONCLICK="getData()">
    </CENTER>
  </BODY>
</HTML>
```

Quando você abre este exemplo no Internet Explorer, ele mostra um botão cujo rótulo é "Ler o cabeçalho", como você pode ver na Figura 1.5. Quando você clica esse botão, o JavaScript lê o texto no elemento <cabecalho> do nosso documento XML, exemp2XML.xml, e apresenta-o, como pode ser visto na figura. Desta forma, fomos capazes de extrair dado de um documento XML – e tendo extraído o dado de um documento XML, estamos livres para tratá-lo da forma que quisermos.

Figura 1.5. Extraindo dados de um documento XML.



Como você pode ver, é possível extrair dados de um documento XML. Sendo assim, um outro desenvolvedor pode escrever tal documento usando tags que vocês dois acordaram, enviar tal documento pela Internet para você, e você pode, então, extrair os dados que precisa do documento procurando por elementos com nomes específicos.

Estruturando seus dados

Um documento XML realmente pode fazer mais do que simplesmente armazenar seus dados; ele também permite que você especifique a estrutura desses dados. Esta estruturação é muito importante quando você está trabalhando com dados complexos. Por exemplo, você poderia armazenar um longo extrato bancário em HTML, mas após as primeiras dez páginas ou mais, esses dados estariam propensos a erros. Mas usando XML, você pode realmente construir as regras de

sintaxe que especificam a estrutura do documento de modo que este possa ser verificado e assim possa-se garantir que ele está sendo implementado corretamente.

Esta ênfase na correta implementação da estrutura dos dados é forte no XML e possibilita uma fácil localização de problemas de implementação. No HTML, o autor de Web poderia (e freqüentemente isso acontece) escrever um HTML malfeito, sabendo que o navegador se encarregará de resolver os problemas de sintaxe. De fato, algumas pessoas estimam que 50% ou mais do código dos navegadores modernos está lá para tratar aqueles HTML malfeitos das páginas Web. Já no XML isso não acontece pois espera-se que o software que lê seu XML – chamado de processador de XML – execute essa verificação do seu documento; se existir um problema, espera-se que o processador interrompa a execução e informe sobre o ocorrido.

Mas como espera-se que um processador XML verifique seu documento? Existem duas verificações que os processadores XML podem fazer: verificar que seu documento está formatado corretamente e verificar que ele é um documento válido. Veremos estas duas formas de verificação com mais detalhes mais tarde; agora, vamos vê-las em formas gerais.

Criando Documentos XML Bem-Formados

O que significa para um documento XMl ser bem-formado? Formalmente, isso significa que o documento precisa seguir as regras de sintaxe especificadas para o XML pelo W3C na recomendação XML 1.0 ou na recomendação XML 1.1. Embora exista uma série de requisitos para um documento ser bem-formado, informalmente, os principais requisitos são que o documento precisa conter um ou mais elementos, e um elemento, o elemento raiz, precisa conter todos os outros

elementos. Além disto, cada elemento precisa estar devidamente encapsulado por outros elementos, quando for o caso.

Abaixo temos um exemplo de um documento que não está bem formado – ele não é um documento bem-formado porque a tag de fechamento </cabecalho> está colocada após a tag de abertura <mensagem>, misturando os elementos <cabecalho> e <mensagem>:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="exem1css.css"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   <mensagem>
   </cabecalho>
      Este e um documento XML!
   </mensagem>
</documento>
```

Criando Documentos XML Válidos

Um processor XML normalmente verificará se os documentos XML estão bem formados, mas somente alguns são capazes de verificar se os arquivos são válidos. Um documento XML é válido se ele está de acordo com a sintaxe que você especificou e se esta sintaxe pode ser especificada ou em um DTD (Document Type Definition) ou em um XML Schema.

Como um exemplo, observe como adicionar um DTD ao seu documento XML na Listagem 1.6. DTDs podem ser documentos separados, ou eles podem ser embutidos em um documento XML como feito nesse exemplo com a utilização de um elemento especial chamado <!DOCTYPE>.

Listagem 1.6 Um Documento XML com um DTD (exemp4.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="exem1CSS.css"?>
<!DOCTYPE documento [
        <!ELEMENT documento (cabecalho, mensagem)>
        <!ELEMENT cabecalho (#PCDATA)>
        <!ELEMENT mensagem (#PCDATA)>
]>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
      Este e um documento XML!
   </mensagem>
</documento>
```

De forma resumida, o DTD da Listagem 1.6 é o elemento <!DOCTYPE> que especifica que o elemento raiz, <document>, deve conter um elemento <heading> e um elemento <message>. Nós também especificamos que os elementos <heading> e <message> podem conter dados na forma de texto. Usando um DTD deste tipo, pode-se especificar a sintaxe que o seu documento XML deve obedecer – que elementos devem ser colocados dentro de outros elementos, que atributos um elemento pode ter, e assim por diante – e se um processador XML pode executar validação, ele pode verificar o seu documento e antecipar alguns problemas que seriam encontrados mais tarde.

Usando Navegadores XML

Chamar um navegador de navegador XML significa uma de duas coisas. Como já vimos, um navegador tipo Internet Explorer pode mostrar documentos XML, e você pode até usar CSS ou XSL para formatar esses documentos. Contudo,

apresentar um documento XML é uma coisa, usar os dados desse documento é uma outra coisa, e é isso que fazemos no segundo tipo de navegador XML.

Por exemplo, com JavaScript você pode acessar o dado em um documento XML em um navegador do tipo Internet Explorer, e você até pode montar o HTML que o navegador irá mostrar. Existem navegadores XMl dedicados para aplicações XML específicas, e estes vão muito além de mostrar somente HTML. Um exemplo de um navegador XML dedicado é o Jumbo, um navegador para apresentação de documentos XML usando a Chemical Markup Language (CML) na representação de moléculas químicas

Usando XML no Internet Explorer

Microsoft's Internet Explorer é longe o mais poderoso navegador XML de uso geral disponível atualmente. A versão 6 é muito poderosa para trabalhar com XML. Internet Explorer pode mostrar documentos XML diretamente, como vimos anteriormente. Ele pode usar linguagens scripting, tipo JavaScript, para acessar o dado em um documento XML e permitir que se trate esse dado contido no código. Ele também permite que se trabalhe documentos XML com folhas de estilo CSS e XSL, permitindo a formatação e apresentação de dados XML na forma que desejarmos. Ele valida documentos XML usando tanto DTDs como XML schemas. O Internet Explorer vincula dados a controles HTML do tipo caixas de texto e botões. Existe até um elemento especial, <XML> , que pode carregar documentos XML automaticamente.

Usando Validadores XML

Um validador XML verifica seu XML para certificar-se de que ele está bem formado e válido, informando se existir um problema. Um link para um validador na Internet é http://www.stg.brown.edu/service/xmlvalid/.

Criando Documentos XML Parte por Parte

Até o momento, criamos o seguinte documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
      Este e um documento XML!
   </mensagem>
</documento>
```

Esse é um documento XML formado corretamente, mas é só um exemplo. Agora, seremos mais sistemáticos sobre o que deve ser utilizado em um documento XML, discutindo todas as possíveis partes de tais documentos. As partes de um documento que serão estudadas são as seguintes:

- Prologs
- Declarações XML
- Instruções de Processamento
- Elementos and atributos
- Comentários
- Seções CDATA
- Entidades

W3C define tudo o que pode ser utilizado em documentos XML nas especificações 1.0 e 1.1, desde o início – o conjunto de caracteres utilizado.

Codificação de Caracteres: ASCII, Unicode, e UCS

Os caracteres em um documento XML são armazenados usando códigos numéricos. Isso pode ser um problema pois existem inúmeros conjuntos de caracteres usando diferentes códigos numéricos, o que significa que um processador XML pode ter problemas ao tentar ler um documento XML que faça uso de um conjunto de caracteres diferente do usual. Por exemplo, um conjunto de caracteres comum usado por editores de texto é o American Standard Code for Information Interchange (ASCII). O código ASCII utiliza códigos numéricos de 0 a 255 – por exemplo, o código ASCII para a letra A é 65, para a letra B é 66, e assim por diante. Se você armazena a palavra “cat” em um documento XML escrito em ASCII, os números 67, 65 e 84 são os códigos que você realmente estará armazenando no arquivo. Por outro lado, a World Wide Web é justo isso, mundial. Por ser mundial, os 256 caracteres do ASCII não poderiam conter todos os inúmeros conjuntos de caracteres existentes no mundo, tais como Cyrillic, Armenian, Hebrew, Thai, Tibetan, e assim por diante.

Por esta razão, o W3C se voltou para o Unicode (http://www.unicode.org), conjunto que contém 65.536 caracteres (embora somente algo próximo a 40.000 códigos Unicode tenham sido reservados até o momento), e não somente os 256 do código ASCII. Para facilitar essa conversão, os primeiros 256 códigos do Unicode são os mesmos do Código ASCII.

Existe um outro conjunto de caracteres que tem até mais espaço do que o Unicode – o Universal Character System (UCS, também chamado ISO 10646) usa 32 bits - dois bytes - por caractere. Isto significa um conjunto de dois bilhões de caracteres – observe que existem mais caracteres Chineses do que o código Unicode pode conter. O UCS também inclui o conjunto de caracteres Unicode – cada caracter Unicode é representado pelo mesmo código UCS, da mesma forma que o Unicode inclui o conjunto de caracteres ASCII.

Assim sendo, quais conjunto de caracteres são suportados pelo XML? ASCII? UCS? Unicode usa dois bytes para cada caractere, logo, um arquivo Unicode teria o dobro do tamanho de um arquivo ASCII. Por essa e por outras razões, é difícil converter muitos dos softwares disponíveis para o código Unicode. O XML na realidade suporta uma versão comprimida do Unicode criada pelo grupo UCS chamada UCS Transformation Format-8 (UTF-8). UTF-8 inclui todos os códigos ASCII, e usa um único byte para os caracteres Unicode mais comuns. Quaisquer outros caracteres Unicode usarão mais do que um byte (até seis bytes).

O próprio UCS também foi comprimido para um conjunto de caracteres chamado UTF-16, que usa dois bytes (em vez dos quatro bytes que o UCS usa normalmente) para os caracteres mais comuns, e mais bytes para os caracteres menos comuns.

O W3C requer que os processadores XML suportem tanto o UTF-8 (Unicode comprimido, incluindo o conjunto ASCII completo) como o UTF-16 (UCS comprimido, incluindo o conjunto ASCII completo), e esses são os únicos dois requisitos. A codificação UTF-8 é a mais popular atualmente em documentos XML, visto que você pode armazenar documentos em ASCII usando um editor de texto e eles podem ser tratados, sem qualquer mudança, como UTF-8 por um processador XML (ASCII usa um byte por caracter, e UTF-8 usa um byte para os caracteres mais comuns, incluindo todos os caracteres do conjunto ASCII). Na realidade, nós estamos usando UTF-8 desde o nosso primeiro exemplo XML, como você pode ver no trecho em que especificamos a codificação de caractere com o atributo encoding na declaração XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
```

```
    Este e um documento XML!
  </mensagem>
</documento>
```

UTF-8 é tão usado que um processador XML irá assumir seu uso se você omitir o atributo de codificação. Embora o W3C requeira que todos os processadores XML suportem o UTF-16 e o UTF-8, a maioria ainda não suporta o UTF-16.

Embora somente o UTF-8 e o UTF-16 sejam requeridos, existem muitos conjuntos de caracteres que um processador XML pode suportar, tais como os seguintes:

- US-ASCII— U.S. ASCII
- UTF-8—Unicode Comprimido
- UTF-16—UCS Comprimido
- ISO-10646-UCS-2— Unicode
- ISO-10646-UCS-4— UCS
- ISO-2022-JP— Japanese
- ISO-2022-CN— Chinese
- ISO-8859-5— ASCII e Cyrillic

A crescente adoção do Unicode é a força que impulsiona o XML 1.1. Existem três áreas principais nas quais o XML 1.1 difere do XML 1.0, todas tendo a ver com caracteres:

- XML 1.1 aceita mais caracteres Unicode do que os disponíveis quando XML 1.0 foi criado.
- XML 1.1 relaxou em algumas regras de criação de nomes (como as usadas para elementos e atributos) para permitir mais caracteres Unicode, e para possibilitar uma expansão no futuro.
- XML 1.1 permite que mais caracteres legais terminem uma linha.

Entendendo Marcas XML e Dados XML

No seu nível mais básico, documentos XML são combinações de marcas e dados na forma de texto. Eles podem vir até a incluir dados binários em algum dia, mas, no momento, não existe nenhuma maneira de incluir dados binários em um documento XML. (Se você quiser associar dado binário com um documento XML, você deve deixá-lo externo ao documento e usar uma referência de entidade, como veremos com detalhes mais tarde).

As marcas em um documento definem sua estrutura. Marcas incluem tags de início, de fim, tags de elementos vazios, referências de entidades, referências de caracteres, comentários, delimitadores de seção CDATA, declarações de tipo de documento e instruções de processamento. E o que falar de dados em um documento XML? Todo o texto em um documento XML não é marca, é dado.

Embora as marcas que vimos até agora tenham consistido de tags, existe um outro tipo de marcas que não usam tags – são as referências de entidades gerais e referências de entidade parâmetros. Enquanto que as tags começam com < e terminam com >, referências de entidades gerais começam com & e terminam com ;. Referências de entidades gerais são substituídas pela entidade a que elas estão se referindo quando o documento é analisado gramaticalmente. Referências de entidades de parâmetros, que começam com % e terminam com ;, são usados em DTDs, como veremos mais tarde.

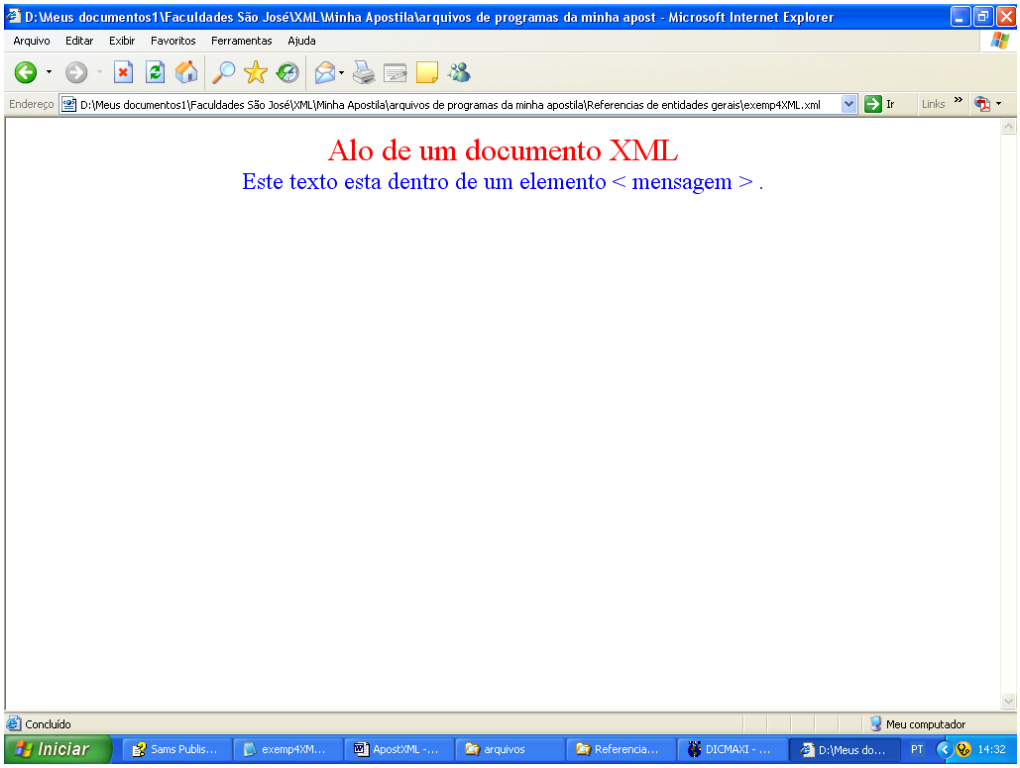
Por exemplo, a marca < é uma referência de entidade geral que se transforma no símbolol < (menor do que) quando analisador gramaticalmente por um processador XML, e a referência de entidade geral > se transforma no símbolo > (maior do que) quando analisado gramaticalmente por um processador XML. Você pode ver um exemplo utilizando referências de entidade gerais na Listagem 2.1.

Listagem 2.1 Usando uma Referência de Entidade (exemp4XML.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="exem1CSS.css"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
      Este texto esta dentro de um elemento &lt; mensagem &gt; .
   </mensagem>
</documento>
```

Você pode ver o exemplo exemp4XML.xml no Internet Explorer na Figura 2.9. Como você pode ver na figura, a marca < se transformou em um símbolo < , e a marca > se transformou em um símbolo > pelo processador XML.

Figure 2.9. Usando Marcas no Internet Explorer.



Além das referências de entidade caracteres, onde um código de caractere é substituído pelo caractere que ele codifica, existem cinco referências de entidade gerais predefinidas no XML, usadas por representarem símbolos que podem confundir o navegador durante a interpretação do documento.

- <— substituído por <
- >— substituído por >
- &— substituído por &
- "— substituído por "
- '— substituído por '

Podemos criar nossas próprias referências de entidade gerais, como veremos mais tarde. Quando um processador XML analisa seu documento XML, ele substitui as referências de entidade gerais, tipo > pela entidade por ela referenciada, que é > neste caso.

Usando Espaço em Branco (Whitespace) e Final de Linha (Ends of Lines)

Espaços, retorno, pular linha, e tabulações são todos tratados como espaços em branco no XML. Isto significa que para um processador XML este documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento>
   <cabecalho>
      Alo de um documento XML
   </cabecalho>
   <mensagem>
      Este e um documento XML!
   </mensagem>
</documento>
```

é o mesmo que este outro, em termos de conteúdo:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento><cabecalho>Alo de um documento XML</cabecalho>
<mensagem>Este e um documento XML!</mensagem></documento>
```