O que é esse tal de XML, afinal?

O XML tem recebido muito destaque em toda a comunidade de usuários de informática. Saiba o que é e qual a sua real utilidade.

Por Dani Edson Pereira*

Se você é de alguma área mesmo que apenas remotamente ligada à informática, então, sem dúvida, já deve ter ouvido falar dessas três letrinhas: X M L. Do que se trata? Para que serve? Por que se fala tanto em XML ultimamente? Será algum tipo de "salvação da humanidade" tecnológica? Neste artigo tentarei explicar do modo mais simples possível o que é e para que serve o XML, bem como por que essa tecnologia está recebendo tanto destaque. Não é de meu propósito entrar em detalhes muito técnicos sobre sintaxes da linguagem. Futuramente, talvez possamos fazer isso.

Inicialmente, para conseguir entender a finalidade da linguagem XML, vamos fazer um breve retrospecto na história da Internet.

No princípio era o HTML...

Tudo começou com a popularidade da Internet, por volta de meados da década de 90. O protocolo HTTP passou a ser usado para transmitir páginas estáticas de servidores Web para os navegadores, ou *web browsers*. Todo mundo queria "publicar" sua *home page* ou seu *site* na Internet. Para conseguir isso era necessário conhecer uma linguagem de marcação de hipertexto, ou HTML (*HiperText Markup Language)*. Com esta linguagem relativamente simples, era possível mostrar ao internauta informações de um modo bastante apresentável, incluindo textos formatados em diversos padrões de letras e cores e até figuras e fotos. Foi fantástico! Começaram a surgir sites de todo tipo. Garotos que gostavam de mexer com computador aprendiam algumas *tags* básicas em HTML e logo se tornavam *Webmasters*. Seus *sites* com visual espalhafatoso ensinavam desde coisas corriqueiras como "técnicas de pedir aumento de mesada aos pais" até receitas assustadoras de "como construir uma bomba caseira". Foi um tempo de euforia. Tudo era novidade e, a cada novo recurso aprendido, os olhos da garotada brilhavam.

Ascensão e queda das empresas "Ponto com"

Com o passar do tempo, muitas empresas começaram a ver na Internet uma grande oportunidade de expandir seus negócios. Outros, vendo que algumas empresas totalmente baseadas na Internet estavam obtendo um tremendo êxito, resolveram arriscar tudo o que tinham em fabulosos projetos *on-line*. Foi outra euforia estrondosa. Algumas dessas empresas "ponto com" tiveram uma valorização impressionante, até mesmo inacreditável, quando grandes investidores começaram a aplicar nelas suas fortunas. A Internet parecia uma panacéia, onde todos os que entrassem ficariam ricos. Mas era evidente que estava havendo uma expectativa exagerada e mal orientada. A entrada do século 21 trouxe à tona esta triste realidade. Dezenas de impérios construídos sobre a Internet desmoronaram. Os grandes investidores arregalaram os olhos e começaram a pensar duas vezes antes de aplicar seu rico dinheirinho em projetos baseados puramente na Internet. Nesse meio tempo, alguns empreendimentos bem estruturados e ponderados conseguiam se firmar na Internet e obter sucesso. Em geral foram projetos que não se apoiaram naquela euforia inicial, mas conseguiram enxergar mais longe e encontrar aplicações realmente úteis e interessantes para a Internet.

Novas tecnologias para novos mercados

Conforme a Internet crescia a um ritmo espantoso, dezenas de novas tecnologias foram lançadas, provendo uma enorme gama de ferramentas tanto para os desenvolvedores de sites como para facilitar a vida dos internautas. *Browsers* cada vez mais poderosos incorporavam recursos dos mais diversos. Linguagens de *script* para a geração de páginas dinâmicas possibilitavam a integração de aplicações Web com grandes bases de dados, oferecendo diversas

facilidades e serviços aos usuários-consumidores, como consultar seu saldo bancário, fazer compras, reservar passagens em vôos, e muito mais, tudo na tela de seu próprio computador pessoal, via Internet. A era do *e-Business*, ou "comércio eletrônico" estava começando. O mercado via Internet destinado ao consumidor final ficou conhecido como B2C ou *Business To Consumer* ("Comércio ao Consumidor").

Indo um pouco mais além, as grandes corporações perceberam que podiam usar a Internet para se comunicar com seus parceiros e fornecedores, economizando uma quantia considerável de tempo e dinheiro, uma vez que as transações poderiam ser feitas *on-line*. O modelo de aplicações Web que fazia esse tipo de serviço foi chamado de B2B, ou *Business To Business* ("Comércio ao Comércio").

Mais recentemente ainda, o comércio eletrônico está encontrando um nicho muito atraente no chamado B2G, ou *Busines To Government* ("Comércio eletrônico voltado aos órgãos do Governo"). Muitas empresas de tecnologia estão se mobilizando para atender a esta nova demanda.

Em todas as aplicações construídas para a Internet, o HTML sempre foi a figura chave para a apresentação das informações na tela do usuário. Mesmo quando um determinado site faz uma consulta a uma base de dados, os servidores de aplicações web precisam formatar seus resultados em HTML para serem apresentado pelo *browser* do internauta. A Figura 1 ilustra uma aplicação web que consulta uma base de dados e mostra o resultado para o usuário. Como podemos ver, a página que chega ao *browser* está sempre no formato HTML.

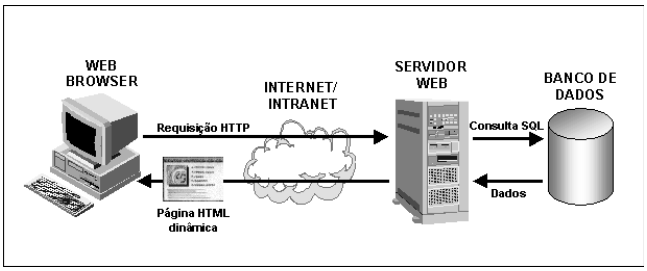

Figura 1 - Geração de uma página HTML dinâmica com consulta a um banco de dados.

Limitações do HTML

Com tudo o que vimos até agora, podemos concluir que o HTML é uma excelente linguagem para *exibir* informações ao usuário. Porém, quando avançamos para a criação de aplicações Web mais complexas, ele deixa muito a desejar.

Com o aumento no volume de transações via Internet, principalmente para atender às aplicações de *e-Commerce*, o setor de desenvolvimento para a Web começou a sentir falta de um recurso que agilizasse a transferência e manutenção de dados por este meio. Como dissemos o HTML cumpre bem o papel de *mostrar* os dados ao usuário de um modo apresentável. Porém, ela não foi concebida para manipular dados. Nesse aspecto, ela impõe as seguintes limitações:

- ✓ É uma tecnologia para apresentações, combinando dados com a forma de apresentá-los e tornando difícil uma separação dos dois elementos.
- ✓ Tem um conjunto fixo e rígido de marcações (*tags*) e não permite que você use suas próprias marcações nos seus programas aplicativos.
- ✓ É uma linguagem "plana" e não permite que você especifique uma hierarquia de dados, incluindo detalhes como limites e importância.
- ✓ Dificulta a transmissão de dados para o cliente, para que sejam processados posteriormente.
- ✓ É gerada continuamente pelo servidor e o cliente é apenas um mecanismo de exibição de dados.
- ✓ Fornece apenas um modo de visualização dos dados. Se quiser fornecer diferentes modos de visualização, você precisa refazer ou regerar os dados e a página HTML completa no servidor, para exibição nos clientes.
- ✓ Não é facilmente legível, nem pelo olho humano nem por máquina.
- ✓ Não é muito consistente. Por exemplo, algumas marcações, ou *tags*, exigem início e fim, como <html> e </html>, outras só têm início, como <p> e <b>. Desse modo, os analisadores de sintaxe HTML (existentes nos *browsers*) têm de saber lidar como essa formatação aleatória.

Surge o XML

O XML surgiu para suprir toda essa carência deixada pelo HTML no desenvolvimento de aplicações avançadas para a Internet, principalmente no que tange àquelas aplicações que precisam manipular um grande volume de dados.

Como vimos há pouco, quando uma aplicação Web tradicional precisa buscar informações em um banco de dados, ela normalmente usa um servidor de aplicativo para fazer uma consulta e retornar os dados no formato HTML para que estes possam ser corretamente exibidos no *browser* do usuário. Depois disso, pouco se pode fazer com os dados no *browser*, uma vez que eles estão embutidos na formatação HTML da página. Por outro lado, se os dados ou o conteúdo estivessem separados do formato de exibição, você poderia baixá-los para o cliente (*browser*) e depois utilizar vários modos de visualização, conforme necessário. Não seria esta uma boa idéia? Pois é exatamente isso o que o XML nos permite fazer.

Um dos objetivos por trás dessa linguagem é possibilitar a transferência e manipulação de dados através da Internet de modo fácil e consistente, de tal forma que qualquer tipo de aplicação, independentemente da

plataforma, sistema operacional, ou linguagem em que foi construída consiga manuseá-los.

Comparando XML com HTML

É somente natural que as pessoas tentem comparar o XML com o HTML, uma vez que ambas as linguagens são usadas para transferência de informações pela Internet. Por isso, acho que poderá ser de ajuda entendermos as diferenças e as semelhanças entre essas linguagens. Estes são alguns pontos bastante importantes:

- ✓ O XML não é uma substituição do HTML. Na realidade, o XML pode ser considerado como um *complemento* ao HTML. O XML e o HTML têm propósitos diferentes: o HTML é projetado para *exibir* dados e é focalizado em *como* os dados são apresentados, ao passo que o XML é projetado para *descrever* dados e é focalizado em *o que* os dados são.
- ✓ Assim como o HTML, o XML não faz nada. Ao passo que *tags* XML podem ser usadas para descrever a estrutura de um item, tal como uma ordem de compra, estas não podem conter qualquer código de programação que possa ser usado para enviar essa ordem de compra, processá-la, ou garantir que ela esteja preenchida. Outras pessoas têm de escrever código para executar realmente estas coisas com seus dados formatados em XML.
- ✓ Diferentemente do HTML, as *tags* XML são definidas pelo autor de um *schema*, ou documento, e são ilimitadas. Já as *Tags* HTML são predefinidas, e os desenvolvedores HTML podem usar somente as *tags* que são suportadas pelo padrão HTML atual.

Principais usos para o XML

O XML oferece um modo extremamente flexível para fazer transferência de dados. A lista seguinte apresenta os principais exemplos onde o XML pode ser usado:

- ✓ Um documento comum.
- ✓ Um registro estruturado, como um registro de compromisso ou pedido de compra.
- ✓ Aplicações Web (Internet/intranet) que transferem dados.
- ✓ Um objeto com dados, como o formato persistente de um objeto ou controle ActiveX.
- ✓ Um registro de dados, como o conjunto de resultados (*dataset*) de uma consulta SQL.
- ✓ Meta-conteúdo sobre um local de Web, como Formato de Definição de Canal (CDF).
- ✓ Apresentação gráfica, como a interface de usuário de uma aplicação.
- ✓ Ligações entre informações e pessoas na Web.

Vantagens e desvantagens do Formato XML

De alguns modos importantes, o XML é simplesmente um outro formato de dados. De outros modos, o XML apresenta várias vantagens fundamentais sobre outros formatos que o ajudaram a se distinguir como um meio para armazenar informações. Algumas dessas vantagens são as seguintes:

- ✓ O XML permite ao desenvolvedor criar suas próprias estruturas rotuladas para armazenar informações.
- ✓ O analisador XML (*parser*) é bem definido e extensamente-implementado, tornando possível recobrar informação de documentos XML em uma variedade de ambientes.
- ✓ O XML é construído em uma fundação de Unicode, tornando mais fácil a criação de documentos internacionalizados.
- ✓ As aplicações podem confiar em parsers de XML para fazer alguma validação estrutural, bem como verificação de tipos de dados (quando *schemas* XML são usados).
- ✓ Os formatos XML são baseados em texto, o que os tornam mais legíveis, mais fáceis de documentar, e às vezes mais fáceis de depurar.
- ✓ Ferramentas para o processamento XML estão disponíveis em plataformas diferentes. Isso faz com que seja mais simples usar XML em vez de formatos binários para trocar fluxos complexos de informação.
- ✓ Os documentos XML já podem usar muito da infra-estrutura construída para HTML, inclusive o protocolo HTTP e alguns *browsers*.

Apesar de todas essas vantagens porém, o XML não é apropriado para todas as situações. Documentos XML tendem a ser maiores que os formatos binários que eles substituem. Eles consomem maior largura de banda da rede e espaço de armazenamento, ou exigem maior tempo de processamento para compressão. Os *parsers* XML podem ser mais lentos que os *parsers* de formatos binários altamente aperfeiçoados e podem requerer mais memória. Porém, o projeto cuidadoso das aplicações pode prevenir alguns destes problemas.

Conhecendo um arquivo XML básico

Um arquivo XML simples contém basicamente um conjunto de descrição de dados. Veja um exemplo:

```
<!-- Arquivo clientes.xml -->

<?xml version = "1.0" encoding="ISO-8859-1" ?>
<doc>
  <clientes>
    <nome>Pâmela Pereira</nome>
    <telefone>(11) 5555-1234</telefone>
    <idade>2</idade>
  </clientes>
  <clientes>
    <nome>Giovana T. O. Pereira</nome>
    <telefone>(11) 5555-6789</telefone>
```

```
    <idade>25</idade>
  </clientes>
</doc>
```

Perceba que não é muito difícil entender o que o arquivo descreve. Observe também que as *tags* são totalmente personalizadas. Se você tentar abrir este arquivo no Internet Explorer 5.0 ou superior (que suporta XML), não verá algo muito diferente do que está listado aqui. A única coisa que o *browser* faz é colocar sinais de subtração (-) ou adição (+) ao lado de cada registro, permitindo que você os encolha ou expanda.

Definindo os dados com arquivos DTD

Se você prestar um pouco mais de atenção à listagem do arquivo clientes.xml mostrada no exemplo anterior, perceberá que não existe nada indicando o tipo dos dados apresentados. E embora você possa encontrar *tags* de campos como nome, telefone e idade, não existe uma definição desses dados. Este tipo de declaração não é realmente necessário. Porém, se você deseja assegurar a integridade e estruturar seu documento XML, é interessante acrescentar uma definição dos dados que ele contém. Isso pode ser feito por meio de um arquivo DTD (*Document Type Definition*, ou Definição de Tipo de Documento). O arquivo DTD define os dados, ou *elementos*, que o documento XML poderá conter. Isso permite que o processador de XML valide com precisão o conteúdo do arquivo XML. Para o arquivo clientes.xml do exemplo anterior, poderíamos ter o seguinte arquivo DTD simples associado:

```
<!-- Arquivo clientes.dtd -->

<!ELEMENT doc (clientes)>

<!ELEMENT clientes (nome,telefone,idade)>

<!ELEMENT nome (#PCDATA)>
<!ELEMENT telefone (#PCDATA)>
<!ELEMENT idade (#PCDATA)>
```

Este arquivo apenas define, de modo bastante simples, os elementos de dados que serão incluídos como *tags* no documento XML.

Formatando dados XML com arquivos XSL

Com os arquivos XML e DTD apresentados anteriormente, temos um conjunto de dados e a respectiva definição de sua estrutura. Mas ainda não temos uma visualização apresentável desses dados. Como podemos conseguir isso?

Um modo interessante é acrescentarmos um arquivo XSL ao conjunto. O arquivo XSL (*eXtensible Stylesheet Language*) nos permite formatar o conjunto de dados contido em um arquivo XML por meio de *tags* HTML. Para que um arquivo XSL consiga formatar um documento XML, é necessário acrescentar uma referência a ele dentro do arquivo XML. O código do exemplo seguinte poderia ser colocado no arquivo clientes.xml para referenciar um arquivo XSL que faça a sua formatação.

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

Depois de acrescentar a referência, podemos criar o arquivo XSL para finalmente formatar o documento. O exemplo a seguir ilustra um arquivo que formata o arquivo clientes.xml, apresentando seus dados em uma tabela HTML.

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

<HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<BODY>
 <TABLE BORDER="1" bordercolor="#000000"
        cellspacing="0" cellpadding="5">
  <xsl:for-each select="doc/clientes">
   <TR>
    <TD>
      <xsl:value-of select="nome"/>
    </TD>
    <TD>
      <xsl:value-of select="telefone"/>
    </TD>
    <TD>
      <xsl:value-of select="idade"/>
    </TD>
    </TR>
  </xsl:for-each>
 </TABLE>
</BODY>

</HTML>
```

Se você conhece um pouco de HTML, pode perceber que este documento mescla tags HTML com tags XSL (iniciadas com <xsl:). As *tags* <xsl:for-each...> e </xsl:for-each> são responsáveis por fazer um *loop* percorrendo todos os registros contidos no arquivo clientes.xml e formatando-os no *browser*. Se abrirmos o arquivo clientes.xml no *browser* agora, obteremos a seguinte apresentação:

Pâmela Pereira	(11) 5555-1234	2
Giovana Teodoro O. Pereira	(11) 5555-6789	25
Dani Edson Pereira	(11) 5555-6543	29

Conclusão

O XML pode mesmo ser um grande aliado no desenvolvimento de aplicações avançadas para a Internet. Vimos que ele não substitui o HTML, mas preenche uma grande carência desta linguagem, oferecendo um meio realmente eficiente de se transmitir dados de todo tipo através da rede mundial de computadores. Espero que este artigo o tenha ajudado ,de um modo definitivo, a entender o que é e para que serve realmente a linguagem XML.

*** Dani Edson Pereira** é analista de desenvolvimento sênior e autor de livros técnicos na área de informática. Seu último lançamento é ***Visual Basic.NET para Programadores***. Você pode contatá-lo pelo site http://daniedson.cjb.net.