XML

XML significa eXtensible Markup Language. XML foi projetado para armazenar e transportar dados. XML foi projetado para ser tanto humano e legível por máquina.

Como XML

<?xml version="1.0" encoding="UTF-8"?>  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>

Como Nota

**To:** Tove **From:** Jani **Heading:** Reminder **Body:** Don't forget me this weekend!

Por que estudar XML?

XML desempenha um papel importante em muitos sistemas de TI diferentes. O XML é freqüentemente usado para distribuir dados pela Internet. É importante (para todos os tipos de desenvolvedores de software!) ter uma boa compreensão de XML. O XML é uma ferramenta independente de software e hardware para armazenar e transportar dados.

O que é XML?

* XML significa eXtensible Markup Language
* XML é uma linguagem de marcação muito parecida com HTML
* XML foi projetado para armazenar e transportar dados
* XML foi projetado para ser auto-descritivo
* XML é uma recomendação do W3C

O XML acima é bastante auto-descritivo:

* Ele tem informações de remetente.
* Possui informações do receptor
* Tem um cabeçalho
* Tem um corpo de mensagem.

Mas ainda assim, o XML acima não faz nada. XML é apenas informações embrulhadas em tags. Alguém deve escrever um pedaço de software para enviá-lo, recebê-lo, armazená-lo e exibi-lo:

A diferença entre XML e HTML

O XML foi projetado para transportar dados - com foco nos dados. O HTML foi projetado para exibir dados - com foco em como os dados parecem. As tags XML não são predefinidas, como as tags HTML são. O XML não usa tags predefinidas. A linguagem XML não tem tags predefinidas.

As tags no exemplo acima (como <to> e <from>) não estão definidas em nenhum padrão XML. Essas tags são "inventadas" pelo autor do documento XML.

O HTML funciona com tags predefinidas como <p>, <h1>, <table>, etc. Com o XML, o autor deve definir tanto as tags como a estrutura do documento.

**XML é extensível**

A maioria dos aplicativos XML funcionará como esperado, mesmo que novos dados sejam adicionados (ou removidos). Imagine um aplicativo projetado para exibir a versão original do note.xml (<para> <from> <heading> <data>). Em seguida, imagine uma versão mais recente do note.xml com elementos <date> e <hour> adicionados e um <heading> removido. A forma como o XML é construído, versão mais antiga do aplicativo ainda pode funcionar: <note>  
  <date>2015-09-01</date>  
  <hour>08:30</hour>  
  <to>Tove</to>  
  <from>Jani</from>  
  <body>Don't forget me this weekend!</body>  
</note>

|  |  |
| --- | --- |
| **Versão antiga** | **Versão nova** |
| Nota | Nota |
| To: Tove | To: Tove |
| From: Jani | From: Jani |
| Head: (none) | Date: 18-05-217 14:30 |
| Don´t forget me this wekeend! | Don´t forget me this wekeend! |

XML simplifica as coisas

* Simplifica o compartilhamento de dados
* Simplifica o transporte de dados
* Simplifica as alterações da plataforma
* Simplifica a disponibilidade de dados

Muitos sistemas de computador contêm dados em formatos incompatíveis. Trocar dados entre sistemas incompatíveis (ou sistemas atualizados) é uma tarefa demorada para desenvolvedores web. Grandes quantidades de dados devem ser convertidas e dados incompatíveis são muitas vezes perdidos.

XML armazena dados em formato de texto sem formatação. Isso fornece uma maneira independente de software e hardware de armazenar, transportar e compartilhar dados. XML também facilita a expansão ou atualização para novos sistemas operacionais, novos aplicativos ou novos navegadores, sem perder dados. Com XML, os dados podem estar disponíveis para todos os tipos de "máquinas de leitura", como pessoas, computadores, máquinas de voz, feeds de notícias, etc.

Como XML pode ser usado?

O XML é usado em muitos aspectos do desenvolvimento da web. O XML é frequentemente usado para separar os dados da apresentação. XML separa dados da apresentação. O XML não contém nenhuma informação sobre como ser exibido.

Os mesmos dados XML podem ser usados em muitos cenários de apresentação diferentes. Devido a isso, com XML, há uma separação completa entre dados e apresentação. XML é muitas vezes um complemento para HTML

Em muitas aplicações HTML, XML é usado para armazenar ou transportar dados, enquanto HTML é usado para formatar e exibir os mesmos dados.

XML separa dados de HTML

Ao exibir dados em HTML, não é necessário editar o arquivo HTML quando os dados forem alterados. Com XML, os dados podem ser armazenados em arquivos XML separados. Com algumas linhas de código JavaScript, você pode ler um arquivo XML e atualizar o conteúdo de dados de qualquer página HTML.

**Estrutura de árvore XML**

Os documentos XML são formados como árvores de elementos. Uma árvore XML começa em um elemento raiz e se ramifica do elemento raiz para elemento filho. Todos os elementos podem ter sub-elementos (elementos filho): <root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>

Os termos pai, filho e irmão são usados para descrever as relações entre os elementos. Os pais têm filhos. As crianças têm pais. Os irmãos são crianças do mesmo nível (irmãos e irmãs). Todos os elementos podem ter conteúdo de texto (Harry Potter) e atributos (category = "cooking").

**Sintaxe Auto-Descritiva**

XML usa uma sintaxe auto-descritiva. Um prólogo define a versão XML ea codificação de caracteres: <?xml version="1.0" encoding="UTF-8**"**?>

A próxima linha é o elemento raiz do documento: <bookstore> A próxima linha inicia um elemento <book>: <book category="cooking"> Os elementos <book> têm 4 elementos filho: <title>, <author>, <year>, <price>: <title lang="en">Everyday Italian</title>  
<author>Giada De Laurentiis</author>  
<year>2005</year>  
<price>30.00</price> A próxima linha termina o elemento do livro: </book>

Regras de sintaxe XML

As regras de sintaxe do XML são muito simples e lógicas. As regras são fáceis de aprender e fáceis de usar. Documentos XML devem ter um elemento raiz. Os documentos XML devem conter um elemento raiz que seja o pai de todos os outros elementos.

O prolog de XML é opcional. Se existir, deve vir primeiro no documento. Os documentos XML podem conter caracteres internacionais, como øæå norueguês ou êèé francês. Para evitar erros, você deve especificar a codificação utilizada ou salvar seus arquivos XML como UTF-8. UTF-8 é a codificação de caracteres padrão para documentos XML. UTF-8 é também a codificação padrão para HTML5, CSS, JavaScript, PHP e SQL. Todos os elementos XML devem ter uma tag de fechamento Em HTML, alguns elementos podem funcionar bem, mesmo com uma tag de fechamento ausente. Em XML, é ilegal omitir a tag de fechamento. Todos os elementos devem ter uma tag de fechamento.

O prolog XML não tem uma tag de fechamento,isso não é um erro. O prólogo não faz parte do documento XML. As tags XML são sensíveis a maiúsculas e minúsculas. A tag <Letter> é diferente da tag <letter>.

Em XML, todos os elementos devem estar devidamente aninhados entre si: <b><i>This text is bold and italic</i></b> No exemplo acima, "devidamente aninhado" significa simplesmente que, uma vez que o elemento <i> é aberto dentro do elemento <b>, ele deve ser fechado dentro do elemento <b>.

Valores de atributo XML devem ser citados. Os elementos XML podem ter atributos em pares nome / valor, como em HTML. Em XML, os valores de atributo devem sempre ser citados.

INCORRETA:

<note date=12/11/2007>  
  <to>Tove</to>  
  <from>Jani</from>  
</note>

CORRETO:

<note date="12/11/2007">  
  <to>Tove</to>  
  <from>Jani</from> </note>

Referências de Entidade

Alguns caracteres têm um significado especial em XML. Se você colocar um caractere como "<" dentro de um elemento XML, ele gerará um erro porque o analisador interpreta-lo como o início de um novo elemento. Isso gerará um erro XML:<message>salary<1000</message>

Para evitar esse erro, substitua o caractere "<" por uma referência de entidade: <message>salary &lt; 1000</message>

Existem 5 referências de entidade predefinidas em XML:

|  |  |  |
| --- | --- | --- |
| &lt; | < | menor que |
| &gt; | > | maior que |
| &amp; | & | E comercial |
| &apos; | ' | apóstrofe |
| &quot; | " | aspa |

Somente < e & são estritamente ilegais em XML, mas é um bom hábito substituir > com &gt; também.

Comentários em XML

A sintaxe para escrever comentários em XML é semelhante à de HTML. <!-- This is a comment --> Dois traços no meio não são permitidos:<!-- This is a -- comment --> Estranho, mas permitido: <!-- This is a - - comment -->

O espaço em branco é preservado em XML. XML não trunca vários espaços em branco (o HTML trunca vários espaços em branco para um único espaço em branco):

|  |  |
| --- | --- |
| XML: | Hello           Tove |
| HTML: | Hello Tove |

**Elementos XML**

Um documento XML contém elementos XML. O que é um elemento XML? Um elemento XML é tudo, desde (inclusive) a tag de início do elemento até (incluindo) a tag final do elemento.

<price>29.99</price>

Um elemento pode conter:

* Texto
* Atributos
* Outros elementos
* Ou uma mistura do acima

<bookstore>  
  <book category="children">  
    <title>Harry Potter</title>  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
  <book category="web">  
    <title>Learning XML</title>  
    <author>Erik T. Ray</author>  
    <year>2003</year>  
    <price>39.95</price>  
  </book>  
</bookstore>

No exemplo acima: <Title>, <author>, <year> e <price> têm conteúdo de texto porque contêm texto (como 29.99). <Bookstore> e <book> têm conteúdo de elemento, porque contêm elementos. <Book> tem um atributo (category = "children").

Elementos XML vazios

Um elemento sem conteúdo é dito estar vazio. Em XML, você pode indicar um elemento vazio como este: <element></element> Você também pode usar uma tag de fechamento automático: <element /> As duas formas produzem resultados idênticos no software XML (Leitores, Navegadores). Elementos vazios podem ter atributos.

Regras de Nomenclatura XML

Os elementos XML devem seguir estas regras de nomenclatura:

* Os nomes dos elementos diferenciam maiúsculas de minúsculas
* Os nomes dos elementos devem começar com uma letra ou sublinhado
* Os nomes dos elementos não podem começar com as letras xml (ou XML, ou Xml, etc.)
* Os nomes dos elementos podem conter letras, dígitos, hifens, sublinhados e períodos
* Os nomes dos elementos não podem conter espaços
* Qualquer nome pode ser usado, nenhuma palavra é reservada (exceto xml).

**Melhores Práticas de Nomeação**

* Crie nomes descritivos, como este: <person>, <firstname>, <lastname>.
* Crie nomes curtos e simples, como este: <book\_title> não assim: <the\_title\_of\_the\_book>.
* Evite "-". Se você nomear algo "first-name", algum software pode pensar que você quer subtrair "nome" de "primeiro".
* Evite ".". Se você nomear algo "first.name", algum software pode pensar que "name" é uma propriedade do objeto "first".
* Evite ":". Doispontos são reservados para namespaces.
* Letras não inglesas como éòá ​​são perfeitamente legais em XML, mas atente para problemas se o software não os suportar.

Elementos XML podem ter atributos, assim como HTML. Os atributos são projetados para conter dados relacionados a um elemento específico. Atributos XML devem ser citados. Podem ser utilizadas aspas simples ou duplas.

Para o gênero de uma pessoa, o elemento <person> pode ser escrito da seguinte forma: <person gender="female"> ou <person gender='female'>

Se o valor do atributo em si contém aspas duplas, você pode usar aspas simples, como neste exemplo: <gangster name='George "Shotgun" Ziegler'>

Ou você pode usar entidades de caracteres: <gangster name="George &quot;Shotgun&quot; Ziegler">

|  |  |
| --- | --- |
| <person gender="female">   <firstname>Anna</firstname>   <lastname>Smith</lastname> </person> | <person>   <gender>female</gender>   <firstname>Anna</firstname>   <lastname>Smith</lastname> </person> |

No primeiro exemplo, gender é um atributo. No segundo, o gênero é um elemento. Ambos os exemplos fornecem as mesmas informações. Não há regras sobre quando usar atributos ou quando usar elementos em XML.

Os seguintes três documentos XML contêm exatamente as mesmas informações:

|  |  |  |
| --- | --- | --- |
| <note date="2008-01-10">   <to>Tove</to>   <from>Jani</from> </note> | <note>   <date>2008-01-10</date>   <to>Tove</to>   <from>Jani</from> </note> | <note>   <date>     <year>2008</year>     <month>01</month>     <day>10</day>   </date>   <to>Tove</to>   <from>Jani</from> </note> |

Evitar atributos XML? Algumas coisas a considerar ao usar atributos são:

* Atributos não podem conter vários valores (elementos podem)
* Atributos não podem conter estruturas de árvore (elementos podem)
* Atributos não são facilmente expansíveis (para futuras alterações)

Não acabe assim: <note day="10" month="01" year="2008"  
 to="Tove" from="Jani" heading="Reminder"  
 body="Don't forget me this weekend!">  
 </note>

Atributos XML para metadados

Às vezes, as referências de ID são atribuídas a elementos. Esses IDs podem ser usados para identificar elementos XML da mesma forma que o atributo id em HTML. Este exemplo demonstra isso: <messages>  
   <note id="501">  
     <to>Tove</to>  
     <from>Jani</from>  
     <heading>Reminder</heading>  
     <body>Don't forget me this weekend!</body>  
   </note>  
   <note id="502">  
     <to>Jani</to>  
     <from>Tove</from>  
     <heading>Re: Reminder</heading>  
     <body>I will not</body>  
   </note>  
 </messages>

Os atributos de id acima são para identificar diferentes <note>. Não é uma parte da <note> em si. Ometadados (dados sobre dados) devem ser armazenados como atributos, e os dados em si devem ser armazenados como elementos.

**XML Namespaces**

Os Namespaces XML fornecem um método para evitar conflitos de nomes de elementos.

Conflitos de Nome

Em XML, os nomes dos elementos são definidos pelo desenvolvedor. Isso muitas vezes resulta em um conflito ao tentar misturar documentos XML de diferentes aplicativos XML. Esse XML carrega informações de tabela HTML: <table>  
  <tr>  
    <td>Apples</td>  
    <td>Bananas</td>  
  </tr>  
</table>

Este XML traz informações sobre uma mesa (um móvel): <table>  
  <name>African Coffee Table</name>  
  <width>80</width>  
  <length>120</length>  
</table>

Se esses fragmentos XML foram adicionados juntos, haveria um conflito de nome. Ambos contêm um elemento <table>, mas os elementos têm conteúdo e significado diferentes. Um usuário ou um aplicativo XML não saberá como lidar com essas diferenças.

**Resolvendo o conflito de nomes usando um prefixo**

Os conflitos de nomes em XML podem ser facilmente evitados usando um prefixo de nome. Este XML traz informações sobre uma tabela HTML e um móvel: <h:table>  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>  
  
<f:table>  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table> No exemplo acima, agora os dois elementos <table> têm nomes diferentes, sem comflito.

Namespaces XML - O atributo xmlns

Ao usar prefixos em XML, um namespace para o prefixo deve ser definido. O namespace pode ser definido por um atributo xmlns na marca inicial de um elemento. A declaração de espaço de nome tem a seguinte sintaxe. Xmlns: prefix = "URI". <root>  
<h:table xmlns:h="http://www.w3.org/TR/html4/">  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>  
  
<f:table xmlns:f="https://www.w3schools.com/furniture">  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table>  
</root>

No exemplo acima: O atributo xmlns no primeiro elemento <table> fornece o prefixo h: um namespace qualificado. O atributo xmlns no segundo elemento <table> fornece o prefixo f: um namespace qualificado.

Quando um namespace é definido para um elemento, todos os elementos filho com o mesmo prefixo são associados com o mesmo namespace. Namespaces também podem ser declarados no elemento raiz XML: <root  
xmlns:h="http://www.w3.org/TR/html4/"  
xmlns:f="https://www.w3schools.com/furniture">  
  
<h:table>  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>  
  
<f:table>  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table>  
</root>

**Nota**: O URI do namespace não é usado pelo analisador para procurar informações. A finalidade de usar um URI é dar o namespace um nome exclusivo. No entanto, as empresas geralmente usam o namespace como um ponteiro para uma página da Web que contém informações de namespace.

Uniform Resource Identifier (URI)

Um Uniform Resource Identifier (URI) é uma seqüência de caracteres que identifica um recurso da Internet. O URI mais comum é o Uniform Resource Locator (URL) que identifica um endereço de domínio da Internet. Outro tipo, não tão comum de URI é o Universal Resource Name (URN).

**Namespaces padrão**

Definir um namespace padrão para um elemento nos poupa de usar prefixos em todos os elementos filho. Tem a seguinte sintaxe: xmlns="*namespaceURI*"

Namespaces em uso real

XSLT é uma linguagem que pode ser usada para transformar documentos XML em outros formatos. O documento XML abaixo, é um documento usado para transformar XML em HTML. O namespace "http://www.w3.org/1999/XSL/Transform" identifica elementos XSLT dentro de um documento HTML:

<?xml version="1.0" encoding="UTF-8**"**?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
<html>  
<body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr>  
      <th style="text-align:left">Title</th>  
      <th style="text-align:left">Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
    <tr>  
      <td><xsl:value-of select="title"/></td>  
      <td><xsl:value-of select="artist"/></td>  
    </tr>  
    </xsl:for-each>  
  </table>  
</body>  
</html>  
</xsl:template>  
</xsl:stylesheet>

**Exibindo XML**

Os arquivos XML brutos podem ser visualizados em todos os principais navegadores. Não espere que os arquivos XML sejam exibidos como páginas HTML. A maioria dos navegadores exibirá um documento XML com elementos codificados por cores.

Muitas vezes um sinal de mais (+) ou menos (-) à esquerda dos elementos pode ser clicado para expandir ou contrair a estrutura do elemento.Para exibir fonte XML bruta, tente selecionar "Exibir origem da página" ou "Exibir origem" no menu do navegador.

**Nota:** No Safari 5 (e anteriores), somente o texto do elemento será exibido. Para exibir o XML bruto, você deve clicar com o botão direito na página e selecionar "Exibir código-fonte".

Exibindo um arquivo XML inválido

Se um arquivo XML errado for aberto, alguns navegadores irão relatar o erro e alguns o exibirão ou exibirão incorretamente. Por que o XML é exibido sem forma?

Documentos XML não carregam informações sobre como exibir os dados. Como as tags XML são "inventadas" pelo autor do documento XML, os navegadores não sabem se uma tag como <tabela> descreve uma tabela HTML ou uma tabela de jantar.

Sem qualquer informação sobre como exibir os dados, os navegadores podem simplesmente exibir o documento XML como ele é. A formatação de XML com CSS não é recomendada. Use JavaScript ou XSLT em vez disso.

**XML HttpRequest**

O objeto XMLHttpRequest

O objeto XMLHttpRequest pode ser usado para solicitar dados de um servidor web. O objeto XMLHttpRequest é um sonho de desenvolvedores, porque você pode:

* Atualizar uma página da Web sem recarregar a página
* Solicitar dados de um servidor - depois que a página foi carregada
* Receber dados de um servidor - depois que a página foi carregada
* Enviar dados para um servidor - em segundo plano

Exemplo XMLHttpRequest

Quando você digita um caractere no campo de entrada abaixo, um XMLHttpRequest é enviado para o servidor e algumas sugestões de nome são retornadas (do servidor).

Enviando um XMLHttpRequest

Uma sintaxe JavaScript comum para usar o objeto XMLHttpRequest se parece muito com isto: var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
       //Ação típica a ser executada quando o documento está pronto  
       document.getElementById("demo").innerHTML = xhttp.responseText;  
    }  
};  
xhttp.open("GET", "filename", true);  
xhttp.send();

**Parser XML**

Todos os principais navegadores têm um analisador XML embutido para acessar e manipular XML. O XML DOM (Document Object Model) define as propriedades e métodos para acessar e editar XML. No entanto, antes de um documento XML poder ser acessado, ele deve ser carregado em um objeto XML DOM.

Analisando uma Cadeia de Texto

Este exemplo analisa uma seqüência de caracteres de texto em um objeto XML DOM e extrai as informações dele com JavaScript: <html>  
<body>  
<p id="demo"></p>

<script>  
var text, parser, xmlDoc;  
  
text = "<bookstore><book>" + //Uma string de texto é definida  
"<title>Everyday Italian</title>" +  
"<author>Giada De Laurentiis</author>" +  
"<year>2005</year>" +  
"</book></bookstore>";  
  
parser = new DOMParser(); //Um analisador XML DOM é criado  
xmlDoc = parser.parseFromString(text,"text/xml"); //O analisador cria um novo objeto XML DOM usando a seqüência de texto:  
  
document.getElementById("demo").innerHTML =  
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;  
</script>  
  
</body>  
</html>

O objeto XMLHttpRequest

O objeto XMLHttpRequest possui um XML Parser incorporado. A propriedade responseText retorna a resposta como uma seqüência de caracteres. A propriedade responseXML retorna a resposta como um objeto XML DOM. Se você quiser usar a resposta como um objeto XML DOM, você pode usar a propriedade response XML.

**XML e XPath**

XPath é um elemento importante no padrão XSLT. XPath pode ser usado para navegar através de elementos e atributos em um documento XML.

* XPath é uma sintaxe para definir partes de um documento XML
* XPath usa expressões de caminho para navegar em documentos XML
* XPath contém uma biblioteca de funções padrão
* XPath é um elemento principal no XSLT e no Xquery
* XPath é uma recomendação do W3C

Expressões de caminho XPath

O XPath usa expressões de caminho para selecionar nós ou conjuntos de nós em um documento XML. Essas expressões de caminho parecem muito com as expressões que você vê quando trabalha com um sistema de arquivos de computador tradicional.

Expressões XPath podem ser usadas em JavaScript, Java, XML Schema, PHP, Python, C e C ++ e muitas outras linguagens. XPath é usado em XSLT. XPath é um elemento importante no padrão XSLT. Com o conhecimento do XPath você será capaz de tirar grande vantagem do XSL.

Usaremos o seguinte documento XML: <?xml version="1.0" encoding="UTF-8"?>  
<bookstore>  
  
<book category="cooking">  
  <title lang="en">Everyday Italian</title>  
  <author>Giada De Laurentiis</author>  
  <year>2005</year>  
  <price>30.00</price>  
</book>  
  
<book category="children">  
  <title lang="en">Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>  
  
<book category="web">  
  <title lang="en">XQuery Kick Start</title>  
  <author>James McGovern</author>  
  <author>Per Bothner</author>  
  <author>Kurt Cagle</author>  
  <author>James Linn</author>  
  <author>Vaidyanathan Nagarajan</author>  
  <year>2003</year>  
  <price>49.99</price>  
</book>  
  
<book category="web">  
  <title lang="en">Learning XML</title>  
  <author>Erik T. Ray</author>  
  <year>2003</year>  
  <price>39.95</price>  
</book>  
  
</bookstore>

Na tabela abaixo apresentamos algumas expressões XPath eo resultado das expressões:

|  |  |
| --- | --- |
| /bookstore/book[1]  /bookstore/book[last()]  /bookstore/book[last()-1]  /bookstore/book[position()<3]  //title[@lang]  //title[@lang='en']  /bookstore/book[price>35.00]  /bookstore/book[price>35.00]/title | Seleciona o primeiro elemento de livro que é filho do elemento de livraria  Seleciona o último elemento de livro que é filho do elemento de livraria  Seleciona o último mas um elemento de livro que é o filho do elemento de livraria  Seleciona os dois primeiros elementos do livro que são filhos do elemento da livraria  Seleciona todos os elementos de título que possuem um atributo chamado lang  Seleciona todos os elementos do título que possuem um atributo "lang" com um valor de "en"  Seleciona todos os elementos de livro do elemento de livraria que têm um elemento de preço com um valor maior que 35,00  Seleciona todos os elementos de título dos elementos de livro do elemento de livraria que têm um elemento de preço com um valor maior que 35,00 |

**XML e XSLT**

Com XSLT você pode transformar um documento XML em HTML.

Exibindo XML com XSLT

XSLT (eXtensible Stylesheet Language Transformações) é a linguagem de folha de estilo recomendada para XML.

XSLT é muito mais sofisticado do que CSS. Com XSLT você pode adicionar / remover elementos e atributos para ou do arquivo de saída. Você também pode reorganizar e classificar elementos, realizar testes e tomar decisões sobre quais elementos ocultar e exibir, e muito mais. XSLT usa XPath para localizar informações em um documento XML. Usaremos o seguinte documento XML: <?xml version="1.0" encoding="UTF-8"?>  
<breakfast\_menu>  
  
<food>  
<name>Belgian Waffles</name>  
<price>$5.95</price>  
<description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>  
<calories>650</calories>  
</food>  
  
<food>  
<name>Strawberry Belgian Waffles</name>  
<price>$7.95</price>  
<description>Light Belgian waffles covered with strawberries and whipped cream</description>  
<calories>900</calories>  
</food>  
  
<food>  
<name>Berry-Berry Belgian Waffles</name>  
<price>$8.95</price>  
<description>Light Belgian waffles covered with an assortment of fresh berries and whipped cream</description>  
<calories>900</calories>  
</food>  
  
<food>  
<name>French Toast</name>  
<price>$4.50</price>  
<description>Thick slices made from our homemade sourdough bread</description>  
<calories>600</calories>  
</food>  
  
<food>  
<name>Homestyle Breakfast</name>  
<price>$6.95</price>  
<description>Two eggs, bacon or sausage, toast, and our ever-popular hash browns</description>  
<calories>950</calories>  
</food>  
  
</breakfast\_menu>

Use XSLT para transformar XML em HTML, antes que ele seja exibido em um navegador:

<?xml version="1.0" encoding="UTF-8"?>  
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
<body style="font-family:Arial;font-size:12pt;background-color:#EEEEEE">  
<xsl:for-each select="breakfast\_menu/food">  
  <div style="background-color:teal;color:white;padding:4px">  
    <span style="font-weight:bold"><xsl:value-of select="name"/> - </span>  
    <xsl:value-of select="price"/>  
    </div>  
  <div style="margin-left:20px;margin-bottom:1em;font-size:10pt">  
    <p>  
    <xsl:value-of select="description"/>  
    <span style="font-style:italic"> (<xsl:value-of select="calories"/> calories per serving)</span>  
    </p>  
  </div>  
</xsl:for-each>  
</body>  
</html>

**XML e Xquery**

XQuery é para XML o que SQL é para bancos de dados. O XQuery foi projetado para consultar dados XML: for $x in doc("books.xml")/bookstore/book  
where $x/price>30  
order by $x/title  
return $x/title

* XQuery é a linguagem de consulta de dados XML
* XQuery para XML é como SQL para bancos de dados
* XQuery é construído em expressões Xpath
* O XQuery é suportado por todos os principais bancos de dados
* XQuery é uma recomendação do W3C

XQuery é sobre consulta XML

XQuery é uma linguagem para encontrar e extrair elementos e atributos de documentos XML. Aqui está um exemplo do que o XQuery poderia resolver: "Selecione todos os registros de CD com um preço inferior a US $ 10 da coleção CD armazenada em cd\_catalog.xml"

XQuery e XPath

XQuery 1.0 e XPath 2.0 compartilham o mesmo modelo de dados e suportam as mesmas funções e operadores.

XQuery pode ser usado para:

* Extrair informações para usar em um serviço da Web
* Gerar relatórios de resumo
* Transforme dados XML em XHTML
* Pesquisar documentos da Web para obter informações relevantes
* XQuery é uma recomendação do W3C

O XQuery é compatível com vários padrões do W3C, como XML, Namespaces, XSLT, XPath e XML Schema. O XQuery 1.0 tornou-se uma Recomendação do W3C em 2007.

**XML, XLink e Xpointer**

XLink é usado para criar hiperlinks em documentos XML.

* XLink é usado para criar hiperlinks dentro de documentos XML
* Qualquer elemento em um documento XML pode se comportar como um link
* Com o XLink, os links podem ser definidos fora dos arquivos vinculados
* XLink é uma recomendação do W3C

Suporte do navegador XLink

Não existe suporte de navegador para XLink em documentos XML. No entanto, todos os principais navegadores suportam XLinks em SVG.

**Sintaxe XLink**

Em HTML, o elemento <a> define uma hiperligação. No entanto, isso não é como ele funciona em XML. Em documentos XML, você pode usar qualquer nome de elemento que desejar - portanto, é impossível para os navegadores prever quais elementos de link serão chamados em documentos XML. Abaixo está um exemplo simples de como usar o XLink para criar links em um documento XML: <?xml version="1.0" encoding="UTF-8"?>  
<homepages xmlns:xlink="http://www.w3.org/1999/xlink">  
  <homepage xlink:type="simple" xlink:href="https://www.w3schools.com">Visit W3Schools</homepage>  
  <homepage xlink:type="simple" xlink:href="http://www.w3.org">Visit W3C</homepage>  
</homepages>

Para acessar os recursos do XLink, devemos declarar o namespace XLink. O namespace XLink é: "http://www.w3.org/1999/xlink". O tipo xlink: e os atributos xlink: href nos elementos <homepage> vêm do espaço de nomes XLink. O xlink: type = "simple" cria um simples link "HTML-like" (significa "clique aqui para ir para lá"). O atributo xlink: href especifica o URL a ser vinculado.

O seguinte documento XML contém recursos XLink: <?xml version="1.0" encoding="UTF-8"?>  
<bookstore xmlns:xlink="http://www.w3.org/1999/xlink">  
<book title="Harry Potter">  
  <description  
  xlink:type="simple"  
  xlink:href="/images/HPotter.gif"  
  xlink:show="new">  
  As his fifth year at Hogwarts School of Witchcraft and  
  Wizardry approaches, 15-year-old Harry Potter is.......  
  </description>  
</book>  
<book title="XQuery Kick Start">  
  <description  
  xlink:type="simple"  
  xlink:href="/images/XQuery.gif"  
  xlink:show="new">  
  XQuery Kick Start delivers a concise introduction  
  to the XQuery standard.......  
  </description>

</book>  
</bookstore>

O espaço para nome XLink é declarado na parte superior do documento (xmlns: xlink = "http://www.w3.org/1999/xlink") O xlink: type = "simple" cria um simples link "HTML-like" O atributo xlink: href especifica a URL a ser vinculada (neste caso - uma imagem) O xlink: show = "novo" especifica que o link deve abrir em uma nova janela

XLink - Indo mais longe

No exemplo acima demonstramos simples XLinks. XLink está ficando mais interessante ao acessar locais remotos como recursos, em vez de páginas autônomas. Se definimos o valor do atributo xlink: show como "**embed**", o recurso vinculado deve ser processado inline dentro da página. Quando você considera que este poderia ser outro documento XML você poderia, por exemplo, construir uma hierarquia de documentos XML. Você também pode especificar quando o recurso deve aparecer, com o atributo **xlink: actuate**.

Referência de atributo XLink

|  |  |
| --- | --- |
| **Valor** | **Descrição** |
| onLoad onRequest other none | Define quando o recurso vinculado é lido e mostrado:  OnLoad - o recurso deve ser carregado e mostrado quando o documento é carregado  OnRequest - o recurso não é lido ou mostrado antes que o link seja clicado |

xlink:actuate

|  |  |
| --- | --- |
| **Valor** | **Descrição** |
| *URL* | Especifica o URL para vincular a |

xlink:href

|  |  |
| --- | --- |
| **Valor** | **Descrição** |
| embed new replace other none | Especifica onde abrir o link. O padrão é "substituir" |

xlink:show

|  |  |
| --- | --- |
| **Valor** | **Descrição** |
| simple extended locator arc resource title none | Especifica o tipo de link |

xlink:type

**Xpointer**

* O XPointer permite que os links apontem para partes específicas de um XML file
* XPointer usa expressões XPath para navegar no documento XML
* XPointer é uma recomendação do W3C

Suporte ao navegador XPointer

Não existe suporte de navegador para o XPointer. Mas o XPointer é usado em outras linguagens XML. Neste exemplo, usaremos XPointer em conjunto com XLink para apontar para uma parte específica de outro documento. Começaremos examinando o documento XML de destino (o documento ao qual estamos vinculados): <?xml version="1.0" encoding="UTF-8"?>  
<dogbreeds>  
  
<dog breed="Rottweiler" id="Rottweiler">  
  <picture url="https://dog.com/rottweiler.gif" />  
  <history>The Rottweiler's ancestors were probably Roman  
  drover dogs.....</history>  
  <temperament>Confident, bold, alert and imposing, the Rottweiler  
  is a popular choice for its ability to protect....</temperament>  
</dog>  
  
<dog breed="FCRetriever" id="FCRetriever">  
  <picture url="https://dog.com/fcretriever.gif" />  
  <history>One of the earliest uses of retrieving dogs was to  
  help fishermen retrieve fish from the water....</history>  
  <temperament>The flat-coated retriever is a sweet, exuberant,  
  lively dog that loves to play and retrieve....</temperament>  
</dog>  
</dogbreeds>

Observe que o documento XML acima usa atributos id em cada elemento! Assim, em vez de vincular a todo o documento (como com XLink), XPointer permite que você vincule a partes específicas do documento. Para vincular a uma parte específica de uma página, adicione um sinal de número (#) e uma expressão XPointer após a URL no atributo xlink: href, como este: xlink: href = "https://dog.com/dogbreeds.xml #xpointer (id ('Rottweiler')) ". A expressão refere-se ao elemento no documento de destino, com o valor id de "Rottweiler".

XPointer também permite um método abreviado para vincular a um elemento com um id. Você pode usar o valor do id diretamente, como este: xlink: href = "https://dog.com/dogbreeds.xml#Rottweiler".

O seguinte documento XML contém links para mais informações da raça do cão para cada um dos meus cães: <?xml version="1.0" encoding="UTF-8"?>  
<mydogs xmlns:xlink="http://www.w3.org/1999/xlink">  
  
<mydog>  
  <description>  
  Anton is my favorite dog. He has won a lot of.....  
  </description>  
  <fact xlink:type="simple" xlink:href="https://dog.com/dogbreeds.xml#Rottweiler">  
  Fact about Rottweiler   
  </fact>  
</mydog>  
  
<mydog>  
  <description>  
  Pluto is the sweetest dog on earth......  
  </description>  
  <fact xlink:type="simple" xlink:href="https://dog.com/dogbreeds.xml#FCRetriever">  
  Fact about flat-coated Retriever   
  </fact>  
</mydog>  
</mydogs>

**XML Validator**

Documentos XML Bem Formados. Um documento XML com sintaxe correta é chamado de "Bem Formado":

* Os documentos XML devem ter um elemento raiz
* Os elementos XML devem ter uma tag de fechamento
* As tags XML são sensíveis a maiúsculas e minúsculas
* Os elementos XML devem estar devidamente aninhados
* Os valores de atributos XML devem ser citados

<?xml version="1.0" encoding="UTF-8**"**?>  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
</note>

Erros XML irá parar você. Erros em documentos XML interromperão seus aplicativos XML. A especificação W3C XML especifica que um programa deve parar de processar um documento XML se encontrar um erro. A razão é que o software XML deve ser pequeno, rápido e compatível. Os navegadores HTML têm permissão para exibir documentos HTML com erros (como tags de extremidade ausentes). Com o XML, os erros não são permitidos.

Documentos XML válidos

Um documento XML "bem formado" não é o mesmo que um documento XML "válido".

Um documento XML "válido" deve ser bem formado. Além disso, ele deve estar em conformidade com uma definição de tipo de documento. Existem duas definições de tipos de documentos diferentes que podem ser usadas com XML:

* DTD - Definição do tipo de documento original
* XML Schema - Uma alternativa baseada em XML para DTD

Uma definição de tipo de documento define as regras e os elementos e atributos legais para um documento XML.

**XML DTD**

Um documento XML com sintaxe correta é chamado de "Bem Formado". Um documento XML validado contra uma DTD é tanto "Bem Formado" e "Válido".

Documentos XML válidos

Um documento XML "válido" é um documento XML "Well Formed", que também está em conformidade com as regras de uma DTD: <?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE note SYSTEM "Note.dtd">  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
</note>

A declaração DOCTYPE, no exemplo acima, é uma referência a um arquivo DTD externo.

XML DTD

A finalidade de uma DTD é definir a estrutura de um documento XML. Define a estrutura com uma lista de elementos legais: <!DOCTYPE note  
[  
<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>

O DTD acima é interpretado como este:

* ! DOCTYPE nota define que o elemento raiz do documento é nota
* ! ELEMENT nota define que o elemento de nota deve conter os elementos: "para, de, título, corpo"
* ! ELEMENT para definir o elemento to para ser do tipo "#PCDATA"
* ! ELEMENT de define o elemento from para ser do tipo "#PCDATA"
* ! O cabeçalho ELEMENT define o elemento de título como sendo do tipo "#PCDATA"
* ! ELEMENT corpo define o corpo elemento para ser do tipo "#PCDATA"

#PCDATA significa dados de texto analisáveis.

Usando DTD para Declaração de Entidade

Uma declaração doctype também pode ser usada para definir caracteres especiais e seqüências de caracteres, usados no documento: <?xml version="1.0" encoding="UTF-8"?>  
  
<!DOCTYPE note [  
<!ENTITY nbsp "&#xA0;">   
<!ENTITY writer "Writer: Donald Duck.">  
<!ENTITY copyright "Copyright: W3Schools.">  
]>  
  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
<footer>&writer;&nbsp;&copyright;</footer>  
</note>

Uma entidade tem 3 partes: um e comercial (&), um nome de entidade e um ponto e vírgula (;).

Quando usar um DTD / Esquema?

Com uma DTD, grupos independentes de pessoas podem concordar em usar uma DTD padrão para troca de dados. Com um DTD, você pode verificar se os dados recebidos do mundo externo são válidos. Você também pode usar um DTD para verificar seus próprios dados. Se você quer estudar DTD, por favor leia nosso Tutorial de DTD.

Quando NÃO usar um DTD / Esquema?

O XML não requer um DTD / Esquema. Quando você está experimentando com XML, ou quando você está trabalhando com pequenos arquivos XML, criar DTDs pode ser um desperdício de tempo. Se você desenvolver aplicativos, aguarde até que a especificação seja estável antes de adicionar uma definição de documento. Caso contrário, o software poderá parar de funcionar devido a erros de validação.

**Schema XML**

Um esquema XML descreve a estrutura de um documento XML, tal como uma DTD. Um documento XML com sintaxe correta é chamado de "Bem Formado". Um documento XML validado contra um Esquema XML é tanto "Bem Formado" e "Válido".

Esquema XML

XML Schema é uma alternativa baseada em XML para DTD: <xs:element name="note">  
 <xs:complexType>  
   <xs:sequence>  
     <xs:element name="to" type="xs:string"/>  
     <xs:element name="from" type="xs:string"/>  
     <xs:element name="heading" type="xs:string"/>  
     <xs:element name="body" type="xs:string"/>  
   </xs:sequence>  
 </xs:complexType>  
</xs:element>

O esquema acima é interpretado como este:

* <Xs: element name = "note"> define o elemento chamado "note"
* <Xs: complexType> o elemento "note" é um tipo complexo
* <Xs: sequência> o tipo complexo é uma seqüência de elementos
* <Xs: element name = "para" type = "xs: string"> o elemento "to" é do tipo string (text)
* <Xs: element name = "from" type = "xs: string"> o elemento "from" é do tipo string
* <Xs: element name = "heading" type = "xs: string"> o elemento "heading" é do tipo string
* <Xs: element name = "body" = "xs: string"> o elemento "body" é do tipo string

Esquemas XML são mais poderosos do que DTD

* XML Schemas são escritos em XML
* XML Schemas são extensíveis a adições
* XML Schemas suporta tipos de dados
* XML Schemas suporta namespaces

Por que usar um esquema XML?

Com XML Schema, seus arquivos XML podem conter uma descrição de seu próprio formato. Com o XML Schema, grupos independentes de pessoas podem concordar com um padrão para troca de dados. Com XML Schema, você pode verificar dados.

Tipos de dados de suporte de esquemas XML

Uma das maiores vantagens dos XML Schemas é o suporte a tipos de dados:

* É mais fácil descrever o conteúdo do documento
* É mais fácil definir restrições de dados
* É mais fácil validar a exactidão dos dados
* É mais fácil converter dados entre diferentes tipos de dados

XML Schemas usa sintaxe XML

Outra grande força sobre XML Schemas é que eles são escritos em XML:

* Você não precisa aprender uma nova língua
* Você pode usar o editor XML para editar seus arquivos de Esquema
* Você pode usar seu analisador XML para analisar seus arquivos de esquema
* Você pode manipular seus Esquemas com o XML DOM
* Você pode transformar seus Esquemas com XSLT

**XML no servidor**

Arquivos XML são arquivos de texto simples, como arquivos HTML. XML pode facilmente ser armazenado e gerado por um servidor web padrão.

Armazenando Arquivos XML no Servidor

Os arquivos XML podem ser armazenados em um servidor de Internet exatamente da mesma forma que os arquivos HTML. Inicie o Bloco de notas do Windows e escreva as seguintes linhas: <?xml version="1.0" encoding="UTF-8"?>  
<note>  
  <from>Jani</from>  
  <to>Tove</to>  
  <message>Remember me this weekend</message>  
</note> **Salve o arquivo em seu servidor web com um nome próprio como "note.xml".**

Gerando XML com PHP

O XML pode ser gerado em um servidor sem qualquer software XML instalado. Para gerar uma resposta XML a partir do servidor usando PHP, use o seguinte código: <?php  
 header("Content-type: text/xml");  
 echo "<?xml version='1.0' encoding='UTF-8**'**?>";  
 echo "<note>";  
 echo "<from>Jani</from>";  
 echo "<to>Tove</to>";  
 echo "<message>Remember me this weekend</message>";  
 echo "</note>";  
?> Observe que o tipo de conteúdo do cabeçalho de resposta deve ser definido como "text / xml".

Gerando XML com ASP

Para gerar uma resposta XML a partir do servidor - basta escrever o seguinte código e salvá-lo como um arquivo ASP no servidor web: <%  
 response.ContentType="text/xml"  
 response.Write("<?xml version='1.0' encoding='UTF-8**'**?>")  
 response.Write("<note>")  
 response.Write("<from>Jani</from>")  
 response.Write("<to>Tove</to>")  
 response.Write("<message>Remember me this weekend</message>")  
 response.Write("</note>")  
%>

Gerando XML a partir de um banco de dados

XML pode ser gerado a partir de um banco de dados sem qualquer software XML instalado. Para gerar uma resposta de banco de dados XML do servidor, basta escrever o seguinte código e salvá-lo como um arquivo ASP no servidor da Web: <%  
response.ContentType = "text/xml"  
set conn=Server.CreateObject("ADODB.Connection")  
conn.provider="Microsoft.Jet.OLEDB.4.0;"  
conn.open server.mappath("/datafolder/database.mdb")  
  
sql="select fname,lname from tblGuestBook"  
set rs=Conn.Execute(sql)  
  
response.write("<?xml version='1.0' encoding='UTF-8**'**?>")  
response.write("<guestbook>")  
while (not rs.EOF)  
response.write("<guest>")  
response.write("<fname>" & rs("fname") & "</fname>")  
response.write("<lname>" & rs("lname") & "</lname>")  
response.write("</guest>")  
rs.MoveNext()  
wend  
  
rs.close()  
conn.close()  
response.write("</guestbook>")  
%>

Transformando XML com XSLT no Servidor

Este ASP transforma um arquivo XML para XHTML no servidor: <%  
'Load XML  
set xml = Server.CreateObject("Microsoft.XMLDOM")  
xml.async = false  
xml.load(Server.MapPath("simple.xml"))  
  
'Load XSL  
set xsl = Server.CreateObject("Microsoft.XMLDOM")  
xsl.async = false  
xsl.load(Server.MapPath("simple.xsl"))  
  
'Transform file  
Response.Write(xml.transformNode(xsl))  
%>

* O primeiro bloco de código cria uma instância do analisador XML da Microsoft (XMLDOM) e carrega o arquivo XML na memória.
* O segundo bloco de código cria outra instância do analisador e carrega o arquivo XSL na memória.
* A última linha de código transforma o documento XML usando o documento XSL e envia o resultado como XHTML para o seu navegador. Agradável!