Introdução

A XML (eXtensible Markup Language, ou Linguagem de Marcação Estendida) é um subconjunto da SGML (Standard Generalized Markup Language, ou Linguagem de Marcação Padrão Generalizada) que permite que uma marcação específica seja criada para especificar idéias e compartilhá-las na rede. Ela tem as virtudes da SGML e da HTML sem qualquer das limitações óbvias.

Pontos fortes da XML

- **Inteligência:** a XML é inteligente para qualquer nível de complexidade. A marcação pode ser alterada de uma marcação mais geral como "<CÃO> Lassie </CÃO>" para uma mais detalhista, como "<CÃO> <VENHA_PARA_CASA> <COLLIE> Lassie </COLLIE> </VENHA_PARA_CASA> </CÃO>". As idéias são bem marcadas para que "<VENDO_DOIS> duplo </VENDO_DOIS>" e "<MAIS_LICOR> duplo </MAIS_LICOR>" sejam sempre valores diferentes. A informação conhece a si mesma. Não é necessária mais nenhuma idéia indesejável;
- **Adaptação:** a XML é a língua-mãe de outras linguagens. Assim, linguagens como DickML e JaneML tornaram-se possíveis. A adaptação é infinita. Marcações personalizadas podem ser criadas para qualquer necessidade. Se uma marcação que descreva como uma pizza pepperoni é diferente de uma pizza calabresa for necessária, ela pode ser feita;
- **Manutenção:** a XML é fácil de manter. Ela contém somente idéias e marcações. Folhas de estilos e links vêm em separado, e não escondidas no documento. Cada um pode ser alterado separadamente quando preciso com fácil acesso e fáceis mudanças. Não é preciso mais se achar em uma bagunça de marcações;
- **Ligação:** a XML possui uma maneira de ligar que inclui todas as formas de ligação. Não só isso; ela liga de maneiras que a HTML não pode. A HTML pode fazer de uma maneira simples, onde um objeto se liga a outro. A XML faz isso, mas também pode ligar dois ou mais pontos a uma idéia. Existem ainda links gêmeos que ligam todas as ideías dentro de uma mesma. Qualquer link entre uma idéia pode ser manipulado de uma única maneira;
- **Simplicidade:** a XML é simples. Um usuário de média experiência que olha a XML pode achá-la difícil de acreditar no que vê. Comparada com a HTML não. Comparada com a SGML é um estudo de simplicidade. A especificação da SGML tem 300 páginas. A da XML, 33. Idéias obscuras e desnecessárias foram retiradas em favor de idéias concisas. A XML vai direto ao ponto;
- **Portabilidade:** a XML é de fácil portabilidade. A razão da sua existência é força e portabilidade. A SGML tem força. A HTML tem portabilidade. A XML tem ambas. A XML pode ser navegada com ou sem o seu DTD (Document Type Definition, ou Definição de Tipo de Documento - as normas que definem como as tags são estruturas nos documentos XML), tornando o download mais rápido. Tudo que um navegador precisa para ver XML é ter a noção que ela própria e a folha de estilos controlam a aparência. Se uma validação estrita é necessária, o seu DTD pode acompanhá-lo e fornecer detalhes exatos da sua marcação.

Objetivos do desenvolvimento da XML

A especificação da XML primou pelos seguintes objetivos:

- Deveria ser claro usar a XML na Internet;

- A XML deveria suportar uma grande variedade de aplicações;
- A XML deveria ser compatíel com SGML;
- Deveria ser fácil escrever programas que processem documentos XML;
- O número de recuros opcionais em XML deveria ser mantido em um mínimo absoluto, idealmente zero;
- Os documentos XML deveriam ser legíveis pelos seres humanos e razoavelmente claros;
- O projeto XML deveria ser preparado rapidamente;
- O projeto XML deveria ser formal e conciso;
- Os documentos XML deveriam ser fáceis de serem criados;
- A concisão na marcação em XML é de mínima importância.

Como a XML é definida

A XML é definida pelas seguintes especificações:

- **Extensible Markup Language (XML) 1.0:** define a sintaxe da XML;
- **XML Pointer Language (XPointer) e XML Linking Language (XLink):** define um padrão para representar os links entre os recursos. Além dos links simples, como a tag <A> da HTML, a XML possui mecanismos para ligar recursos múltiplos e diferentes. A XPointer desceve como endereçar um recurso, e a XLink descreve como associar dois ou mais recursos;
- **Extensible Style Language (XSL):** define a linguagem de folhas de estilos padão para a XML.

Documentos

Se você já está acostumado com a HTML ou a SGML, os documentos XML parecer-lhe-ão familiar. Um documento XML simples é apresentado a seguir:

Exemplo 1: um documento XML simples

<?xml version="1.0"?>

<piada>

<João>Diga <citação>boa noite</citação>, Maria.</João>

<José><citação>Boa noite, Maria.</citação></José>

<aplausos/>

</piada>

Algumas coisas podem sobressair-se para você:

- O documento começa com uma instrução de processamento: `<?xml ...?>`. Esta é a *declaração XML*. Embora não seja obrigatória, a sua presença explícita identifica o documento como um documento XML e indica a versão da XML com a qual ele foi escrito.

- Não há declaração do tipo do documento. Diferentemente da SGML, a XML não requer uma declaração de tipo de documento. Entretanto, uma declaração de tipo de documento pode ser fornecida; além disso, alguns documentos irão precisar de uma para serem entendidos sem ambigüidade.

- Elementos vazios (`<aplausos/>` neste exemplo) tem uma sintaxe modificada. Enquanto que a maioria dos elementos em um documentos envolvem algum conteúdo, elementos vazios são simplesmente marcadores onde alguma coisa ocorre (uma separador horizontal para a marca em `<hr>` em HTML, por exemplo, ou uma referência cruzada para DocBook's para a marca `<xref>`). O final **/>** na sintaxe modificada indica a um programa que processa o documento XML que o elemento é vazio e uma marca de fim correspondente não deve ser procurada. Visto que os documentos XML não requerem uma declaração de tipo de documento, sem esta pista seria impossível para um analisador XML determinar quais marcas são intencionalmente vazias e quais teriam sido deixadas vazias por um erro.
  A XML suavizou a distinção entre elementos declarados como `EMPTY` e elementos que meramente não têm conteúdo. Em XML, é válido usar uma marca de elemento vazio para qualquer um destes casos. Também é válido usar um par de marcas início-fim para elementos vazios: `<aplausos></aplausos>`. Se a interoperabilidade interessa, é melhor reservar a sintaxe de marcas de elementos vazios para elementos declarados como `EMPTY` e usar a marca de elemento vazio somente para estes elementos.

Os documento XML são compostos de marcas e conteúdos. Existem seis tipos de marcações que podem ocorrer em um documento XML: elementos, referências a entidades, comentários, instruções de processamento, seções marcadas e declarações de tipos de documento. As seções seguintes introduzem cada um destes conceitos de marcação.

Elementos

Elementos são a mais comum forma de marcação. Delimitados pelos sinais de menor e maior, a maioria dos elementos identificam a natureza do conteúdo que envolvem. Alguns elementos podem ser vazios, como visto acima; neste caso eles não têm conteúdo. Se um elemento não é vazio, ele inicia com uma marca de início , `<elemento>`, e termina com uma marca de término, `</elemento>`.

Atributos

Atributos são pares de valores nomeados que ocorrem dentro dasmarcas de início após o nome do elemento. Por exremplo:

** **

é um elemento `div` cujo atributo `class` possui o valor `prefácio`. Em XML, todos os valores de atributos devem estar entre aspas.

Referências a Entidades

A fim de introduzir a marcação em um documento, alguns documentos foram reservados para identificar o início da marcação. O sinal de menor, < , por exemplo, identifica o início de uma marca de inicio ou término. Para inserir estes caracteres em seu documento como conteúdo, deve haver uma alternativa para representá-los. Em XML, entidades são usadas para representar estes caracteres especiais. As entidades também são usadas para referenciar um texto freqüentemente repetido ou alterado e incluí-lo no conteúdo de arquivos externos.

Cada entidade deve ter um nome único. A definição dos seus próprios nomes de entidades é discutido na seção declarações de entidades. Para usar uma entidade, você simplesmente a referencia pelo nome. As referências às entidades iniciam com o E comercial e terminam com um ponto-e-vírgula.

Por exemplo, a entidade `&lt;` insere um literal < em um documento. A cadeia de caracteres `<elemento>` pode ser representada em um documento XML como **<**`&lt;elemento>`.

Uma forma especial de referência a entidades, chamada de referência a caracter, pode ser usada para inserir arbitrariamente caracteres Unicode em seu documento. Este é um mecanismo para inserir caracteres que não podem ser diretamente digitados pelo seu teclado.

Referências a caracter podem ter uma das duas formas: referências decimais, `&#8478;`, e referências hexadecimais, `&#x211E;`. Ambas se referem ao caracter Unicode número U+211E.

Comentários

Comentários iniciam com `<!--` e terminam com `-->`. Os comentários podem conter qualquer dado, exceto a literal "`--`". Você pode colocar comentários entre marcas em qualquer lugar em seu documento.

Comentários não fazem parte de um conteúdo textual de um documento XML. Um processador XML não é preciso para reconhecê-los na aplicação.

Instruções de Processamento

Instruções de processamento (PIs) são formas de fornecer informações a uma aplicação. Assim como os comentários, elas não são textualmente parte de um documento XML, mas o processador XML é necessário para reconhecê-las na aplicação.

As instruções de processamento têm a forma: **`<?nome dadospi?>`**. O nome, chamado de alvo PI, identifica a PI na aplicação. As aplicações processariam somente os alvos que eles reconhecem e ignoram todas as outras PIs. Qualquer dado que segue oalvo PI é opcional; é para a aplicação que reconhece o alvo. Os nomes usados em PIs podem ser declarados como notações a fim de identificá-los formalmente.

Os nomes de PI que iniciam com **`xml`** são reservados para a padronização da XML.

Seções CDATA

Em um documento, uma seção `CDATA` instrui o analisador para ignorar a maioria dos caracteres de marcação.

Considere um código-fonte em um documento XML. Ele pode conter caracteres que o analisador XML iria normalmente reconhecer como marcação (`<` e `&`, por exemplo). Para prevenir isto, uma seção CDATA pode ser usada.

<![CDATA[

***p = &q;**

b = (i <= 3);

]]>

Entre o início da seção, `<![CDATA[`, e o fim da seção, `]]>`, todos os dados de caracteres são passados diretamente para a aplicação, sem interpretação. Elementos, referências a entidades, comentários e instruções de processamento são todos irreconhecíveis e os caracteres que os compõem são passados literalmente para a aplicação.

A única cadeia de caracteres que não pode ocorrer em uma seção CDATA é `"]]>"`.

Declarações de Tipos de Documentos

Uma grande porcentagem da especificação da XML trata de vários tipos de declarações que são permitidas em XML. Se você tem experiência com SGML, você reconhecerá estas declarações dos SGML DTDs (Definições de Tipos de Documentos). Se você já viu isto antes, o seu significado pode não ser imediatamente óbvio.

Um dos maiores poderes da XML é que ela permite que você crie seus próprios nomes para marcas. Mas, para uma dada aplicação, é provável não ser significativo para marcas que ocorrer em uma ordem completamente arbitrária. Considere o exemplo 1. Isto teria significado?

<João>Diga <citação>boa noite</citação>, Maria.</João>

<José><citação>Boa noite, Maria.</citação></José>

Sai totalmente do que normalmente esperamos que tenha senso. Simplesmente não *significa* nada.

Entretanto, de um ponto de vista estritamente sintático, não há nada de errado com este documento XML. Assim, se o documento deve ter significado, e certamente há se você estiver escrevendo uma folha de estilos ou aplicação para processá-lo, deve haver alguma restrição na seqüência e aninhamento das marcas. Declarações são onde estas restrições podem ser expressadas.

Mais genericamente, as declarações permitem a um documento comunicar meta-informações ao analisados a respeito do seu

conteúdo. Meta-informação inclui as seqüências e aninhamentos permitidos para as marcas, valores de atributos e seus tipos e padrões, os nomes de arquivos externos que podem ser refernciados e se eles podem ou não conter XML, o formato de algum dado (não-XML) externo que pode ser referenciado e as entidades que podem ser encontradas.

Há quatro tipos de declarações em XML: declarações de tipos de elementos, declarações de listas de atributos, declarações de entidades e declarações de notações.

Declarações de Tipos de Elementos

Declarações de tipos de elementos identificam os nomes dos elementos e a natureza do seu conteúdo. Uma declaração de tipo de elemento típica se parece com isto:

<!ELEMENT piada (João+, José, aplausos?)>

Esta declaração identifica o elemento nomeado como `piada`. Seu *modelo de conteúdo* segue o nome do elemento. O modelo de conteúdo define o que um elemento pode conter. Neste caso, uma `piada` deve conter `João` e `José` e pode conter `aplausos`. As vírgulas entre os nomes dos elementos indicam que eles devem ocorrer em sucessão. O sinal de adição após `João` indica que ele pode ser repetido mais de uma vez, mas deve ocorrer pelo menos uma vez. O ponto de interrogação após `aplausos` indica que ele é opcional (pode estar ausente ou ocorrer somente uma vez). Um nome sem pontuação, como `José`, deve ocorrer somente uma vez.

As declarações para todos os elementos usados em qualquer modelo de conteúdo deve estar presente para que um processador XML verifique a validade do documento.

Além dos nomes de elementos, o símbolo especial `#PCDATA` é reservado para indicar dados de caracter. A cláusula **PCDATA** significa dado de caracter analisável.

Os elementos que contêm somente outros elementos são ditos que têm *conteúdo de elemento*s. Os elementos que contêm outros elementos e `#PCDATA` são ditos que têm *conteúdo misturado*.

Por exemplo, a definição para `José` pode ser

<!ELEMENT José (#PCDATA | citação)*>

A barra vertical indica um relacionamento "ou" e o asterisco indica que o conteúdo é opcional (pode ocorrer zero ou mais vezes); por esta definição, portanto, `José` pode conter zero ou mais caracteres e marcas `citação`, misturadas em qualquer ordem. Todos os modelos de conteúdo misturado devem ter esta forma: `#PCDATA` deve vir primeiro, todos os elementos devem ser separados por barras verticais e o grupo inteiro deve ser opcional.

Outros dois modelos de conteúdo são possíveis: `EMPTY` indica que o elemento não possui conteúdo (e, conseqüentemente, não tem marca de término) e `ANY` indica que *qualquer* conteúdo é permitido. O modelo de conteúdo `ANY` é algumas vezes útil durante a conversão de documentos, mas deveria ser evitado ao máximo em um ambiente de produção, pois desabilita toda a verificação do conteúdo deste elemento.

Aqui está um conjunto completo das declarações de elementos para o Exemplo 1:

Exemplo 2: declarações de elementos para Exemplo 1

<!ELEMENT piada (João+, José, aplausos?)>

<!ELEMENT João (#PCDATA | citação)*>

<!ELEMENT José (#PCDATA | citação)*>

<!ELEMENT citação (#PCDATA)*>

<!ELEMENT aplausos EMPTY>

Declarações de Listas de Atributos

Declarações de listas de atributos identificam que elementos podem ter atributos, que atributos eles podem ter, que valores os atributos podem suportar e qual valor é o padrão. Uma dclaração de lista de atributos típica se parece com isto:

<!ATTLIST piada

nome
ID
#REQUIRED

rótulo
CDATA
#IMPLIED

estado (engraçada | nãoengraçada) 'engraçada'>

Neste exemplo, o elemento `piada` possui três atributos: `nome`, que é um ID e é obrigatório; `rótulo`, que é uma cadeia de caracteres (dados de caracter) e não é obrigatório; e `estado`, que deve ser ou `engraçada` ou `nãoengraçada` e por padrão é `engraçada`, se nenhum valor é especificado.

Cada atributo em uma declaração tem três partes: um nome, um tipo e um valor padrão.

Você tem liberdade para selecionar qualquer nome desejado, sujeito a algumas pequenas restrições, mas os nomes não podem ser repetidos no mesmo elemento.

Existem seis tipos de atributos possíveis:

1. CDATA

Atributos CDATA são cadeias de caracteres; qualquer texto é permitido. Não confunda atributos CDATA com seções CDATA; eles não têm relação.

2. ID

O valor de um atributo ID deve ser um nome. Todos os valores usados para IDs em um documento devem ser diferentes. Os IDs identificam unicamente elementos individuais em um documento. Os elementos podem ter um único atributo ID.

3. IDREF

ou `IDREFS`

O valor de um atributo IDREF deve ser o valor de um único atributo ID em algum elemento no documento. O valor de um atributo IDREFS pode conter valores IDREF múltiplos separados por espaços em branco.

4. ENTITY

ou `ENTITIES`

O valor de um atributo ENTITY deve ser o nome de uma

única entidade (veja sobre declarações de entidades abaixo). O valor de um atributo ENTITIES pode conter valores de entidades múltiplos separados por espaços em branco.

5. NMTOKEN

ou **NMTOKENS**

Atributos de símbolos de nome são uma forma restrita do atributo de cadeia de caracteres. Em geral, um atributo NMTOKEN deve consistir de uma única palavra, mas não há restrições adicionais para a palavra; não tem que estar associado com outro atributo ou declaração. O valor de um atributo NMTOKENS pode conter valores NMTOKEN múltiplos separados por espaços em branco.

6. Uma lista de nomes

Você pode especificar que o valor de um atributo deve ser pego de uma lista específica de nomes. Isto é freqüentemente chamado de tipo enumerado, porque cada um dos valores possíveis está explicitamente enumerado na declaração.

Alternativamente, você pode especificar que os nomes devem atender a um nome de notação (veja sobre declarações de notação abaixo).

Há quatro valores padrão possíveis:

#REQUIRED

O atributo deve ter um valor explicitamente especificado em cada ocorrência do elemento no documento.

#IMPLIED

O valor do atributo não é requerido, e nenhum valor padrão é fornecido. Se um valor não é especificado, o processador XML deve proceder sem um.

"valor"

Qualquer valor válido pode ser dado a um atributo como padrão. O valor do atributo não é requerido em cada elemento no documento, e se ele estiver presente, será dado a ele o valor padrão.

#FIXED

"value"

Uma declaração de atributo pode especificar que um atributo tem um valor fixo. Neste caso, o atributo não é requerido, mas se ele ocorrer deve ter o valor especificado. Se não estiver presente, será dado a ele o valor padrão. Um uso de atributos fixos é para associar

semântica e um elemento. Uma discussão completa vai além do propósito deste trabalho, mas você pode achar vários exemplos de atributos fixos na especificação de XLink.

O processador XML executa a *normalização dos valores dos atributos* nos valores dos atributos: as referências de caracter são substuituídas por caracteres referenciados, referências a entidades são resolvidas (recursivamente) e espaços em branco são normalizados.

Declarações de Entidades

Declarações de entidades lhe permitem associar um nome com algum outro fragmento de conteúdo. Essa construção pode ser um pedaço de texto normal, um pedaço de uma declaração de tipo de documento ou uma referência a um arquivo externo que contém ou texto ou dados binários.

Declarações de entidades típicas são mostradas no Exemplo 3.

Exemplo 3: declaração de entidades típica

```
<!ENTITY
ATI
"ArborText, Inc.">

<!ENTITY boilerplate     SYSTEM
"/standard/legalnotice.xml">

<!ENTITY ATIlogo
SYSTEM "/standard/logo.gif" NDATA GIF87A>
```

Existem três tipos de entidades:

Entidades Internas

Entidades internas associam um nome com uma cadeia de caracteres ou texto literal. A primeira entidade no Exemplo 3 é uma entidade interna. Usando `&ATI;` em qualquer lugar do documento inserirá "ArborText, Inc" naquele local. Entidades internas permitem a você definir atalhos para textos freqüentemente digitados ou textos que se espera que sejam alterados, como o estado de revisão de um documento.

Entidades internas podem incluir referências para outras entidades internas, mas é errado elas serem recursivas.

A especificação XML pré-define cinco entidades internas:

- `<` produz o sinal de menor, <

- `>` produz o sinal de maior, >
- `&` produz o E comercial, &
- `&apos;` produz um apóstrofo, '
- `&quot;` produz aspas, "

Entidades Externas

Entidades externas associam um nome com o conteúdo de um outro arquivo. Entidades externas permitem a documento XML referenciar o conteúdo de um outro arquivo; elas contém ou texto ou dados binários. Se elas contém texto, o conteúdo do arquivo externo é inserido no ponto de referência e analisado como parte do documento referente. Dados binários não são analisados e podem somente ser referenciados em um atributo; eles são usados para referenciar figuras e outro conteúdo não-XML no documento.

A segunda e a terceira entidades no Exemplo 3 são entidades externas.

O uso de `&boilerplate;` inserirá o *conteúdo* do arquivo `/standard/legalnotice.xml` no local da referência da entidade. O processador XML analisará o conteúdo deste arquivo como se ele ocorresse literalmente no local.

A entidade `ATTlogo` também é uma entidade externa, mas o seu conteúdo é binário. A entidade `ATTlogo` pode ser usada somente como o valor de um atributo ENTITY (ou ENTITIES) (em um elemento `graphic`, talvez). O processador XML passará esta informação para a aplicação, mas ele não tenta processar o conteúdo de `/standard/logo.gif`.

Entidades Parâmetro

A entidade parâmetro somente pode ocorrer na declaração de tipo de documento. Uma declaração de uma entidade parâmetro é identificada por "`%` " (porcento e espaço) defronte ao seu nome na declaração. O sinal de porcento também é usado em referências para entidades parâmetro, ao invés do E comercial. As referências a entidade parâmetro são imediatamente expandidas na declaração de tipo de documento e seu texto de substituição é parte da declaração, onde as referências a entidades normais não são expandidas. Entidades parâmetro não são reconhecidas no corpo de um documento.

Voltando às declarações de elementos no Exemplo 2, você perceberá que dois deles têm o mesmo modelo de conteúdo:

<!ELEMENT João (#PCDATA | citação)*>

<!ELEMENT José (#PCDATA | citação)*>

Até o momento, estes dois elementos são a mesma coisa somente porque eles têm a mesma definição literal. A fim de tornar mais explícito o fato de que estes dois elementos são semânticamente a mesma coisa, é usada uma entidade parâmetro para definir seus modelos de conteúdo. Há duas vantagens em se usar uma entidade parâmetro. Primeiramente, ela lhe permite dar um nome descritivo ao conteúdo, e segundo que lhe permite alterar o modelo de conteúdo em somente um local, se você desejar atualizar as declarações do elemento, garantindo que elas sempre fiquem as mesmas:

<!ENTITY % pessoascontentes "#PCDATA | citação">

<!ELEMENT João (%pessoascontentes;)*>

<!ELEMENT José (%pessoascontentes;)*>

Declarações de Notação

Declarações de notação identificam tipos específicos de dados binários externos. Estas informações são passadas para a aplicação de processamento, que pode fazer o uso que quiser ou que desejar. Uma declaração de notação típica é:

<!NOTATION GIF87A SYSTEM "GIF">

Eu preciso de uma Declaração de Tipo de Documento?

Como foi visto, o conteúdo XML pode ser processado sem uma declaração de tipo de documento. Entretanto, existem alguns casos onde a declaração é necessária:

Ambientes de autoria

A maioria dos ambientes de autoria precisa ler e processar declarações de tipo de documento a fim de entender e reforçar o modelo de conteúdo do documento.

Valores padrões de atributos

Se um documento XML conta com valores padrões de atributos, pelo menos uma parte da declaração deve ser processada a fim de se obter os valores padrões corretos.

Manipulação de espaços em branco

A semântica associada com espaço em branco em conteúdo de elementos diferem da semântics associada com espaço em branco em conteúdo misturado. Sem um DTD, não há maneira para o processador distingüir os casos, e todos os elementos são efetivamente conteúdo misturado. Para mais detalhes, veja a seção chamada Manipulação de Espaços em Branco, neste trabalho.

Em aplicações onde uma pessoa compõe ou edita os dados, um DTD provavelmente vai ser preciso se qualquer estrutura deve ser garantida.

Incluindo uma Declaração de Tipo de Documento

Se presente, a declaração de tipo de documento deve ser a primeira coisa em um documento depois de comentários e instruções de processamento opcionais.

A declaração de tipo de documento identifica o elemento raiz do documento e pode conter declarações adicionais. Todos os documentos XML devem ter um elemento raiz único que contenha todo o conteúdo do documento. Declarações adicionais podem vir de um DTD externo, chamado de subconjunto externo, ou ser incluído diretamente no documento, o subconjunto interno, ou ambos:

```
<?XML version="1.0" standalone="no"?>

<!DOCTYPE chapter SYSTEM "dbook.dtd" [

<!ENTITY %ulink.module "IGNORE">

<!ELEMENT ulink (#PCDATA)*>

<!ATTLIST ulink

   xml:link      CDATA  #FIXED "SIMPLE"

   xml-attributes CDATA  #FIXED "HREF URL"

   URL          CDATA  #REQUIRED>

]>

<chapter>...</chapter>
```

Este exemplo referencia um DTD externo, `dbook.dtd`, e inclui declarações de elementos e atributos para o elemento `ulink` no subconjunto interno. Neste caso, `ulink` dá a semântica de um link simples da especificação XLink.

Note que as declarações no subconjunto interno não leva em conta as declarações no subconjunto externo. O processador XML lê o subconjunto interno antes do externo e a *primeira* declaração tem precedência.

A fim de determinar se um documento é válido, o processador XML deve ler a declaração de tipo de documento inteira (ambos os subconjuntos). Mas para algumas aplicações, a validação pode não ser precisa, e pode ser suficiente para o processador ler somente o subconjunto interno. No exemplo acima, se a validade não é importante e a única razão para ler a declaração de tipo de documento é identifcar a semântica de `ulink`, a leitura do subconjunto externo não é necessária.

Você pode comunicar estas informações na *declaração de documento standalone*. A declaração de documento standalone, `standalone="yes"` ou `standalone="no"`, ocorre na declaração XML. Um valor `yes` indica que somente declarações internas precisam ser processadas. Um valor `no` indica que *ambas* as declarações interna e externa devem ser processadas.

Outras questões de marcação

Além da marcação, existem algumas outras questões a considerar: manipulação de espaços em branco, normalização de valores dos atributos e a linguagem com a qual o documento foi escrito.

Manipulação de Espaços em Branco

A manipulação de espaços em brancos é uma questão sutil. Considere o seguinte fragmento de conteúdo:

```
<piada>

<João>Diga <citação>boa noite</citação>, Maria.</João>
```

O espaço em branco (a nova linha entre `<piada>` e `<João>`) é significante?

Provavelmente não.

Mas como você pode afirmar isto? Você somente pode determinar se um espaço em branco é significante se você conhece o modelo de conteúdo dos elementos em questão. Em resumo, um espaço em branco é significante em conteúdo misturado e insignificante em conteúdo de elemento.

A regra para os processadores XML é que eles devem passar por todos os caracteres que não são marcação na aplicação. Se o processador é

um processador de validação, ele também deve informar à aplicação se os caracteres espaços em branco são significantes.

O atributo especial `xml:space` pode ser usado para indicar explicitamente que os espaços em branco são significantes. Em qualquer elemento que inclua a especificação de atributo `xml:space='preserve'`, todos os espaços em branco naquele elemento (e dentro dos subelementos que não alteram explicitamente `xml:space`) serão significantes.

Os únicos valores válidos para `xml:space` são `preserve` e `default`. O valor `default` indica que o processamento padrão é desejado. Em um DTD, o atributo `xml:space` deve ser declarado como um tipo enumerado com somente estes dois valores.

Uma última observação sobre espaços em branco: em texto analisável, os processadores XML são requeridos para normalizar todas as marcas de final de linha para um um único caracter de alimentação de linha (&#A;). Isto raramente é de interesse dos autores, mas elimina um número de questões de portabilidade de plataformas cruzadas.

Normalização dos valores de atributos

O processador XML executa a normalização dos valores de atributos em valores de atributos: referências a caracteres são substituídas por caracteres referenciados, referências a entidades sã resolvidas (recursivamente) e os espaços em branco são normalizados.

Identificação da linguagem

Muitas aplicações de processamento de documentos podem se beneficiar da informação sobre a linguagem natural com a qual o documento foi escrito. A XML define o atributo `xml:lang` para identificar a linguagem. Visto que o propósito deste atributo é padronizar a informação entre as aplicações, a especificação XML também descreve como as linguagens devem ser identificadas.

Validação

Dada a discussão precedente de declarações de tipos, conclui-se que uns documentos são válidos e outros não. Existem duas categorias de documentos XML: bem formatados e válidos.

Documentos Bem Formatados

Um documento somente pode ser bem formatado se ele obedece a sintaxe da XML. Um documento que inclui seqüências de caracteres de marcação que não podem ser analisadas ou são inválidas não podem ser bem formatados.

Além disso, o documento deve atender a todas as seguintes condições (subentendendo-se que algumas destas condições podem exigir experiência com SGML):

- A instância do documento deve estar conforme a gramática dos documentos XML. Em particular, algumas construções de marcações (referências a entidades parâmetro, por exemplo) são somente permitidas em locais específicos. O documento não é bem formatadose tais ocorrerem em outros locais, ainda que o documento esteja bem formatado nos outros casos.

- O texto de substituição para todas as entidades parâmetro referenciadas dentro de uma declaração de marcação consiste em zero ou mais declarações de marcações completas. (Nenhuma entidade usada no documento pode consistir de somente uma parte de uma declaração de marcação.)

- Nenhum atributo pode aparecer mais do que uma vez na mesma marca de início.

- Valores de atributos cadeias de caracteres não podem conter referências a entidades externas.

- Marcas não-vazias devem ser apropriadamente aninhadas.

- Entidades parâmetro devem ser declaradas antes de serem usadas.

- Todas as entidades devem ser declaradas, exceto as seguintes: `amp`, `lt`, `gt`, `apos` e `quot`.

- Uma entidade binária não pode ser referenciada no fluxo do conteúdo; ela pode ser usada somente em um atributo declarado

como `ENTITY` ou `ENTITIES`.

- Nem a texto ou entidades parâmetro são permitidas recursividade, direta ou indiretamente.

Por definição, se um documento não está bem formatado, ele não é XML. Isto significa que não há documento XML que não seja bem formatado e os processadores XML não fazem nada com tais documentos.

Documentos

Um documento bem formatado é válido somente se ele contém uma declaração de tipo de documento e se o documento obedece as restrições da declaração (seqüência e aninhamento de elementos é válido, atributos necessários são fornecidos, valores de atributos são do tipo correto, etc.). A especificação XML identifica todos os creitérios em detalhes.

Ligação

As especificações XPointer e XLink, atualmente em desenvolvimento, introduz um modelo ligação padrão para a XML. Em consideração ao espaço e o fato de que rascunho da XLink ainda está sendo desenvolvido, o que segue é um exame dos recursos da XLink, em vez de uma descrição detalhada da especificação.

Em XLink, um link expressa um relacionamento entre recursos. Um recurso é qualquer local (um elemento, o seu conteúdo, ou uma parte do seu conteúdo, por exemplo) que é endereçável em um link. A natureza exata do relacionamento entre os recursos depende da aplicação que processa o link e da informação semântica fornecida.

Alguns destaques da XLink são:

- a XLink lhe dá controle sobre a semântica do link.
- a XLink introduz Links Extendidos. Links Extendidos podem envolver mais de dois recursos.
- a XPointer introduz Ponteiros Extendidos (XPointers). Os XPointers fornecem um método sofisticado de localizar recursos. Em particular, os XPointers lhe permitem localizar recursos arbitrários em um documento, sem que seja necessário que o recurso seja identificado com um atributo ID.

Visto que a XML não tem um conjunto fixo de elementos, o nome do elemento de ligação não pode ser usado para localizar links. Em vez disso, os processadores XML identificam os links pelo reconhecimento do atributo **`xml:link`**. Outros atributos

podem ser usados para fornecer informações adicionais ao processador XML. Um recurso de renomeação de atributos existe para contronar colisões de nome em aplicações existentes.

Dois atributos, **show** e **actuate** lhe permitem exercer algum controle sobre o comportamento da ligação. O atributo **show** determina se o documento para o qual está se fazendo a ligação está embutido no documento atual, substitui o documento atual ou é mostrado em uma nova janela quando o link é acionado. **actuate** determina como o link é acionado, ou automaticamente ou quando selecionado pelo usuário.

Algumas aplicações irão necessitar de um controle muito mais acurado sobre os comportamento dos links. Para estas aplicações, locais padrão são fornecidos onde a semântica adicional pode ser expressa.

Links Simples

Um link simples lembra fortemente um link HTML <A>:

```
<link xml:link="simple" href="locator">Texto do Link</link>
```

Um link simples identifica um link entre dois recursos, um dos quais é o próprio conteúdo do elemento do link. Este é um link in-line.

Um *localizador* identifica o outro recurso. Um localizador pode ser um URL, uma consulta ou um Ponteiro Extendido.

Links Extendidos

Links extendidos lhe permitem expressar relacionamentos entre mais de dois recursos:

```
<elink xml:link="extended" role="annotation">

<locator xml:link="locator" href="text.loc">Texto</locator>

<locator xml:link="locator" href="annot1.loc">Anotações</locator>

<locator xml:link="locator" href="annot2.loc">Mais Anotações</locator>

<locator xml:link="locator" href="litcrit.loc">Literatura Crítica</locator>

</elink>
```

Este exemplo mostra como os relacionamentos entre um trabalho literário, anotações e literatura crítica deste documento podem ser expressos. Note que este link é separado de todos os recursos envolvidos.

Links extendidos podem ser in-line, para que o conteúdo do elemento de ligação (outro que não dseja elemento localizadores) participe no link como um recurso, mas esse não é necessariamente o caso. O exemplo acima é um link out-of-line link, pois não usa seu conteúdo como um recurso.

Ponteiros Extendidos

Referências cruzadas com o mecanismo XML ID/IDREF (que é similar ao mecanismo `#fragment` em HTML) requer que o documento para o qual se está ligando tenha âncoras definidas, onde os links são desejados (e tecnicamente requer que ID e IDREF ocorram no mesmo documento). Este pode não ser sempre o caso, e algumas vezes não é possível modificar o documento que você deseja ligar.

Os XML XPointers tomam emprestados conceitos de HyTime e da Text Encoding Initiative (TEI). Os XPointers oferecem a sintaxe que lhe permite localizar um recurso através da árvore de elementos do documento que contém o recurso.

Por exemplo:

```
criança(2,piada).(3,.)
```

locazida a terceira **criança** (qulaquer que possa ser) da segunda **piada** no documento.

Os XPointers podem expandir as regiões da árvore. A expansão XPointer

```
span(criança(2,piada),criança(3,piada))
```

seleciona a segunda e terceira **oldjoke** no documento.

Além da seleção através de elementos, os XPointers permitem a seleção por ID, valores de atributos e correspodência de cadeias de caracteres. Neste trabalho, o XPointer

```
span(raiz()criança(3,set1)string(1,"Aqui",0),
     raiz()criança(3,set1)string(1,"Aqui",4))
```

seleciona a primeira ocorrência da palavra "Aqui". O link pode ser estabelecido por um link extendido *sem modificar* o documento alvo.

Note que o alcance de um XPointer pode expandir uma seção estruturamente inválida do documento. A especificação XLink não especifica como as aplicações devem tratar tais alcances.

Grupos de Links Extendidos

Links out-of-line introduzem a possibilidade de que um processador XML possa precisar processar vários arquivos para mostrar corretamento o documento de hipertexto.

De acordo com o exemplo de texto de antoação acima, e assumindo que o texto é, na verdade, somente para leitura, o processador XML deve carregar pelo menos o texto e o documento que contém o link extendido.

A XLink define os Grupos de Links Extendidos para este propósito. Grupos de Links Extendidos podem ser usados recursicamente, e um atributo **steps** é fornecido para limitar o nível da recursividade.

Entendendo as Partes

Alguns documentos, particularmente documentos compostos que colaboram com XLinks, são provavelmente compostos de elementos de múltiplos conuntos de marcas. Por exemplo, um artigo técnico pode ser escrito usando-se um DTD, mas

inclui equações matemáticas escritas em MathML e gráficos vetoriais escritos em um terceiro DTD.

A fim de uma aplicação de processamento associar a semântica correta com um elemento, ela deve conhecer que conjunto de marcas o elemento possui. A XML resolve este problema com nomes espaçados. Nomes espaçados em XML descreve este sistema em detalhes.

O princípio é permitir a um prefixo delimitado por dois pontos associar-se com alguma semântica externa através de uma URI. O uso deste prefixo identifica o elemento como tendo a semântica descrita pela URI. Por exemplo:

<bk:para>A fração 3/4 pode ser expressa em MathML como:

<ml:cn type="rational">3<ml:sep/>4</ml:cn>.</bk:para>

O elemento **para** neste exemplo é identificado explicitamente como sendo o nome espaçado identificado pelo prefixo **bk**, que deve ter sido definido anteriormente no documento, e os elementos **cn** e **sep** são do elemento **ml** (presumivelmente associado de alguma maneira com MathML).

Estilo e Substância

Os navegadores HTML são de difícil codificação. Embora alguns navegadores possam basear sua formatação em Cascading Style Sheets (CSS), eles ainda contêm convenções de difícil codificação para documentos que não fornecem uma folha de estilos. Um cabeçalho de primeiro nível aparece da maneira correta, pois o navegador reconhece a marca <h1>.

Novamente, visto que os documentos XML não têm um conjunto fixo de marcas, testa aproximação não funcionará. A apresentação de um documento XML é *dependente* de uma folha de estilos.

A linguagem de folha de estilos padrão para os documentos XML é a Extensible Style Language (XSL).

Outras linguagens de folhas de estilos, como Cascading Style Sheets, também são suportadas.