Licenciatura em Engenharia Informática – DEI/ISEP
Linguagens e Programação 2017/2018
Aula Teórico - Prática

Ficha TP 4

XML, XML Schema

Objetivos:

- Conhecer o modelo de dados XML e a sua representação textual;
- Mediante conhecimento sobre a definição de um vocabulário XML e regras de boa marcação definir um vocabulário para um fim específico e o XML *Schema*;
- Validar documentos com XML *Schemas*;
- Identificar documentos XML bem formados e válidos e as suas componentes estruturais.

4. TECNOLOGIAS XML

Desde a sua especificação em 1998, a eXtended Markup Language (XML) rapidamente adquiriu grande popularidade entre os especialistas informáticos, dado que incorpora conceitos para descrever, armazenar, integrar e manipular dados estruturados. Uma determinante do sucesso da XML está fundamentada na sua própria "natureza": o formato XML tornou-se, em pouco tempo um formato universal para a interoperabilidade e a partilha de dados entre aplicações. O conceito e a filosofia da XML são simples, os seus derivados e as possíveis aplicações são inúmeras.

Num documento de formato XML os dados são descritos em texto (Unicode). É genericamente aceite que um formato de texto não é a solução mais económica para armazenar dados, mas muitas vezes é a mais eficaz. Mais eficaz, porque é mais fácil trabalhar com um texto formatado com uma determinada norma, do que trabalhar com os formatos binários típicos de muitas bases de dados.

A (XML) surge como uma ferramenta extremamente poderosa no que respeita à gestão, organização e visualização de bases de dados textuais. O XML é uma norma que descreve ou codifica a estrutura e conteúdo de informação legível por computadores e seres humanos. Descreve o conteúdo e a estrutura lógica de como um conteúdo se deve agregar, mas não a forma como é representada e/ou deve ser apresentada num navegador. A flexibilidade do XML provém da possibilidade de transportar qualquer tipo de dados, inclusive binários, mantendo-os estruturalmente coesos e inteligíveis, através da estrutura de marcação (`<etiqueta>value</etiqueta>`).

Devido a esta estrutura, é também possível combinar num mesmo documento, vários objetos com tipos de dados diferentes. A XML facilita declarações precisas, não só dos conteúdos de um documento, como também dos elementos convenientes à estruturação desses conteúdos. Para demonstrar esta característica básica do XML, sugere-se a análise do exemplo seguinte.

```
…
<pessoa>
    <nome>Pedro Tripeiro</nome>
    <morada>Rua dos Prazeres, 233</morada>
    <codigo.postal>4000 - 319</codigo.postal>
    <localidade>Porto</localidade>
    <telefone>22 33 44 55</telefone>
</pessoa>
```

Listagem 4.1: Extracto de um documento XML

Este pequeno exemplo demonstra de que forma a XML serve para a estruturação de dados e para descrevê-los sem dúvidas ou ambiguidades em formato texto. Isto porque a XML permite ao autor do documento definir os seus próprios elementos, criar e usar as <etiquetas> (Tags[3]) mais adequadas à descrição e à estruturação dos dados em causa: nome, morada, código postal, etc.

Recapitulemos os principais aspetos:

- XML é a abreviação para eXtensible Markup Language;
- Com a XML "definem-se" linguagens de marcação (Markup Languages);
- A XML foi desenvolvida para descrever dados com forte estruturação. Serve para estruturar todos os tipos de dados;
- As etiquetas da XML não foram pré-definidos; qualquer autor tem a liberdade (e a necessidade!) de definir as suas próprias etiquetas.

4.1 Estrutura de documentos XML

As regras de sintaxe da XML são bastante simples e bastante rígidas. As regras sintáticas são fáceis de aprender e fáceis de aplicar. Esta situação simplifica o processo de desenvolvimento de aplicações para o tratamento de dados em formato XML considerando o uso de <etiquetas> descritivas (Descriptive Markup) e de Tags semânticos. É possível usar <etiquetas> com nomes apropriados à descrição do conteúdo dos diversos elementos. Sem dúvida que as etiquetas semânticas fazem parte do que poderíamos chamar "a lógica XML". Porém, a XML não estabelece que as etiquetas tenham que ser obrigatoriamente descritivas ou semânticas, a única coisa que a XML realmente define são algumas regras para escrever "documentos XML bem formados".

A anotação de um documento descreve a sua estrutura e induz uma interpretação do seu conteúdo. A anotação é composta por: etiquetas de início de elementos, etiquetas de fim de elementos, etiquetas de elementos vazios, referências a entidades, comentários, limitadores de secções especiais de texto, declarações de tipo de documento e instruções de processamento.

Como se pode verificar na Listagem 4.2, logo a seguir à declaração XML surge o elemento raiz (*root element*). Todos os documentos XML incluem um só elemento raiz. O elemento raiz (<memorando>) envolve todos os demais elementos.

[3] Ao longo do texto será usado o termo etiqueta para referir uma tag no XML. Refira-se no entanto que os termos marca, marcador e anotação são comummente aceites neste contexto.

No seu interior são descritos os 6 sub-elementos (<titulo>, <data>, <para>, <de>, <cabecalho> e <mensagem>). Na última linha é fechado o elemento raiz (</memorando>) terminando o documento XML.

```
<?xml version=" 1.0" encoding="UTF-8"?>
<Memorando>
  <titulo> Memorando urgente</titulo>
  <data>23.08.01 </data>
  <para> Maria</para>
  <de>Paulo</de>
  <cabecalho> Não esqueças !</cabecalho>
  <mensagem> Submeter Ficha de Avaliação formativa  </mensagem>
</Memorando>
```

Declaração XML

Listagem 4.2: Documento XML com declaração ?xml

A XML é uma meta-linguagem de estruturação em árvore, isto é: hierarquiza os <elementos> a vários níveis, sendo o elemento raiz (a raiz da árvore), e os <elementos> encaixados os ramos e as folhas (os "nós") da árvore – segundo o Document Object Model (DOM). Esta estrutura implica as seguintes consequências:

- Todos os <elementos> tem que ser corretamente encaixados ou encadeados.
- O emparelhamento ("nesting") defeituoso de elementos conduz invariavelmente a erros de processamento do documento XML.
- O primeiro <elemento> de um documento XML é sempre o elemento de raiz. Em cada documento XML existe sempre um elemento de raiz.
- Todos os documentos de formato XML requerem um par de etiquetas para a definição do elemento raiz.

  ```
  <elementoraiz> </elementoraiz>
  ```

- Todos os demais <elementos> de um documento XML estarão sempre delimitados pelo <etiqueta-de-Inicio> e pela <etiqueta-de-Fim> do elemento de raiz.
- Os documentos XML são extensíveis; é possível incorporar mais <elementos>. Nenhuma regra define que um documento deve conter um número limitado de <elementos>.
- Qualquer <elemento> pode incluir <sub-elementos>. Os <sub-elementos> devem ser sempre corretamente encaixados nos seus elementos pais.

  ```
  <root>
     <child>
       <subchild> ..... </subchild>
     </child>
  </root>
  ```

- Qualquer documento de tipo XML pode ser expandido, para incluir mais informações. Essa extensibilidade é importante quando queremos incorporar mais detalhes, mais pormenores, aspetos complementares, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoas>
  <pessoa id="123">
        <nome>  António José Silva</nome>
        <data-nascimento>
                <ano>1965</ano>
                <mes>10</mes>
                <dia>3</dia>
        </data-nascimento>
        <bi>4025527</bi>
  </pessoa>
  <pessoa id="234">
        <nome>  Carlos Tavares</nome>
        <data-nascimento>
                <ano>1975</ano>
                <mes>10</mes>
                <dia>3</dia>
        </data-nascimento>
        <bi>8085527</bi>
  </pessoa>
</pessoas>
```

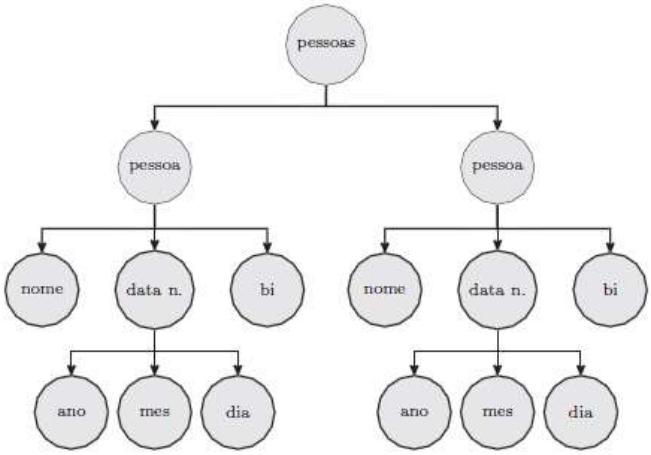

Figura 4.1: Representação Textual/Representação Hierárquica

4.2 *A declaração XML*

De acordo com a norma W3C[4], um documento XML deve começar sempre com uma declaração XML (designada por vezes por **prólogo**). Normalmente, se o documento tiver algo antes da declaração ou esta estiver ausente, qualquer processador identificará um erro ao tentar processar o documento. Todos os documentos XML devem conter esta declaração para especificarem que são documentos XML.

As componentes da declaração XML vêm sempre dentro de um **<?marcador especial?>** entre pontos de interrogação.

Uma declaração XML é uma anotação especial com a seguinte sintaxe:

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

Podem ser usados três atributos numa declaração XML:

- **version** - o valor deste atributo indica a versão de XML que está a ser utilizada. Este atributo é obrigatório e deve estar presente em todas as declarações.
- **standalone** - este atributo é opcional e pode ter um de dois valores: o valor "yes" indica que o documento está auto-contido, não tem referências a entidades externas; o valor "no"indica que o documento contém referências a entidades externas.
- **encoding** - Este atributo é também opcional e indica qual a codificação usada para os caracteres: o valor por omissão é UTF-8.

4.3 *Comentários*

Um comentário pode aparecer em qualquer ponto de um documento XML. Começa pela marca **<!-- e termina com a marca -->**.

[4] World Wide Web Consortium http://www.w3.org/

```
<?xml version="1.0" encoding=" UTF-8"?>
   <!--Isto é um comentário no início-->
<doc>Olá Mundo!!!</doc>
```

Listagem 4.3: Exemplo de um comentário em XML

Existem algumas restrições à utilização de comentários:

- Não podem aparecer antes da declaração XML;
- Não podem aparecer dentro de uma anotação;
- Não se pode utilizar a sequência de caracteres "--"dentro de um comentário.

Além da função de documentação, os comentários podem ainda ser utilizados para remover temporariamente partes do documento, desde que essas partes não contenham comentários.

```
<?xml version="1.0" encoding=" UTF-8"?>
<RECEITAS>
  <TITULO> O Meu Livro de Receitas </TITULO>
  <RECEITA ORIGEM="Portugal">
     <TITULO> Bolo </TITULO>
     <!--
     <INGREDIENTE> 500g de farinha </INGREDIENTE>
     -->
     <INGREDIENTE> 200g de açucar </INGREDIENTE>
     <INGREDIENTE> 300g de manteiga </INGREDIENTE>
  </RECEITA>
</RECEITAS>
```

Listagem 4.4: Comentário usado para remover partes de um documento

4.4 *Instruções de processamento*

Uma instrução de processamento não faz parte do conteúdo do documento. É uma indicação directa ao processador do documento de que algo deve ser executado naquele ponto quando o documento estiver a ser transformado.

Uma instrução de processamento começa por **<?**`id-processador` e termina por **?>**. O `id-processador` deverá indicar a que tipo de processamento a instrução se destina. Note-se que a declaração XML não é mais do que uma instrução de processamento. A seguir é apresentado um documento XML que contém algumas instruções de processamento.

```
<?xml version="1.0" encoding=" UTF-8"?>
<agenda>
  <?html action="hr"?>
  <entrada id="e1" tipo="pessoa">
  <nome>Maria de Belém</nome>
  <email>Mbelem@vaievem.pt</email>
  <telefone>22 8340600</telefone>
</agenda>
<?html action="hr"?>
```

Listagem 4.5: Instruções de processamento

Neste exemplo, além da já referida declaração XML, estão presentes duas instruções de processamento análogas **<?html ...?>.** Estas duas instruções de processamento identificam ordens que deverão ser executadas aquando da transformação deste documento para HTML. Neste caso concreto, indica-se que, quando o processamento do documento XML coincidir com a geração da sua versão HTML, deverão ser geradas duas marcas **hr** nas posições onde se encontram as instruções de processamento.

Antes do corpo do documento poderão surgir: a declaração XML, instruções de processamento e comentários. Um documento XML bem formado tem várias componentes descritas nas secções seguintes. As instruções de processamento e os comentários podem aparecer em qualquer ponto do corpo do documento.

4.5 *Elementos*

A estrutura de um documento XML é definida à custa de elementos que determinam os blocos lógicos em que o texto global pode ser decomposto.

Um **elemento (**não vazio**)** é composto pela sua **etiqueta de início**, pelo seu **conteúdo** e pela sua **etiqueta de fim**.

Uma etiqueta de início começa por '<' e termina com '>', e uma etiqueta de fim começa por '</' e termina com '>'. Quer uma quer outra contêm no meio o nome do elemento que inicia ou que termina, respectivamente.

Assim sendo, um elemento **(**não vazio**)** é constituído por:

- `<elemento>` a **etiqueta de início**
- `</elemento>` a **etiqueta de fim**
- valor/conteúdo: os dados posicionados entre as duas referidas etiquetas

Numa definição rigorosa, o conteúdo delimitado pelas etiquetas é designado por valor/conteúdo do elemento. No caso do autor usar marcadores semânticos, a **etiqueta de início** e a **etiqueta de fim** de um `<elemento>` descrevem o conteúdo dos dados por eles delimitados.

Vejamos o exemplo de um documento XML que tem um `<elemento>` designado pelo nome "director", com o valor "Joana Nunes". No documento que descreve os membros do teatro, a notação será:

```
<teatro>
…
  <diretor> Joana Nunes</diretor>
…
</teatro>
```

O `<elemento>` com a designação "`diretor`" possibilita o uso de um marcador semântico com o valor "Joana Nunes".

Deste modo é possível distinguir esta informação de outra de carácter semelhante, mas não idêntico. Considerando-se o caso de existir um outro elemento com o mesmo valor "Joana Nunes". Este elemento poderia usar, por exemplo, a etiqueta "atriz":

```
<atriz>Joana Nunes</atriz>
```

Dado que cada `<elemento>` pode ter as suas etiquetas, é possível distinguir os dois `<elementos>` de forma inequívoca. Podendo-se concluir que existe uma Joana Nunes no cargo de diretor do teatro, e uma Joana Nunes que é uma atriz de teatro.

No texto abaixo, a palavra Braga está anotada como sendo um elemento de nome lugar.

```
Vais ver o espetáculo a <lugar>Braga</lugar>?
```

Uma restrição inerente ao XML é a de que o nome de uma etiqueta de fim tem de ser igual ao da etiqueta de início imediatamente anterior (este será um dos princípios de boa formação de um documento).

Um elemento tem de estar completamente contido/encaixado noutro elemento, exceto o ancestral de todos os elementos (o elemento raiz). Algumas estruturas hierárquicas podem ser recursivas. Esta possibilidade de recursividade poderá causar problemas no momento de processar a informação, mas é necessária para modelar alguns tipos de informação.

A definição do nome de um elemento deve obedecer às seguintes regras:

- o primeiro carácter deverá ser uma letra, um *underscore* (_) ou um sinal de dois pontos;
- os caracteres seguintes podem ser letras, dígitos, *underscores*, hífens, pontos e dois pontos;
- o espaço em branco não pode aparecer no nome de uma anotação.

Basicamente são permitidos todos os nomes para um `<elemento>` que se deseje definir. Referem-se, no entanto algumas restrições:

- O nome de um `<elemento>` pode começar com uma letra, e poderá conter letras, dígitos, traços (-), dois pontos (:) ou pontos (.). Podem incluir dígitos e caracteres de ponto (:;,.), mas o nome de um `<elemento>` não pode começar por nenhum deles.
- As letras e o traço podem ser usados em qualquer parte do nome.
- Os nomes que comecem com a string "*xml*" ficam reservados para o uso interno de XML e portanto estão vedados para uso dos autores.
- Não são permitidos nomes de `<elementos>` que comecem com "*xml*", "*xMl*", "*XML*" ou qualquer outra variante.
- Os restantes caracteres, como alguns símbolos e espaços em branco, não podem ser utilizados num nome de <elemento>.
- Em XML há distinção entre maiúsculas e minúsculas (XML é *case sensitive).* Assim, lugar, Lugar e LUGAR são referências a elementos distintos.

De seguida apresentam-se alguns exemplos de **nomes válidos e nomes inválidos**:

Tabela 4.1: Nomes válidos/ Nomes inválidos

Nomes válidos	Nomes inválidos

`<Doc:princ> </Doc:princ>`
`<documento> </documento>`
`<_secreto> </_secreto>`
`<aluno4> </aluno4>`
`<DB_tab5> </DB_tab5>`
	`<1documento> </1documento>` -- começa por dígito
	`<aluno(4)> </aluno(4)>` -- tem parentesis
	`<DB tab5> </DB tab5>` -- tem espaço em branco

Por outro lado, no conteúdo de um elemento nunca deverão aparecer os caracteres '<' e '>' pois são os caracteres que limitam as etiquetas. Em sua substituição devem-se usar, respetivamente, as entidades do tipo carácter '<' e '>'. Os caracteres especiais estabelecem referência a entidades pré-definidas.

Tabela 4.2: Caracteres reservados

Entidades	Caracteres
<	<
>	>
&	&
"	"
'	'

Qualquer processador ou editor de XML fará a substituição automática das entidades pelos caracteres correspondentes. Se, num documento XML, estivesse escrito desta forma:

```
A anotação &lt;nome&gt; é usada para anotar nomes.
```

Um editor mostraria o mesmo texto da seguinte maneira:

```
A anotação <nome> é usada para anotar nomes.
```

4.5.1 Tipos de conteúdo

Os elementos podem incluir outros elementos (elementos filho) e texto. Há elementos que não contêm texto diretamente, apenas contêm outros elementos. São **elementos estruturantes**. Num dos exemplos anteriores, apresentou-se o livro de receitas: o elemento RECEITAS, tem como conteúdo apenas elementos RECEITA, é um elemento deste tipo.

```
<RECEITAS>
    <RECEITA> ... </RECEITA>
    <RECEITA> ... </RECEITA>
    <RECEITA> ... </RECEITA>
</RECEITAS>
```

Listagem 4.6: Elementos estruturantes

Os caracteres brancos (espaços, mudanças de linha, tabulações) que surgem no conteúdo de elementos que não contêm texto são irrelevantes e não são considerados como fazendo parte do documento. Assim, o excerto anterior poderia ser igualmente representado por:

```
<RECEITAS><RECEITA> ... </RECEITA><RECEITA> ...
</RECEITA><RECEITA> ...</RECEITA>...</RECEITAS>
```

No entanto, em algum ponto da hierarquia do documento, o texto irá aparecer. Neste ponto o elemento contém texto ou uma mistura de texto com elementos filho. Um elemento que contenha apenas texto é designado como tendo **conteúdo textual**.

```
<lugar>Braga</lugar>
   <INGREDIENTE>Meia dúzia de ovos</INGREDIENTE>
<data>(1922)</data>
```

Listagem 4.7: Elementos com conteúdo textual

Quando um elemento contém simultaneamente texto e elementos filho, designa-se por **elemento de conteúdo misto**. Nos exemplos abaixo, `<verso>` e `<p>` são instâncias de elementos de conteúdo misto enquanto `<nome>` e `<lugar>` são instâncias de elementos de conteúdo textual.

```
<verso>Olha, <nome>Daisy</nome>: quando ...</verso>
<p>Vais ver o espetáculo a <lugar>Braga</lugar>?</p>
```

Listagem 4. 1: Elementos de conteúdo misto

Por último, um elemento pode não ter qualquer conteúdo. Este tipo de elemento é normalmente designado por **elemento vazio**. Estes elementos são normalmente utilizados pelo seu significado posicional. Alguns exemplos conhecidos do HTML são os elementos `<br>` e `<hr>` que marcam, respetivamente, a quebra de uma linha e o traçar de uma linha horizontal.

O XML também permite o uso de elementos vazios, com a seguinte notação:

```
<elemento-sem-valor/>
```

Ou

```
<elemento-sem-valor></elemento-sem-valor>
```

Um elemento com conteúdo é por exemplo:

```
<nome> Linguagens e Programação</nome>
```

Um elemento vazio não tem conteúdo para não ter uma etiqueta de fim, o elemento vazio usa a notação <`identificador` />, sendo facultativo o emprego de atributos:

```
<identificador DNI="235138767"/>
```

4.6 *Atributos*

Um **elemento** pode ter um ou mais **atributos** que, por sua vez, podem ser opcionais ou obrigatórios. Os atributos visam qualificar o elemento a que estão associados. Ao pensar no problema da distinção entre elemento e atributo, é possível traçar uma analogia com a língua portuguesa: os elementos são os substantivos, e os atributos os adjetivos. Podemos então estabelecer a seguinte equivalência:

```
Isto é uma casa - <CASA> … </CASA>
```

```
Isto é uma casa verde - <CASA COR="verde"> …</CASA>
```

Não há limite para o número de atributos que podem estar associados a um elemento. Os atributos aparecem sempre na etiqueta que marca o início de um elemento, uma vez que vão qualificar o conteúdo que se segue. Um atributo é definido por um par constituído por um **nome** e um **valor**: o nome e o valor devem estar separados pelo sinal = e o valor deverá estar colocado dentro de aspas simples ou duplas.

O nome de um atributo segue as mesmas regras do nome dos elementos (secção 4.5). O valor será sempre o texto que estiver dentro das aspas.

O problema da representação de uma dada informação num elemento ou num atributo torna-se por vezes complexa. Não existe uma fronteira bem definida entre os dois conceitos e muitas vezes a decisão não é um processo simples.

```
<agenda>
  <entrada id="e1" tipo="pessoa">
    <nome>Ana de Belém</nome>
    <email>abelem@isep.ipp.pt</email>
    <telefone>22 8340500</telefone>
    <data-nascimento>
          <ano>1975</ano>
          <mes>10</mes>
          <dia>3</dia>
    </data-nascimento>
  </entrada>
…
</agenda>
```

Listagem 4.8: Informação nos elementos

Uma agenda é composta por um ou mais elementos entrada. Cada um destes elementos é estruturado em vários sub-elementos, nome, email, telefone e data de nascimento, e tem dois atributos – id e tipo – que lhe atribuem certas propriedades. Uma solução alternativa seria colocar a informação dos elementos nos atributos.

```
<agenda>
    <entrada id="e1" tipo="pessoa" nome="Ana de Belém"
email="abelem@isep.ipp.pt" telefone="22 8340500"/>
...
</agenda>
```

Listagem 4.9: Informação nos atributos

Este documento é também um documento XML bem formado – o que demonstra a fragilidade da fronteira entre os conceitos de atributo e elemento. Assim sendo é possível referir que:

- Os atributos contribuem para que um <elemento> venha enriquecido com informações secundárias - por vezes meta-informações dirigidas a aplicações que processam documentos XML.
- Quando existem vários atributos, estes devem ser separados por um espaço. O uso de vírgulas para separação de atributos não é permitido. Não é relevante a sequência de especificação dos atributos.
- Qualquer atributo deve ser especificado uma só vez no marcador de abertura. Os atributos não podem ser repetidos dentro de um elemento.

4.6.1 Limitações no uso de atributos

Considerando que os <elementos> podem ser enriquecidos através da inserção de pares nome-valor chamados atributos. A sintaxe dos atributos é simples e facilmente compreensível. Mas nem

sempre são de fácil processamento. Considerando o facto de que qualquer elemento XML é extensível, é possível propor uma alternativa às soluções com atributos.

Elementos com atributos	Elementos encaixados

`<nota data="12/11/99" região="Norte">`
`  Faltam professores no Minho`
`</nota>`
	`<nota>`
	`  <região>Norte</região>`
	`  <data>12/11/99</data>`
	`  Faltam professores no Minho`
	`</nota>`

As duas versões transportam o mesmo conteúdo e o mesmo teor de informação. Ambas apresentam sintaxe correta. Qual será então a versão "mais genuinamente XML"?

Considera-se que a segunda variante - a que usa um `<sub-elemento>` encaixado para substituir o atributo, será "a mais típica XML", porque permite, por exemplo, uma futura extensão, o que não é possível realizar com atributos.

Referem-se as principais limitações na utilização de atributos:

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

4.6.2 Vantagens

Existem situações onde a utilização dos atributos oferece vantagens sobre o encadeamento de `<elementos>`. Os atributos agregam informações complementares aos <elementos>, com valores "por omissão", se necessário.

Outro especto refere-se ao facto dos atributos poderem transportar "meta-informação", que não faz parte dos dados, mas que pode ser relevante para as aplicações que processam os dados. Por exemplo, o tipo de ficheiro não tem relevância de maior para os dados em si, mas "avisa" a aplicação que deverá manipular um ficheiro de imagens de tipo GIF.

```
<file type="gif">LPROG.gif</file>
```

4.6.3 Atributos reservados

Há características universais que o conteúdo dos elementos pode partilhar em diferentes aplicações. Incluem-se nestas características a língua utilizada e a importância dos caracteres brancos. Para evitar conflitos com nomes de atributos definidos pelo utilizador, a norma XML[1] reservou o prefixo "**xml:**". Identificam-se dois atributos reservados, "*xml:lang*" e "*xml:space*", que se descrevem de seguida:

- **xml:lang** – Este atributo pode ser associado a qualquer elemento e indica a língua em que o texto desse elemento está escrito. Esta característica é útil em ambientes multilingues. Exemplos:

```
<para xml:lang="en">Hello</para>
<para xml:lang="pt">Olá</para>
<para xml:lang="fr">Bonjour</para>
```

  Desta forma seria possível seleccionar o elemento a mostrar em dada altura ao utilizador, dependendo da escolha linguística deste.

- **xml:space** – Este atributo, também associável a qualquer elemento, pode assumir um de dois valores: **default** ou **preserve**. Serve para indicar se o espaço branco no conteúdo do elemento em causa é ou não relevante. O valor **default** indica que o espaço não é importante e fará com que a maioria dos processadores compacte qualquer sequência de caracteres brancos num espaço. O valor **preserve** fará com que os caracteres brancos sejam mantidos como parte integrante do conteúdo do elemento.

4.7 *Regras de boa formação*

Após ter-se analisado os componentes de um documento XML é possível enunciar um conjunto de regras a ser seguido na construção de um documento XML bem formado:

- Um documento XML deve ter sempre uma declaração XML no início. A declaração XML deverá assumir a seguinte forma:

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

- O documento deve incluir um ou mais elementos. O primeiro, que delimitará todo o corpo do documento, é o elemento raiz e todos os outros deverão estar incluídos dentro dele. No documento XML apresentado de seguida, o elemento raiz é <sumários>, contendo no seu interior, os elementos <disciplina>, <professor> e <aula>, os quais, por sua vez, são constituídos por outros elementos e texto:

```
<?xml version="1.0" encoding="UTF-8"?>
<sumarios>
    <disciplina> Linguagens e Programação</disciplina>
    <professor>
        <nome>Ana Pereira</nome>
        <url>http://www.moodle.isep.ipp.pt/</url>
    </professor>
    <aula tipo="T">
       <data>2009.04.23</data>
       <sumario>
          <p> Tecnologias XML: perspectiva histórica.</p>
          <p>Validação de Documentos.</p>
           <p>XML Schema</p>
       </sumario>
    </aula>
    ...
</sumarios>
```

- Todos os elementos têm etiquetas de início e fecho. A única exceção são os elementos vazios, cujas etiquetas podem ser substituídas por uma única etiqueta de início que termina em '/ >'.
- Os elementos deverão estar encaixados corretamente. No exemplo seguinte, apresenta-se uma situação com elementos aninhados incorretamente.

```
O título do livro é <B><I>XML</B> ... Prática</I>
```

Ao contrário do que acontece no exemplo, o primeiro elemento a começar terá sempre de ser o último a fechar e o último a abrir deverá ser sempre o primeiro a fechar. Neste caso, as anotações de B e de I estão cruzadas. O elemento B deveria terminar antes do I iniciar, ou iniciar depois do início de I, ou ainda fechar depois do fecho de I.

- Os valores de atributos devem estar dentro de aspas. Se numa dada situação o valor do atributo necessitar de conter aspas, então deverão ser usados apóstrofes (ou aspas simples) para limitar o valor, como se pode verificar no exemplo apresentado de seguida:

```
<citação texto='O miúdo falou:"O Rei vai nú!"`/>
```

4.8 *NameSpaces*

Ao usar XML, o utilizador tem liberdade total na definição da sua linguagem, podendo atribuir os nomes que considere relevantes aos seus elementos e atributos. No entanto, à medida que o número de utilizadores e de aplicações XML foi crescendo, surgiram conflitos nos nomes dos elementos quando num documento XML se importam excertos de outros documentos XML escritos por outros autores. Poderão ainda surgir conflitos como anotações com o mesmo nome, mas com semânticas diferentes ou utilizadas em contextos diferentes. Consideremos por exemplo, o caso do elemento livro usado em dois contextos diferentes: num catálogo e num pedido de encomenda.

Elemento livro no catálogo:

```
<livro>
   <titulo>XML: da teoria à prática</titulo>
     <temas>
        <tema>XML</tema>
        <tema>XSL</tema>
   </temas>
</livro>
```

Elemento livro no pedido de encomenda:

```
<encomenda>
  ... pagamento e envio ...
  <item>
    <livro>
      <titulo>XML: da teoria à prática</titulo>
      <isbn>1-999999-88-7</isbn>
    </livro>
    <quantidade>4</quantidade>
    <preço EUR="17 euros"/>
</item>
</encomenda>
```

O elemento `<livro>` apresenta duas semânticas diferentes nos dois contextos descritos. Perante um outro documento formado com elementos de ambos os documentos apresentados ter-se-ia alguma dificuldade em interpretar os componentes livro que surgissem, pois não seria possível identificar a proveniência desses elementos livro.

Os *NameSpaces* surgem com o intuito de resolver este problema. Um *NameSpace* pode ser visto como uma super-etiqueta formada pelo nome da etiqueta ao qual é concatenado um prefixo. Esse prefixo é definido pelo utilizador e deverá ser único. Para garantir tal unicidade, convencionou-se que se usaria a sintaxe dos URI[5] para o prefixo. Assim, partindo do princípio de que cada utilizador tem um URI próprio, limitam-se os conflitos.

4.8.1 Criação de *NameSpaces*

O World Wide Web Consortium (W3C) publicou uma recomendação com o título *Namespaces* in XML[4], onde se define um *Namespace* como sendo uma coleção de nomes, identificados por uma

[5] Universal Resource Identifier

referência URI (Universal Resource Identifier) que é usada nos documentos XML como prefixo dos nomes de elementos e de atributos.

A mesma recomendação define o modo como devem ser criados e declarados os *Namespaces*. Para esse efeito, convencionou-se existir um atributo global (pode ser instanciado em qualquer elemento do documento) de nome **xmlns**. É com este atributo que se define um *NameSpace*.

No exemplo seguinte, apresentam-se algumas declarações de *NameSpaces* válidos.

```
xmlns="http://XMLSamples/livro.xsd"
xmlns="livro.xsd"
xmlns="URL/XSDs/livro.xsd"
```

As três declarações são válidas, apesar de só a primeira seguir a recomendação de utilizar o formato de um URI para o identificador do *NameSpace*.

4.8.2 Prefixos

A designação de um *NameSpace* é demasiado extensa para ser manipulada diretamente. Assim, para que a utilização de um *NameSpace* seja viável, associa-se uma abreviatura (um identificador ou qualificador do nome) que pode ser utilizado como referência ao *NameSpace*. Este qualificador é designado por **prefixo**. Em relação ao último exemplo, os NameSpaces podiam ter sido declarados da seguinte forma:

```
xmlns:catalogo="http://XMLSamples/livro.xsd"
xmlns:encomenda="livro.xsd"
xmlns:exp="URL/XSDs/livro.xsd"
```

Neste caso, **catalogo**, **encomenda** e **exp** seriam abreviaturas dos respetivos *NameSpaces*. Imaginemos agora um cenário ligeiramente diferente: consideremos um catálogo de livros em XML ao qual queremos acrescentar os nossos comentários usando as nossas anotações para comentários. Será possível utilizar os *NameSpaces* da seguinte forma:

```
<catalogo:livro xmlns:catalogo="http://xml.livrariaonline.pt/Samples/livro"
     xmlns:exp="http://exp.livraria.pt/coment">

<catalogo:titulo>XML: da teoria à prática</catalogo:titulo>
<exp:opiniao>Um bom ponto de partida ...</exp:opiniao>
<catalogo:resumo>Livro sobre a temática XML.</catalogo:resumo>
...
</catalogo:livro>
```

4.8.3 *NameSpaces* locais

Como referido anteriormente, o atributo **xmlns** pode ser utilizado em qualquer elemento e não apenas no elemento raiz. Assim, e para o tornar mais claro, o exemplo anterior poderia ser reescrito da seguinte forma:

```
<catalogo:livro xmlns:catalogo="http://exp.livraria.pt/livro">
   <catalogo:titulo>XML:da teoria à prática</catalogo:titulo>
   <exp:opinião xmlns:exp="http://exp.livraria.pt/coment.dtd">
   Um bom ponto de partida...</exp:opiniao>
   <catalogo:resumo>Livro sobre a temática XML.</catalogo:resumo>
...
</catalogo:livro>
```

A declaração do *Namespace* ficou agora junto do elemento que faz uso dela (`exp:opiniao`).

4.8.4 NameSpaces por omissão

O atributo **xmlns** pode ser utilizado isoladamente sem a declaração de um prefixo. Quando nenhum prefixo é especificado diz-se que foi declarado um **NameSpace por omissão**. Todos os elementos filho do elemento onde o *NameSpace* foi declarado pertencem agora a esse *NameSpace*. O mesmo exemplo do catálogo com comentários pode então ser reescrito da seguinte forma:

```
<livro
   xmlns="http://exp.livraria.pt/livro ">
   <titulo>XML: da teoria à prática</titulo>
   <exp:opiniao
   xmlns:exp=" http://exp.livraria.pt/coment">
   Um bom ponto de partida ...</exp:opiniao>
   <resumo>Livro que cobre a temática XML.</resumo>
   ...
</livro>
```

Como o *Namespace* associado a livro foi declarado sem prefixo, todos os elementos filhos de livro são considerados como pertencentes a esse *NameSpace*, à excepção daqueles que tenham um *NameSpace* declarado localmente (como é o caso de `exp:opiniao`).

5. Validação de documentos XML

Em muitas aplicações XML, os documentos são gerados automaticamente. Nestas aplicações, o gerador garante por si só a boa-formação e a validação dos documentos. Porém, na maioria dos casos em que é o utilizador a editar e manipular os documentos XML, surge a necessidade de estipular um conjunto de regras que estabeleçam a validade dos documentos. Este conjunto de regras define uma classe, ou tipo de documento e permite a validação e processamento dos documentos criados.

À especificação de um tipo de documentos dá-se o nome de **DTD** (Document Type Definition) ou XML Schema (evolução do DTD). O processo que verifica se um documento está de acordo com um DTD ou **Schema** designa-se por **Validação**. A um documento depois de verificado atribuímos a classificação de **documento XML Válido**.

5.1 *XML-Schema*

Um documento *Schema* especifica as regras para a **validação** de um documento XML, nomeadamente:

- os elementos e atributos que podem constar do documento
- a ordem e número dos elementos filhos
- tipos de dados para os elementos e atributos
- se um elemento é vazio ou pode incluir texto
- valores por omissão e fixos para elementos e atributos

Um *Schema* utiliza um sistema de tipos baseado na norma ISO 11404 *(Language Independent Data Types).* Para além do tipo de dados simples, a *XML-Schema* permite também que se construam tipos

de dados compostos ou dados complexos destinados a refletir estruturas de grau mais elevado nos documentos XML. Um *Schema XML* é uma evolução e alternativa ao DTD baseado em XML[3].

Um *Schema* descreve a estrutura e as regras de estruturação de objetos, indicando quais os elementos permitidos assim como as combinações entre eles. Pode também definir valores por omissão para os atributos. Um *Schema XML* descreve a estrutura de um documento XML. A linguagem *XML Schema* é também designada de XML Schema Definition (XSD).

Um *schema* descreve a gramática e a estrutura dos componentes de um documento XML. Os schemas usam uma gramática formalizada para dar expressão a essas estruturas; é escrito em sintaxe XML e pode ser um *schema* interno se vem incorporado dentro de um documento XML (ficheiro com sufixo `.xml`). Mas normalmente é ativado por referência a um *schema* externo (ficheiro com o sufixo `.xsd`). Assim, um grupo de documentos XML pode partilhar o mesmo *schema*; os vários documentos XML podem ter conteúdos diferentes, mas esses conteúdos podem ser sempre processados pela mesma aplicação, pois obedecem às mesmas regras de constituição e de estruturação.

Considere-se o seguinte exemplo de um documento XML simples designado "`mensagem.xml`":

```
<?xml version="1.0"?>
  <mensagem>
       <to>alunos</to>
       <from>AMD</from>
       <heading>Lembrete</heading>
       <body>A Ficha Formativa já está disponível!</body>
  </mensagem>
```

Listagem 5.1: mensagem.xml

Para escrever um *schema* para este documento, segue-se simplesmente a sua estrutura e define-se cada elemento pela ordem em que surge. O exemplo apresentado de seguida é um documento *XML Schema* chamado "`mensagem.xsd`" que define os elementos do documento XML acima apresentado ("mensagem.xml").

```
<?xml version="1.0"?>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3schools.com"
  xmlns="http://www.w3schools.com"
  elementFormDefault="qualified">
     <xsd:element name="mensagem">
     <xsd:complexType>
       <xsd:sequence>
          <xsd:element name="to" type="xsd:string"/>
          <xsd:element name="from" type="xsd:string"/>
          <xsd:element name="heading" type="xsd:string"/>
          <xsd:element name="body" type="xsd:string"/>
       </xsd:sequence>
     </xsd:complexType>
     </xsd:element>
  </xsd:schema>
```

Listagem 5.1: mensagem.xsd

O elemento <mensagem> é considerado um tipo complexo porque contém outros elementos. Os outros elementos são ditos tipos simples porque não contém outros elementos.

Elementos globais são aqueles que são filhos imediatos do elemento *schema*. **Elementos locais** são elementos encaixados a outros elementos.

5.1.1 Referênciar um XML Schema

Os documentos XML podem referenciar um XML *Schema*. O documento seguinte contém uma referência a um XML Schema:

```
<?xml version="1.0"?>
<mensagem xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" http://www.w3schools.com mensagens.xsd”>
    <to>LPROG0809</to>
    <from>amd</from>
    <heading>Lembrete</heading>
    <body> A Ficha Formativa já está disponível!!</body>
</mensagem >
```

O excerto `xmlns="http://www.w3schools.com"` especifica a declaração do *namespace* por omissão. Esta declaração indica ao validador do schema que os elementos usados neste documento XML são declarados no *namespace* "`http://www.w3schools.com`".

Considerando a existência de uma instância *XML Schema* do *namespace* disponível: xmlns:`xsi=http://www.w3.org/2001/XMLSchema-instance`, pode usar-se o atributo *schemaLocation* (`xsi:schemaLocation="`**`http://www.w3schools.com mensagens.xsd`**`">`). Este atributo assume dois valores. O primeiro é o *namespace* usado (**`http://www.w3schools.com`**). O segundo é a localização do schema XML a ser usado pelo *namespace*: **`mensagens.xsd`**.

5.1.2 Elementos XSD Simples

Um XML Schema define os elementos de um arquivo XML. Um **elemento simples** é um elemento XML que contém apenas texto. Ele não pode conter outros elementos ou atributos.

A restrição "apenas texto" é pouco clara. O texto pode ser de diferentes tipos. Pode ser um dos tipos básicos na definição de XML Schema (*boolean*, *string*, *date*, etc.) ou um tipo definido pelo utilizador.

```
<xsd:element name="titulo" type="xsd:string"/>
<xsd:element name="autor" type="xsd:string"/>
```

A sintaxe para definir um elemento simples é:

```
<xsd:element name="xxx" type="yyy"/>
```

onde **`xxx`** é o nome do elemento e **`yyy`** é o tipo de dados do elemento.

Elementos XML	Definições
<lastname>Silva</lastname> <age>34</age> <dateborn>1968-03-27</dateborn>	<xsd:element name="lastname" type"xsd:string"/> <xsd:element name="age" type="xsd:integer"/> <xsd:element name="dateborn" type="xsd:date"/>

Um XML Schema tem vários tipos de dados próprios - tipos **simples primitivos**. Referem-se os seguintes:

- xsd:string
- xsd:decimal
- xsd:integer
- xsd:boolean
- xsd:date
- xsd:time

Os elementos simples podem apresentar um conjunto de valores por **omissão** ou **fixos**. Um **valor por omissão** é automaticamente atribuído ao elemento quando nenhum outro valor for especificado. No exemplo seguinte, o valor por omissãoé "red":

```
<xsd:element name="color" type="xsd:string" default="red"/>
```

Um **valor fixo** é também atribuído automaticamente ao elemento. Não é possível especificar outro valor. No exemplo a seguir, o valor fixo é "red":

```
<xsd:element name="color" type="xsd:string" fixed="red"/>
```

5.1.3 Restrições de conteúdo/facets XSD

Quando um elemento ou um atributo XML tem um tipo definido, este cria uma restrição ao respetivo conteúdo. Se um elemento XML é do tipo "xsd:date" e contém um *string* como "Olá mãe", o elemento não vai ser validado. É possível acrescentar restrições aos elementos e atributos XML. As restrições são usadas para controlar os valores aceites para elementos e atributos XML. Restrições em elementos XML são designadas *facets*. O elemento restrição indica que o tipo de dados é derivado de um namespace da W3C XML Schema. Assim, <xsd:restriction base="xsd:string">, significa que o valor do elemento ou atributo deve ser uma string. Na tabela seguinte é apresentado um resumo das restrições de conteúdos definidos no âmbito do XSD.

Tabela 5.1 - Restrições de conteúdos

Restrição	Descrição
Enumeration	Define uma lista de valores válidos
fractionDigits	Específico o número máximo de casas decimais permitidas. Deve ser maior ou igual que zero.
Length	Especifica o número exacto de caracteres ou itens permitidos. Deve ser maior ou igual que zero.
maxExclusive	Especifica o valor máximo para valores numéricos (o valor deve ser menor que este valor).
maxInclusive	Especifica o valor máximo para valores numéricos (o valor deve ser menor ou igual a este valor).
maxLength	Especifica o número máximo de caracteres ou itens permitidos. Deve ser maior ou igual que zero.
minExclusive	Especifica o valor mínimo para valores numéricos (o valor deve ser maior que este valor).
minInclusive	Especifica o valor mínimo para valores numéricos (o valor deve ser maior ou igual a este valor)
minLength	Especifica o número mínimo de caracteres ou itens permitidos. Deve ser maior ou igual que zero.
Pattern	Define a sequência exacta de caracteres permitidos
totalDigits	Especifica o número exacto de dígitos permitidos. Deve ser maior que zero
whiteSpace	Especifica como caracteres vazios (tabs, espaços e retornos de carro) são tratados

Restrições de valores

A restrição de valores define o intervalo de valores que o elemento ou atributo pode assumir. Tal é definido através dos atributos `minInclusive e maxInclusive`. O exemplo define um elemento chamado "idade" com uma restrição. O valor da idade não pode ser menor que 0 ou maior que 100.

```
<xsd:element name="idade">
  <xsd:simpleType>
   <xsd:restriction base="xsd:integer">
       <xsd:minInclusive value="0"/>
       <xsd:maxInclusive value="100"/>
   </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Restrições em conjuntos de valores

Para limitar o conteúdo de um elemento XML num conjunto de valores aceitáveis, é possível definir restrições enumeradas. O exemplo seguinte define um elemento chamado "carro":

```
<xsd:element name="carro">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
       <xsd:enumeration value="Audi"/>
       <xsd:enumeration value="Golf"/>
       <xsd:enumeration value="BMW"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

O elemento "carro" é um tipo simples com restrições. Os valores aceitáveis são: Audi, Golf, BMW. O exemplo acima também poderia ser escrito da seguinte forma.

```
<xsd:element name="carro">
   <xsd:simpleType name="carType">
     <xsd:restriction base="xsd:string">
         <xsd:enumeration value="Audi"/>
         <xsd:enumeration value="Golf"/>
         <xsd:enumeration value="BMW"/>
     </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

Neste caso o tipo "carType" pode ser usado por outros elementos, porque ele não é parte do elemento "carro".

Restrições em séries de valores ou Padrão

Para limitar o conteúdo de um elemento XML numa série de números ou letras, podemos utilizar a restrição padrão. O exemplo seguinte define o elemento chamado "letra":

```
<xsd:element name="letra">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[a-z]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

O elemento <letra> é do tipo simples com uma restrição. O único valor aceitável é uma das letras minúsculas de **a** até **z**.

O exemplo seguinte define um elemento chamado <iniciais> :

```
<xsd:element name="iniciais">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
       <xsd:pattern value="[A-Z][A-Z][A-Z]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

O elemento <iniciais> é simples com uma restrição. O único valor aceitável são três letras maiúsculas de A até Z.

Outras restrições em séries de valores

Outras restrições podem ser definidas por restrição de padrão através da utilização dos caracteres especiais “*” (zero ou mais ocorrências) e “+” (é uma ou mais ocorrências):

O exemplo seguinte define um elemento chamado <letra>:

```
<xsd:element name="letra">
   <xsd:simpleType>
     <xsd:restriction base="xsd:string">
        <xsd:pattern value="([a-z])*"/>
     </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

O elemento <letra> é um tipo simples com uma restrição. O valor aceitável é zero ou mais ocorrências de letras minúsculas de **a** até **z**.

O exemplo seguinte define um elemento chamado "genero":

```
<xsd:element name="genero">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
         <xsd:pattern value="masculino|feminino"/>
     </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

O elemento <género> é um tipo simples com uma restrição. O único valor aceitável é masculino ou feminino.

O exemplo seguinte define um elemento chamado <password>:

```
<xsd:element name="password">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
         <xsd:pattern value="[a-zA-Z0-9]{8}"/>
     </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

O elemento <password> é um tipo simples com uma restrição. Deve haver exatamente oito caracteres e estes caracteres devem ser letras minúsculas ou maiúsculas de a até z, ou um dígito de 0 a 9.

Restrições em caracteres vazios

Para especificar como um carácter vazio deve ser tratado, deve ser usada a restrição ***whiteSpace***. Esta restrição pode assumir os valores especificados de seguida:

- "**preserve**" - significa que o processador XML não vai remover nenhum carácter vazio
- "**replace**" - significa que o processador XML vai substituir todos os caracteres vazios (quebras de linha, tabs, espaços) por espaços.
- "**collapse**" - significa que o processador XML vai remover todos os caracteres vazios (quebras de linha, tabs, espaços são substituídos com espaços, espaços iniciais e finais são removidos, espaços múltiplos são reduzidos a um).

O exemplo seguinte define um elemento chamado <endereço> com uma restrição:

```
<xsd:element name="endereço">
  <xsd:simpleType>
     <xsd:restriction base="xsd:string">
         <xsd:whiteSpace value="collapse"/>
     </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

A restrição **whiteSpace** é definida como "**collapse**", que significa que o processador XML vai remover todos os caracteres vazios (quebras de linha, tabs, espaços são substituídos com espaços, espaços iniciais e finais são removidos, espaços múltiplos são reduzidos a um).

Restrições de comprimento

Para limitar o comprimento de um elemento, usam-se as restrições de comprimento, ***maxLength*** e ***minLength***. O exemplo seguinte define um elemento chamado <password>:

```
<xsd:element name="password">
   <xsd:simpleType>
     <xsd:restriction base="xsd:string">
        <xsd:length value="8"/>
     </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

O elemento <password>é um tipo simples com uma restrição. O valor deve ter exactamente oito caracteres. O exemplo seguinte define outro elemento chamado <password>:

```
<xsd:element name="password">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
         <xsd:minLength value="5"/>
         <xsd:maxLength value="8"/>
     </xsd:restriction>
    </xsd:simpleType>
 </xsd:element>
```

Neste caso, o elemento <password> é um tipo simples com uma restrição. O valor deve ter no mínimo cinco e no máximo oito caracteres.

5.1.4 Atributos XSD

Todos os atributos são declarados como tipos simples. Apenas os elementos complexos têm atributos. Elementos simples não podem ter atributos. Um elemento com atributos tem sempre uma definição do tipo complexo. A definição dos atributos num tipo complexo é realizada após a definição dos elementos.

A sintaxe para definir um atributo é:

```
<xsd:attribute name="xxx" type="yyy"/>
```

onde **xxx** é o nome do atributo e **yyy** é o tipo de dados do atributo.

Apresenta-se de seguida um exemplo de um elemento XML com um atributo:

```
<lastname lang="pt”>Silva</lastname>
```

E a definição correspondente do atributo:

```
<xsd:attribute name="lang" type="xsd:string"/>
```

Os atributos podem ter valores por omissão ou fixos predefinidos. Um valor por omissão é atribuído automaticamente ao atributo quando nenhum outro valor for especificado. No exemplo seguinte, o valor por omissão é "pt":

```
<xsd:attribute name="lang" type="xsd:string" default="pt">
```

Um valor fixo é atribuído automaticamente ao atributo. Não pode ser especificado outro valor. No exemplo seguinte, o valor fixo é "pt":

```
<xsd:attribute name="lang" type="xsd:string" fixed="pt"/>
```

Todos atributos são opcionais por omissão. Para especificar explicitamente que um atributo é opcional, deve usar-se o atributo "**use**" com o valor "**optional**":

```
<xsd:attribute name="lang" type="xsd:string" use="optional"/>
```

Para especificar que um atributo deve ser obrigatório usar "**use**"com o valor "**required**":

```
<xsd:atrribute name="lang" type="xsd:string" use="required"/>
```

5.1.5 Elementos XSD complexos

Um elemento complexo é um elemento XML que contém outros elementos e/ou atributos. Cada elemento pode conter atributos.

Identificam-se quatro tipos de elementos complexos:

- elementos vazios

```
<product pid="1245"/>
```

- elementos que contém apenas outros elementos

```
<funcionário>
    <firstname>João</firstname>
    <lastname>Hugo</lastname>
</funcionário >
```

- elementos com atributos

```
<comida type="sobremesa">Ice cream</comida>
```

- elementos que contém outros elementos e texto

```
<descrição>
  It happened on <date lang="pt">03.03.99</date> ....
</descrição>
```

5.1.5.1 Definição de elementos complexos

A - Elementos complexos que contém outros elementos e atributos

Observe este elemento XML complexo <funcionário>, que contém apenas outros elementos:

```
<funcionário>
    <firstname>João</firstname>
    <lastname>Hugo</lastname>
</funcionário>
```

É possível definir um elemento complexo num XML Schema de duas formas diferentes:

1. O elemento <funcionário> pode ser declarado directamente, designando o elemento:

```
<xsd:element name=" funcionário">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element name="firstname" type="xsd:string"/>
           <xsd:element name="lastname" type="xsd:string"/>
       </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Se for usado o método descrito anteriormente, apenas o elemento <funcionário> pode usar o tipo complexo definido. Refira-se que os elementos filhos, <firstname> e <lastname>, são envolvidos pela etiqueta <sequence>. Isto significa que os elementos filhos devem aparecer na mesma ordem da declaração: <firstname> primeiro e <lastname> depois.

2. O elemento <funcionário> pode ter um atributo tipo que faz referência ao nome do tipo complexo que deve ser usado:

```
<xsd:element name="funcionário" type="pessoa"/>

<xsd:complexType name="pessoa">
   <xsd:sequence>
       <xsd:element name="firstname" type="xsd:string"/>
       <xsd:element name="lastname" type="xsd:string”/>
   </xsd:sequence>
</xsd:complexType>
```

Se for usado o método descrito anteriormente, vários elementos podem fazer referência ao mesmo tipo complexo, da seguinte forma:

```
<xsd:element name="funcionário" type="pessoa"/>
<xsd:element name="estudante" type="pessoa"/>
<xsd:element name="membro" type="pessoa"/>
<xsd:complexType name="pessoa">
    <xsd:sequence>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
    </xsd:sequence>
 </xsd:complexType>
```

Pode-se basear um elemento do tipo complexo num tipo complexo já existente e acrescentar alguns elementos, da seguinte forma:

```
<xsd:element name="employee" type="fullpersoninfo"/>
<xsd:complexType name="personinfo">
   <xsd:sequence>
       <xsd:element name="firstname" type="xsd:string"/>
       <xsd:element name="personinfo" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fullpersoninfo">
   <xsd:complexContent>
       <xsd:extension base="personinfo">
           <xsd:sequence>
               <xsd:element name="address" type="xsd:string"/>
               <xsd:element name="city" type="xsd:string"/>
               <xsd:element name="country" type="xsd:string"/>
           </xsd:sequence>
       </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

3. O elemento <funcionário> pode ter um atributo id:

```
<xsd:element name="funcionário" type="pessoa"/>

<xsd:complexType name="pessoa">
   <xsd:sequence>
       <xsd:element name="firstname" type="xsd:string"/>
       <xsd:element name="lastname" type="xsd:string”/>
   </xsd:sequence>
   <xsd:attribute name="id" type="xsd:positiveInteger"/>
</xsd:complexType>
```

B - Elementos complexos vazios

Um elemento complexo vazio pode conter atributos, mas não pode ter nenhum conteúdo entre as etiquetas de início e de fim.

Um elemento XML vazio:

```
<product prodid="1345"/>
```

O elemento "product" não tem conteúdo. Para definir um tipo sem conteúdo, deve-se definir um tipo que permita apenas atributos no seu conteúdo.

```
<xsd:element name="product">
  <xsd:complexType>
      <xsd:attribute name="prodid" type="xsd:positiveInteger"/>
  </xsd:complexType>
</xsd:element>
```

C - Tipos complexos que contêm apenas outros elementos

Um tipo complexo que contém apenas outros elementos. O elemento XML <pessoa> contém apenas outros elementos:

```
<pessoa>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</person>
```

O *schema* para o documento anterior poderia ser definido da seguinte forma:

```
<xsd:element name="pessoa">
    <xsd:complexType>
       <xsd:sequence>
            <xsd:element name="firstname" type="xsd:string"/>
            <xsd:element name="lastname" type="xsd:string"/>
       </xsd:sequence>
    </xsd:complexType>
 </xsd:element>
```

Refira-se que a anotação **<xsd:sequence>** significa que os elementos definidos (<firstname> e <lastname>) devem aparecer na ordem em que estão definidas no elemento <pessoa>.

Em alternativa pode definir-se um nome para o **complexType** e um atributo **type** ao elemento <pessoa> que faz referência a este. Desta forma vários elementos podem referenciar o mesmo tipo complexo.

```
<xsd:element name="pessoa" type="pessoatipo"/>

<xsd:complexType name="pessoatipo">
   <xsd:sequence>
       <xsd:element name="firstname" type="xsd:string"/>
       <xsd:element name="lastname" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
```

D - Elementos complexos com apenas texto

Um elemento complexo com apenas texto pode conter atributos e texto. Acrescenta-se um elemento **simpleContent** no conteúdo. Quando se usar conteúdo simples, é necessário definir uma extensão ou uma restrição com o elemento **simpleContent**, da seguinte forma:

```
<xsd:element name="somename">
   <xsd:complexType>
       <xsd:simpleContent>
           <xsd:extension base="basetype">
               ....
           </xsd:extension>
       </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

Ou então:

```
<xsd:element name="somename">

   <xsd:complexType>
       <xsd:simpleContent>
           <xsd:restriction base="basetype">
               ....
               ....
           </xsd:restriction>
       </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

Usa-se o elemento ***extension*** para expandir o tipo simples base de um elemento, e o elemento ***restriction*** para limitá-lo.

De seguida apresenta-se o exemplo de um elemento XML "tamanho" que contém apenas texto:

```
<tamanho country="portugal">35</tamanho>
```

No exemplo a seguinte declara-se um *complexType* "tamanho". O conteúdo é definido com o tipo de dados *integer* e o elemento "tamanho" também contêm um atributo chamado "país":

```
<xsd:element name="tamanho">
   <xsd:complexType>
       <xsd:simpleContent>
           <xsd:extension base="xsd:integer">
               <xsd:attribute name="pais" type="xsd:string"/>
           </xsd:extension>
       </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

É também possível definir um nome para o ***complexType*** e especificar um atributo *type* ao elemento "tamanho " que faz referência ao nome do ***complexType***. Desta forma vários elementos podem referenciar o mesmo tipo complexo.

```
<xsd:element name="tamanho" type="tamanhotipo"/>
  <xsd:complexType name="tamanhotipo">
   <xsd:simpleContent>
       <xsd:extension base="xsd:integer">
           <xsd:attribute name="pais" type="xsd:string"/>
       </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

E- Tipos complexos com conteúdo misto

Um elemento do tipo **complexo misto** pode conter atributos, elementos, e texto. Considere o elemento XML, <carta>, que contém outros elementos e texto:

```
<carta>
  Caro Sr.<name>João Silva</name>
  A sua encomenda <orderid>1032</orderid>
  Será enviada a <shipdate>2001-07-13</shipdate>.
</carta>
```

O texto que aparece entre os elementos: <name>, <orderid>, e <shipdate> são todos filhos de <carta>. O seguinte *schema* declara o elemento <letra>:

```
<xsd:element name="carta">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="orderid" type="xsd:positiveInteger"/>
            <xsd:element name="shipdate" type="xsd:date"/>
        </xsd:sequence>
    </xsd:complexType>
 </xsd:element>
```

Para permitir que apareçam caracteres entre os elementos filhos de <carta>, o atributo *mixed* deve ser definido a "*true*".

Se for atribuído um nome ao elemento <xsd:complexType> e definido o atributo *type* <lettertype > como sendo uma referência a este, vários elementos podem fazer referência ao mesmo <xsd:complexType>, como apresentado de seguida:

```
<xsd:element name="letter" type="lettertype"/>
<xsd:complexType name="lettertype" mixed="true">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="orderid" type="xsd:positiveInteger"/>
        <xsd:element name="shipdate" type="xsd:date"/>
    </xsd:sequence>
</xsd:complexType>
```

5.1.5.2 Indicadores de tipos complexos

É possível controlar a forma como os elementos serão usados em documentos através de indicadores:

- **Indicadores de ocorrência**: ***maxOccurs*** (especifica o número máximo de vezes que um elemento pode ocorrer) e ***minOccurs*** (especifica o número mínimo de vezes que um

elemento pode ocorrer). Por omissão, o valor destes atributos para qualquer dos indicadores de ordem é 1.

- **Indicadores de ordem** são usados para definir a ordem em que os elementos ocorrem.
  - ***All*** - especifica que os elementos filhos podem aparecer em qualquer ordem e que cada um deve ocorrer apenas uma vez (`minOccurs=0|1` e `maxOccurs= 1`). Por omissão, o valor destes atributos/indicadores de ocorrência é 1.

```
<xsd:element name="person">
  <xsd:complexType>
     <xsd:all>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
     </xsd:all>
  </xsd:complexType>
</xsd:element>
```

  - **Choice** - especifica que o conteúdo é formado por apenas um elemento dos elementos alternativos (`minOccurs= nonNegativeInteger` e `maxOccurs= nonNegativeInteger|unbounded`). Por omissão, o valor destes atributos/indicadores de ocorrência é 1.

```
<xsd:element name="identificacao">
  <xsd:complexType>
     <xsd:choice>
        <xsd:element name="bi" type="BI"/>
        <xsd:element name="cc" type="CC"/>
     </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

  - ***Sequence*** - especifica que os elementos filhos devem aparecer na ordem definida (`minOccurs=nonNegativeInteger` e `maxOccurs=nonNegativeInteger|unbounded`). Por omissão, o valor dos atributos/indicadores de ocorrência é 1.

```
<xsd:element name="nome">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="primeiro" type="xsd:string"/>
         <xsd:element name="ultimo" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

A especificação do número de ocorrências de um determinado elemento é realizada através dos atributos *minOccurs* e *maxOccurs* (indicadores de ocorrência). A especificação do atributo *minOccurs* = "0" indica que o elemento é opcional. A colocação do atributo *maxOccurs* = "*unbounded*" indica que não há limite para o número de vezes que o elemento pode aparecer. O valor por omissão para os atributos `minOccurs` e `maxOccurs` é 1, que significa que o elemento tem de ocorrer exatamente 1 vez.
Quando só é especificado um valor para o atributo minOccurs, este tem de ser menor ou igual ao valor por omissão de `maxOccurs`, isto é 0 ou 1. Da mesma forma quando só é

especificado um valor para o atributo `maxOccurs`, este tem de ser maior ou igual ao valor por omissão de `minOccurs`, isto é superior ou igual.
Os indicadores de ocorrência podem ser associados aos elementos ou aos indicadores de ordem. O exemplo indica que o elemento <child_name> pode ocorrer num elemento <pessoa> no mínimo zero vezes e no máximo dez vezes. Para permitir que um elemento apareça um número ilimitado de vezes, deve ser usada a instrução **maxOccurs="unbounded"**.

```
<xsd:element name="pessoa">
     <xsd:complexType>
         <xsd:sequence>
             <xsd:element name="full_name" type="xsd:string"/>
             <xsd:element name="child_name" type="xsd:string"
             maxOccurs="10" minOccurs="0"/>
         </xsd:sequence>
     </xsd:complexType>
  </xsd:element>
```

O exemplo seguinte indica que o elemento <pessoas> pode possuir no mínimo zero ocorrências dos elementos <full_name> e <child_name> na sequência especificada, e no máximo dez vezes. Para permitir que um elemento apareça um número ilimitado de vezes, deve ser usada a instrução **maxOccurs="unbounded"**.

```
<xsd:element name="pessoas">
   <xsd:complexType>
       <xsd:sequence maxOccurs="10" minOccurs="0"/>
           <xsd:element name="full_name" type="xsd:string"/>
           <xsd:element name="child_name" type="xsd:string"/>
       </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Os elementos filhos de um `sequence` ou `choice` podem ser: `group`, `choice`, `sequence`, `complexType`, `restriction` (both `simpleContent` and `complexContent`), `extension` (both `simpleContent` and `complexContent`. Enquanto num all podem ser: `group`, `complexType`, `restriction` (both `simpleContent` and `complexContent`), `extension` (both `simpleContent` and `complexContent`.

- **Indicadores de grupo** são usados para definir grupos de elementos ou de atributos relacionados.
    - ***Group name*** - usado para definir grupos de elementos.

```
<xsd:group name="persongroup">
     <xsd:sequence>
         <xsd:element name="firstname" type="xsd:string"/>
         <xsd:element name="lastname" type="xsd:string"/>
         <xsd:element name="birthday" type="xsd:date"/>
     </xsd:sequence>
</xsd:group>
```

O grupo definido pode ser referenciado na definição de outro grupo ou de um tipo complexo.

```
<xsd:element name="person" type="personinfo"/>

<xsd:complexType name="personinfo">
  <xsd:sequence>
    <xsd:group ref="persongroup"/>
    <xsd:element name="country" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

- ***AttributeGroup name*** - usados para definir grupos de atributos.

```
<xsd:attributeGroup name="personattrgroup">
      <xsd:attribute name="firstname" type="xsd:string"/>
      <xsd:attribute name="lastname" type="xsd:string"/>
      <xsd:attribute name="birthday" type="xsd:date"/>
</xsd:attributeGroup>
```

O grupo definido pode ser referenciado na definição de outro grupo ou de um tipo complexo.

```
<xsd:element name="person">
  <xsd:complexType>
     <xsd:attributeGroup ref="personattrgroup"/>
  </xsd:complexType>
</xsd:element>
```

5.1.6 A Construção de um Schema

Vejamos o exemplo de um documento XML simples, chamado "*shiporder.xml*", que descreve um sistema de expedição de encomendas.

```
<?xml version="1.0" encoding="UTF-8"?>
    <shiporder orderid="889923"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="shiporder.xsd">
    <orderperson>John Smith</orderperson>
    <shipto>
       <name>Ola Nordmann</name>
       <address>Langgt 23</address>
       <city>4000 Stavanger</city>
       <country>Norway</country>
    </shipto>
    <item>
       <title>Empire Burlesque</title>
       <note>Special Edition</note>
       <quantity>1</quantity>
       <price>10.90</price>
    </item>
    <item>
       <title>Hide your heart</title>
       <quantity>1</quantity>
       <price>9.90</price>
    </item>
  </shiporder>
```

Listagem 5.: "*shiporder.xml*"

O documento XML é constituído por um elemento raiz <shiporder>, que contém um atributo obrigatório chamado "orderid". O elemento <shiporder> possui três elementos filhos diferentes:

<orderperson>, <shipto> e <item>. O elemento "item" aparece duas vezes e contém um elemento <title>, um elemento opcional "note", um elemento <quantity> e um <price>.

A linha `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` diz ao parser XML que o documento deve ser validado por um *schema*. A linha: `xsi:noNamespaceSchemaLocation="shiporder.xsd"` especifica onde o *schema* reside (neste caso, ele está no mesmo directório que "shiporder.xml").

Para criar o *schema* "`shiporder.xsd`" podemos simplesmente seguir a estrutura no documento XML e definir cada elemento à medida que é encontrado. Começa-se com a declaração XML padrão, seguida do elemento `xsd:schema` que define um *schema*:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    ...
</xsd:schema>
```

No *schema* acima definido usa-se o namespace padrão (`xsd`), e a URI associada com este *namespace* na linguagem de definição *Schema*, que tem o valor padrão `http://www.w3.org/2001/XMLSchema`.

De seguida, é necessário definir o elemento <shiporder>. Este elemento tem um atributo e contém outros elementos (tipo complexo). Os elementos filhos de <shiporder> são englobados por um elemento `xsd:sequence` que define a respectiva ordem:

```
<xsd:element name="shiporder">
    <xsd:complexType>
        <xsd:sequence>
        ...
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

O elemento <orderperson> é definido como um tipo simples (porque ele não contém nenhum atributo ou elemento). O tipo (`xsd:string`) é definido com o prefixo do *namespace* associado ao *XML Schema* que indica um tipo de dados pré-definido:

```
<xsd:element name="orderperson" type="xsd:string"/>
```

Em seguida, é necessário definir dois elementos do tipo complexo: <shipto> e <item>. Começamos por definir o elemento <shipto>:

```
<xsd:element name="shipto">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="address" type="xsd:string"/>
            <xsd:element name="city" type="xsd:string"/>
            <xsd:element name="country" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

O elemento "item" pode aparecer muitas vezes dentro do elemento <shiporder>. Isto é especificado definindo o atributo ***maxOccurs*** do elemento <item> para "***unbounded***" que significa que podem haver tantas ocorrências do elemento <item> quanto o autor desejar. Refira-se que o elemento <note> é opcional (***minOccurs*** em zero):

```
<xsd:element name="item" maxOccurs="unbounded">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="title" type="xsd:string"/>
            <xsd:element name="note" type="xsd:string" minOccurs="0"/>
            <xsd:element name="quantity" type="xsd:positiveInteger"/>
            <xsd:element name="price" type="xsd:decimal"/>
        </xsd:sequence>
    </xsd:complexType>
 </xsd:element>
```

Falta apenas declarar o atributo do elemento <shiporder>. Como se trata de um atributo obrigatório especifica-se **use="required"**. Segundo o W3C XML Schema Working Group [2] as declarações de atributos devem vir sempre no fim.

```
<xsd:attribute name="orderid" type="xsd:string" use="required"/>
```

A forma descrita de anotar um *schema* segue a estrutura do documento. O processo básico consiste em definir cada elemento e cada atributo dentro do seu contexto e de permitir que ocorrências múltiplas do mesmo <elemento> possam obter definições diferentes.

Apresenta-se de seguida o *schema* "shiporder.xsd" para o documento XML anterior apresentado na listagem 5.17.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="shiporder">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="orderperson" type="xsd:string"/>
            <xsd:element name="shipto">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="name" type="xsd:string"/>
                        <xsd:element name="address" type="xsd:string"/>
                        <xsd:element name="city" type="xsd:string"/>
                        <xsd:element name="country" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="item" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="title" type="xsd:string"/>
                        <xsd:element name="note" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="quantity" type="xsd:positiveInteger"/>
                        <xsd:element name="price" type="xsd:decimal"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    <xsd:attribute name="orderid" type="xsd:string" use="required"/>
    </xsd:complexType>
 </xsd:element>
 </xsd:schema>
```

Listagem5.2: Schema "shiporder.xsd

Com Referência a outros elementos

O método descrito no exemplo anterior é muito simples mas pode tornar-se difícil de ler e manter perante documentos complexos. O método descrito de seguida baseia-se na definição de todos os elementos e atributos primeiro, com a referência a estes através do atributo ***ref***. Este atributo não pode ser usado se o elemento pai for o elemento de schema.

De seguida apresenta-se o novo documento *schema* ("shiporder.xsd"):

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of simple elements -->
<xsd:element name="orderperson" type="xsd:string"/>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="address" type="xsd:string"/>
<xsd:element name="city" type="xsd:string"/>
<xsd:element name="country" type="xsd:string"/>
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="note" type="xsd:string"/>
<xsd:element name="quantity" type="xsd:positiveInteger"/>
<xsd:element name="price" type="xsd:decimal"/>

<!-- definition of attributes -->
<xsd:attribute name="orderid" type="xsd:string"/>

<!-- definition of complex elements -->
<xsd:element name="shipto">
  <xsd:complexType>
     <xsd:sequence>
       <xsd:element ref="name"/>
       <xsd:element ref="address"/>
       <xsd:element ref="city"/>
       <xsd:element ref="country"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="item">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="title"/>
      <xsd:element ref="note" minOccurs="0"/>
      <xsd:element ref="quantity"/>
      <xsd:element ref="price"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="shiporder">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="orderperson"/>
      <xsd:element ref="shipto"/>
      <xsd:element ref="item" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute ref="orderid" use="required"/>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Listagem 5.3: XML schema "shiporder.xsd

Com Named Types

O terceiro método define classes e tipos que permitem reutilizar definições de elementos. Isto é realizado através da definição de ***simpleTypes*** e ***complexTypes***, e a sua posterior reutilização através do atributo ***type*** do respectivo elemento, como exemplificado no *schema* XML ("shiporder.xsd"):

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xsd:simpleType name="stringtype">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="inttype">
  <xsd:restriction base="xsd:positiveInteger"/>
</xsd:simpleType>

<xsd:simpleType name="dectype">
  <xsd:restriction base="xsd:decimal"/>
</xsd:simpleType>

<xsd:simpleType name="orderidtype">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9]{6}"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="shiptotype">
  <xsd:sequence>
    <xsd:element name="name" type="stringtype"/>
    <xsd:element name="address" type="stringtype"/>
    <xsd:element name="city" type="stringtype"/>
    <xsd:element name="country" type="stringtype"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="itemtype">
  <xsd:sequence>
    <xsd:element name="title" type="stringtype"/>
    <xsd:element name="note" type="stringtype" minOccurs="0"/>
    <xsd:element name="quantity" type="inttype"/>
    <xsd:element name="price" type="dectype"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="shipordertype">
  <xsd:sequence>
    <xsd:element name="orderperson" type="stringtype"/>
    <xsd:element name="shipto" type="shiptotype"/>
    <xsd:element name="item" maxOccurs="unbounded" type="itemtype"/>
  </xsd:sequence>
  <xsd:attribute name="orderid" type="orderidtype" use="required"/>
</xsd:complexType>

<xsd:element name="shiporder" type="shipordertype"/>
</xsd:schema>
```

Listagem 5.4: schema "shiporder.xsd

Bibliografia:

[1]. World Wide Web Consortium http://www.w3.org/ .

[2]. W3C XML Schema Working Group http://www.w3.org/2003/09/xmlap/xml-schema-wg-charter.html

[3]. XML Schema Tutorial http://www.w3schools.com/Schema/schema_schema.asp

[4]. Namespaces in XML 1.0 http://www.w3.org/TR/REC-xml-names/

[5]. Anders Møller and Michael I. Schwartzbach, An Introduction to XML and Web Technologies, Addison-Wesley, 2006.

[6]. Paulo Heitlinger, O Guia Prático da XML, Centro Atlântico, 2001.

[7]. XML e XSL: da Teoria à Prática, José Carlos Leite Ramalho, Pedro Rangel Henriques, FCA - EDITORA DE INFORMÁTICA, 2001.