UD_3_ Validación de documentos XML

A_1. A necesidade de crear documentos XML válidos. Validación empregando DTDs.

Índice

1. Validación de documentos XML 3

2. A validación por DTD 3

2.1 Os analizadores 4

3. Construción dunha DTD 4

4. Vincular unha DTD aos datos XML 5

4.1 Declaracións DTD dentro dun documento XML (DTD internas) 5

4.2 Declaracións DTD nun arquivo .dtd (DTD externas) 5

4.3 Declaracións DTD nun documento cunha referencia pública 6

5. Definir elementos 7

5.1 Modelos de contidos 7

6. Definir atributos 9

6.1 Lista de atributos 9

6.2 Os valores dos atributos IMPLIED, REQUIRED e FIXED 10

6.3 Tipos de atributos 10

6.3.1 Atributos CDATA, NMTOKEN e NMTOKENS 10

6.3.2 Atributos de tipo ID, IDREF ou IDREFS 11

6.3.3 Atributos de tipo ENTITY 12

6.3.4 Atributos de tipo NOTATION 12

6.4 Exemplo de DTD con atributos 12

7. As entidades 13

7.1 Entidades internas 14

7.2 Entidades externas 15

7.3 Entidades de parámetro 16

8. Notacións 17

9. Seccións condicionais 18

1. Validación de documentos XML

Un documento XML *ben estruturado*, pódese empregar nunha gran variedade de tarefas. O problema é que determinar se un documento XML está ben estruturado é algo que non é sinxelo de predecir.

Con XML pódense crear novos elementos e atributos libremente, poñéndolles os nomes que queiramos e organizando os elementos seguindo unha orde arbitraria e aniñándoos ata o nivel de profundidade que nos pareza máis convinte. A consecuencia disto é que os documentos XML non nos permiten saber se a súa estrutura é aceptable ou non. Para sabelo, deberiamos ter un xeito de especificar:

- Qué nomes se deben empregar para os elementos nun documento XML, cántas veces se poden usar e en qué orde.
- Qué nomes se deben empregar para os atributos, qué elementos os usan e se son obrigatorios ou opcionais.
- Os valores posibles, predeterminados ou permitidos que admiten os atributos.

Con isto, pódese describir cal é a estrutura dun documento XML dun determinado tipo, de xeito que distintos usuarios poderán crear documentos XML válidos que cumplan as mesmas regras.

Ao proceso de comparar un documento XML cun destes documentos de regras chámaselle **validación**.

Por exemplo, imaxinemos que se queren crear os apuntes deste módulo entre varios alumnos onde cada un destes elaborará un tema. Cada tema constará dun título, un autor, apartados, subapartados, párrafos de texto, párrafos con código (para os exemplos) e imaxes.

Estes contidos terán relacións entre eles, así, por exemplo:

- Un tema terá un título, un número, un autor e constará dun ou máis apartados que a súa vez se dividirán en subapartados.
- Os apartados e subapartados terán un título que aparecerá ao principio e que precederá a un ou máis párrafos de texto. Además poden conter párrafos con código e imaxes.

Imaxinemos que despois queremos obter os apuntes do módulo completo a partir dos temas que foron elaborando os alumnos en formato XML e que teñen a estrutura que acabamos de describir. Para implementar este tipo de estrutura teremos que *validar* todos os temas elaborados como documentos XML con algún tipo de definición destas regras que determinan cal debe ser a súa organización.

Estas definicións adoitanse expresar, ou ben mediante arquivos de definicións DTD ou ben mediante esquemas XML. Inda que existen outras opcións (RELAX NG, Schematron) os DTD e os esquemas son os máis usados.

Cada tipo de validación ten as súas vantaxes, e deberemos elixir entre un ou outro.

2. A validación por DTD

Xa vimos como crear un documento baseado na gramática especificada polo W3C. Un documento XML que respecta esta norma estará *ben formado*.

Cando un documento respecta a gramática descrita nun DTD (Document Type Definition), como por exemplo as normas que indicamos no punto anterior para especificar a estrutura dun tema, o documento será *válido*.

Xa vimos na unidade anterior que un documento non se considera "ben formado" se non respecta as normas descritas na especificación oficial de XML 1.0. Un documento XML ben formado, terá un prólogo como o seguinte:

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

O termino *standalone* permítenos especificar se o documento depende doutra gramática distinta ás normas impostas por XML. Se definimos standalone como "no" estaremos indicando que o documento depende dun documento distinto que especifica a gramática que o describe: DTD.

Un documento válido contén e respecta a gramática (DTD) que o describe, que é a que declara as normas de validez dun documento XML. Define, entre outras cousas, a orde de aparición e de imbricación dos elementos, os atributos e os seus tipos.

2.1 Os analizadores

A palabra analizador (parser) úsase frecuentemente cando se fala de XML, e é a ferramenta que permite interpretar un documento XML. Esta interpretación consiste a miúdo en transformar un documento XML textual nunha páxina Web. Os navegadores Web posúen o seu propio analizador XML.

O analizador debe detectar se se trata dun documento ben formado. Se un documento depende dunha DTD, o interpretados debe validar o cumprimento da gramática especificada.

3. Construción dunha DTD

Definir explicitamente unha linguaxe permite implementar procesos de validación que se axusten aos documentos. A continuación, imos ver un exemplo de uso dunha DTD para definir a gramática do seguinte documento XML de descrición de temas:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tema unidade="5" titulo="A linguaxe XML">
  <autor>Sabela Varela</autor>
  <apartado numero="1">
       Contido do apartado 1
  </apartado>
  <apartado numero="2">
       Contido do capitulo 2
  </apartado>
</tema>
```

A gramática dun documento XML defínese nun documento DTD. A etiqueta `<!DOCTYPE>` contén a definición da gramática. A súa sintaxe é a seguinte:

```
<!DOCTYPE nomeElementoRaiz [ declaracions ]>
```

Ou ben

```
<!DOCTYPE nomeElementoRaiz SYSTEM "nomeArquivo.dtd">
```

Onde, `nomeElementoRaiz` é o nome do elemento raiz; `declaracions` define unha lista do conxunto de declaracións dos elementos e atributos do documento; e `nomeArquivo.dtd` é o nome do arquivo da DTD.

A continuación vemos a gramática DTD correspondente ao documento XML anterior:

```
<!DOCTYPE tema [
  <!ELEMENT tema (autor, apartado+)>
  <!ATTLIST tema
          titulo CDATA #REQUIRED
          unidade CDATA #REQUIRED>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT apartado (#PCDATA)>
  <!ATTLIST apartado
          numero CDATA #REQUIRED>
]>
```

4. Vincular unha DTD aos datos XML

Os analizadores necesitan coñecer a gramática que define a un documento XML para poder validar a súa construcción. Existen dúas maneiras de vincular os datos DTD aos datos XML: mediante a inserción da DTD dentro do arquivo XML ou mediante a integración dunha referencia a un arquivo DTD distinto.

4.1 Declaracións DTD dentro dun documento XML (DTD internas)

Consiste en definir a gramática no interior do documento XML. Para facer isto engadiremos a declaración `<!DOCTYPE>` xusto despois do prólogo:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Declaracións DTD-->
<!DOCTYPE tema [
  <!ELEMENT tema (autor, apartado+)>
  <!ATTLIST tema
          titulo CDATA #REQUIRED
          unidade CDATA #REQUIRED>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT apartado (#PCDATA)>
  <!ATTLIST apartado
          numero CDATA #REQUIRED>
]>
<!-- Datos XML-->
<tema unidade="5" titulo="A linguaxe XML">
  <autor>Sabela Varela</autor>
  <apartado numero="1">Contido do apartado 1</apartado>
  <apartado numero="2">Contido do capitulo 2</apartado>
</tema>
```

É útil cando queremos transmitir o documento a un destino no que non sabemos se se ten acceso á DTD, polo que o enviamos embebido no propio documento. Neste caso o documento é autosuficiente e o valor do seu atributo `standalone` é `yes`.

4.2 Declaracións DTD nun arquivo .dtd (DTD externas)

Incluír a DTD no mesmo documento é unha boa idea, pero non poderemos empregar o DTD con múltiples documentos. Cando temos moitos documentos XML almacenados, e todos eles deben cumprir coas restriccións dun mesmo DTD, usaremos esta solución onde incluiremos a seguinte instrucción xusto despois do prólogo e antes dos datos XML para indicar o arquivo que contén a definición da gramática DTD:

```
<!DOCTYPE nomeElementoRaiz SYSTEM "nomeArquivo.dtd">
```

Imos ver como quedaría o exemplo anterior separando os datos XML das definicións DTD. Para elo, teremos que dividir o arquivo en dous: un coas definicións DTD e outro cos datos xml e a instrucción de enlace coa DTD.

En primeiro lugar, imos ver o contido do arquivo `tema.dtd`:

```
<!ELEMENT tema (autor, apartado+)>
  <!ATTLIST tema
          titulo CDATA #REQUIRED
          unidade CDATA #REQUIRED>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT apartado (#PCDATA)>
  <!ATTLIST apartado
          numero CDATA #REQUIRED>
```

E a continuación vemos o contido do arquivo `tema.xml` que contén o vínculo co arquivo `.dtd` asociado:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tema SYSTEM "tema.dtd">

<!-- Datos XML-->
<tema unidade="5" titulo="A linguaxe XML">
  <autor>Sabela Varela</autor>
  <apartado numero="1">Contido do apartado 1</apartado>
  <apartado numero="2">Contido do capitulo 2</apartado>
</tema>
```

Con esta solución, o documento XML non é es autosuficiente e, polo tanto, o valor do atributo `standalone` é `no`.

É posible combinar os dous tipos de vínculos. Neste caso, as declaracións realizadas dentro dun documento XML terán prioridade sobre as declaracións externas. E se o atributo `standalone` ten o valor `yes`, ignoraranse as declaracións externas.

4.3 Declaracións DTD nun documento cunha referencia pública

Cando o documento é un estándar usaremos o identificador `PUBLIC`, a cadea de texto que o identifica e a súa URL. Por exemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
```

> No TA1 explícase como validar un documento XML para comprobar que responde a unha gramática definida nunha DTD, empregando distintas ferramentas.

5. Definir elementos

Os elementos son os nodos da árbore dun documento XML. As declaracións de tipo de elemento deben comezar con `<!ELEMENT` seguidas polo *identificador xenérico* do elemento que se declara. A continuación debe incluirse unha *especificación do seu contido*.

Existen dous tipos de elementos: os elementos terminais e os elementos non terminais. Os elementos terminais son as follas da árbore XML.

A sintaxe para definir elementos é a seguinte:

```
<!ELEMENT nomeElemento contido>
```

ou ben

```
<!ELEMENT nomeElemento (tipoDeContido)>
```

As especificacións de contido poden ser as seguintes:

- EMPTY – Úsase para definir aos elementos baleiros. Prohibe a un elemento ter un elemento fillo ou datos textuais (podería ter atributos). Por exemplo:

DTD	XML
`<!ELEMENT br EMPTY>`	` `

- ANY – Permite que o elemento conteña calquera cousa: datos textuais, outros elementos, etc. (non é recomendable o seu uso xa que é aconsellado estruturar adecuadamente os nosos documentos XML).
- PCDATA – Úsase para indicar que o contido do elemento son datos de texto. Exemplo:

DTD	XML
`<!ELEMENT titulo (#PCDATA)>`	`<titulo>A linguaxe XML</titulo>`

- MIXED – Permite que o contido do elemento sexan carácteres textuais ou unha mestura de carácteres e subelementos. Por exemplo, para:

```
<!ELEMENT obxecto (#PCDATA|imaxe)*>
```

Obxecto podería conter cero ou máis ocurrencias de datos de carácter `(#PCDATA)` e/ou subelementos de tipo *imaxe*.

Esta declaración debe respectar as seguintes condicións:

  - Os datos textuais #PCDATA deben aparecer sempre en primeira posición.
  - O grupo debe ser unha elección (separado co carácter |).
  - O grupo debe aparecer cero, unha ou varias veces (operador *).
- ELEMENT - Unicamente pode conter subelementos especificados empregando modelos de contidos.

5.1 Modelos de contidos

Un modelo de contido é un patrón que establece os subelementos aceptados, e a orde na que estes deben estar. A continuación imos ver as distintas posibilidades:

- Fillo único - o elemento secundario debe aparecer unha única vez dentro do elemento que se está a definir. Por exemplo:

DTD

```
<!ELEMENT titor (nome)>
```

XML

```
<titor>
 <nome>Sara Vila Ferreiro</nome>
</titor>
```

- Fillos nunha orde determinada - Os elementos cun ou máis fillos decláranse co nome dos elementos dos fillos entre parénteses e separados por comas. Por exemplo:

DTD

```
<!ELEMENT ciclo (codigo,nome,grao)>
```

XML

```
<ciclo>
 <codigo>CSIFC03</codigo>
 <nome>Desenvolvemento de aplicacións
       Web </nome>
 <grao>Superior</grao>
</ciclo>
```

- Opción a que aparezan uns fillos ou outros – Se en lugar de comas empregamos unha barra vertical estaremos indicando "opción". O número de opcións non está limitado a dúas, e pódese agrupar empregando paréntese.

DTD

```
<!ELEMENT ciclo ((codigo|nome),grao)>
```

Indicaría que un *ciclo* debe conter en primeiro lugar o seu *código* ou o seu *nome* e a continuación debe indicarse o seu *grao*.

Frecuencia

Ademais, cada partícula de contido pode levar un indicador da frecuencia na que este pode aparecer, que se sitúan a continuación do identificador xeral, secuencia ou opción, e non poden ir precedidos por espazos en branco.

- O operador (?) define un compoñente opcional, que pode aparecer ou non (0 ou 1 vez). Por exemplo, se queremos almacenar os números de teléfono obrigando a introducir o número de móbil, pero permitindo que non se indique o fixo, fariamos:

DTD

```
<!-- O teléfono fixo é opcional -->
<!ELEMENT telefono (mobil, fixo?)>
```

XML

```
<telefono>
 <mobil>632323232</mobil>
</telefono>
```

- O operador (+) define un compoñente presente polo menos unha vez (1 ou máis veces).

DTD

```
<!-- O subgrupo (cp, poboacion)
     aparece polo menos unha vez -->
<!ELEMENT provincia (nome,
             (cp,poboacion)+)>
```

XML

```
<provincia>
 <nome>Lugo</nome>
 <cp>27003</cp>
 <poboacion>Lugo</poboacion>
 <cp>27850</cp>
 <poboacion>Viveiro</poboacion>
</provincia>
```

- O operador (*) define un compoñente presente cero, unha ou máis veces (0 ou máis veces).

```
<!—O grupo (ip, nomemaquina) aparece cero, unha ou varias veces. -->
<!ELEMENT maquinas (ip, nomemaquina)*)>
```

DTD

```
<!-- O grupo (ip, maquina)
   aparece 0, 1 ou varias veces -->
<!ELEMENT maquinas (ip, maquina)*)>
```

XML

```
<maquinas>

</maquinas>
```

Na tarefa 1 crearemos unha DTD para validar un documento XML que contén unha mensaxe de correo electrónico.

Na tarefa 2 comprobaremos se os documentos XML que se amosan responden as DTD e faremos as modificacións que sexan oportunas para que estes respondan as regras definidas na DTD.

6. Definir atributos

Ata agora, os exemplos XML que temos feito constaban unicamente de elementos, pero as veces, os atributos permítennos facer cousas que non poderiamos facer empregando so elementos. A principal diferencia entre os elementos e os atributos é que os atributos non poden conter subatributos. Además, os atributos poden empregarse para:

- Definir un valor por defecto.
- Definir un conxunto de valores válidos.
- Definir valores fixos (constantes).
- Crear referencias entre distintos elementos.

Os elementos poden ter cero, un ou varios atributos. Nunha DTD os atributos decláranse empregando a etiqueta `ATTLIST`.

```
<!ATTLIST nomeElemento
    nomeAtributo1 tipo valor
    nomeAtributo2 tipo valor
    ...
>
```

Por exemplo:

```
<!ELEMENT actor (#PCDATA)>
<!ATTLIST actor sexo CDATA #IMPLIED>
```

Estes son datos XML conformes á anterior declaración:

```
<actor sexo="masculino">
   Gael García Bernal
</actor>
```

6.1 Lista de atributos

A etiqueta `ATTLIST` permite declarar varios atributos para un mesmo elemento. Ou tamén se poden declarar varios atributos para o mesmo elemento. Imos ver dous exemplos equivalentes que amosan estes principios:

```
<!-- A etiqueta ATTLIST permite declarar os dous atributos do elemento ->
<!-- O elemento ciclo posúe os atributos codigo e grao ->
<!ATTLIST auto
  codigo CDATA #REQUIRED
  grao CDATA #REQUIRED>
```

podería definirse tamén da seguinte maneira:

```
<!-- A repetición de ATTLIST permite crear os dous atributos para
     o elemento ->
<!ATTLIST ciclo codigo CDATA #REQUIRED>
<!ATTLIST ciclo grao CDATA #REQUIRED >
```

6.2 Os valores dos atributos IMPLIED, REQUIRED e FIXED

Os atributos poden ser opcionais `#IMPLIED` ou obrigatorios `#REQUIRED`. Además, en ocasións poden presentar valores fixos `#FIXED` e os atributos opcionais poden ter valores por defecto.

O seguinte exemplo é unha declaración dun elemento que ten un atributo opcional. Non se ten especificado ningún valor por defecto:

```
<!ATTLIST alumno nacionalidade CDATA #IMPLIED>
```

A continuación vemos como indicar o valor por defecto para un atributo opcional (Cando se especifica un valor por defecto *non* se engade a palabra clave `#IMPLIED`):

```
<!ATTLIST alumno nacionalidade CDATA "española">
```

No seguinte exemplo vemos como declarar un elemento que ten un atributo obrigatorio:

```
<!ATTLIST alumno sexo CDATA #REQUIRED>
```

Se queremos especificar que un atributo debe ter sempre o mesmo valor usaremos a palabra clave `#FIXED`. Por exemplo:

```
<!-- O valor do atributo tipo é sempre pdf ->
<!ATTLIST documento tipo CDATA #FIXED "pdf">
```

6.3 Tipos de atributos

Existen varios tipos de atributos:

6.3.1 Atributos CDATA, NMTOKEN e NMTOKENS

Os atributos `CDATA` (characterdata) son os máis habituais cando queremos declarar un atributo que conten datos de texto.

Os atributos `NMTOKEN` (nametoken) son parecidos, pero unicamente aceptan os carácteres permitidos por XML para nomear cousas (letras, números, puntos, guións, suliñados e os dous puntos).

```
<!ATTLIST nota data NMTOKEN #REQUIRED>
<nota data="6-11-2006">
```

Os atributos de tipo `NMTOKENS` poden conter unha ou varias palabras cos caracteres permitidos por `NMTOKEN` separadas por espazos en branco.

Enumeracións

Cando un atributo unicamente pode tomar valores dunha lista podemos indicar estes entre paréntese e separados polo operador |

```
<!ATTLIST curso nivel (baixo | medio | alto) #IMPLIED>
```

Na tarefa 3 modificaremos a DTD creada na tarefa 1 para engadir atributos.

6.3.2 Atributos de tipo ID, IDREF ou IDREFS

Un atributo pode empregarse como identificador dun elemento declarándoo como `ID`. Este identificador debe ser único no documento e debe comezar por unha letra ou polo carácter de subliñado. Por exemplo:

```
<!ATTLIST profesor id ID #REQUIRED>
```

Os atributos `IDREF` son referencias aos identificadores (atributos definidos como `ID`). O valor do atributo `IDREF` debe corresponder a un identificador de elemento existente.

```
<!--O atributo titor apunta ao identificador doutro elemento ->
<!ATTLIST alumno titor IDREF #IMPLIED>
```

A palabra clave `IDREFS` permite especificar varias referencias a identificadores dentro da declaración dun atributo. As distintas referencias irán separadas por un espazo en branco. A continuación vemos a DTD completa que especifica a gramática dun libro de cociña en XML:

```
<!ELEMENT libro (receita | ingrediente)*>
<!ELEMENT receita #PCDATA>
<!ATTLIST receita id ID #REQUIRED>
<!ELEMENT ingrediente #PCDATA>
<!ATTLIST ingrediente ref IDREFS #IMPLIED>
```

Empregaremos o atributo ref para saber que receitas levan ese ingrediente. A continuación vemos un documento XML válido segundo esta DTD:

```
<libro>
   <receita id="rec_1">Filloas</receita>
   <receita id="rec_2"> Flan de queixo </receita>
   <receita id="rec_3"> Torta de mazá </receita>
   <ingrediente ref="rec_1 rec_3">ovos</ingrediente>
   <ingrediente ref="rec_2">Queixo</ingrediente>
   <ingrediente ref="rec_1 rec_2">Leite</ingrediente>
</libro>
```

6.3.3 Atributos de tipo ENTITY

`ENTITY` é a palabra clave que designa as entidades. No seguinte punto imos ver en detalle as entidades. As entidades permiten definir as constantes para o documento e por iso os atributos poden ser de tipo entidad. Tamén poden facer referencia a varias entidades (separadas por un espazo en branco) empregando a palabra clave `ENTITIES`:

```
<!--El atributo a apunta a una única entidad ->
<!ATTLIST A a ENTITY #IMPLIED>
<!--El atributo b apunta a varias entidades ->
<!ATTLIST A b ENTITIES #IMPLIED>
```

6.3.4 Atributos de tipo NOTATION

Este tipo de atributo permite ao autor declarar que o seu valor se axusta a unha notación previamente declarada.

```
<!ATTLIST foto formato NOTATION (BMP | JPG) #IMPLIED>
```

Neste exemplo declaramos unha lista de notacións como tipo do atributo formato, o que permitirá que o atributo tome un deste valores. Para que esto sexa válido haberá que declarar as notacións da lista. Para declarar as notacións úsase `<!NOTATION`.

No apartado de definición de notacións vemos un ejemplo do seu uso.

6.4 Exemplo de DTD con atributos

A continuación vemos un exemplo dun documento XML coa DTD interna:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE cursos [
   <!ELEMENT cursos (nomeEmpresa, curso*,alumnos*)>
   <!ELEMENT nomeEmpresa (#PCDATA)>
   <!ELEMENT alumnos (alumno+)>
  <!ELEMENT curso (#PCDATA)>
   <!ATTLIST curso
       codigo ID #REQUIRED
       nome CDATA #REQUIRED
       dataInicio CDATA #REQUIRED
       nivel (baixo | medio | avanzado) #IMPLIED>
   <!ELEMENT alumno (nome, apelidos, telefono?)>
   <!ATTLIST alumno cursos IDREFS #IMPLIED>
   <!ELEMENT nome (#PCDATA)>
   <!ELEMENT apelidos (#PCDATA)>
   <!ELEMENT telefono (#PCDATA)>
]>

<cursos>
   <nomeEmpresa>Xunta de Galicia</nomeEmpresa>
   <curso codigo="C89" nome="Java" dataInicio="7/10-2009" nivel="avanzado" />
   <curso codigo="C90" nome="PHP" dataInicio="6/11-2009" />
   <curso codigo="C100" nome="XML" dataInicio="30/03/2010" nivel="medio" />
  <alumnos>
     <alumno cursos="C90">
       <nome>Pilar</nome>
       <apelidos>Pérez Sousa</apelidos>
     </alumno>
```

```
    <alumno>
      <nome>Carmen</nome>
      <apelidos>Novoa Real</apelidos>
    </alumno>

    <alumno cursos="C89 C90 C100">
       <nome>Santiago</nome>
      <apelidos>Souto Lema</apelidos>
      <telefono>698811111</telefono>
    </alumno>

    <alumno cursos="C100 C89">
       <nome>Antón</nome>
      <apelidos>Rioboo Vila</apelidos>
      <telefono>698811111</telefono>
    </alumno>
  </alumnos>
</cursos>
```

Na tarefa 4 comprobaremos se os documentos XML con atributos que se amosan responden as DTD e faremos as modificacións que sexan oportunas para que estes respondan as regras definidas na DTD.

Na tarefa 5 resolveremos os exercizos propostos de construción de documentos XML e DTD con elementos e atributos.

7. As entidades

As entidades, nun documento XML, poden ser usadas como constantes dentro do documento XML, ou poden facer referencias a obxectos externos (imaxes, ficheiros, páxinas web, etc.).

As entidades permiten facer referencia a outros contidos que se incrustarán no documento no momento de ser procesados, e que non deben ser analizados sintacticamente segundo as regras de XML.

Baixo o nome de entidades encóntranse tanto as entidades definibles como as predefinidas.

As entidades definibles decláranse como sigue:

```
<!ENTITY nomeEntidad definicionEntidad>
```

Por exemplo:

```
<!ENTITY cidade "Santiago de Compostela">
```

Para usalas no documento teremos que poñer: `&nomeEntidad;` (no exemplo: `&cidade;`).

XML ten algunhas entidades internas *predefinidas* para permitir inserir caracteres que teñen un significado especial en XML, como por exemplo o símbolo <. As principais entidades internas predefinidas son:

ENTIDADE	CARACTER
&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	
&#codChar;	Substitúese un caracter empregando o seu código hexadecimal. Por exemplo: ©

As veces, o uso das entidades pode ser moi incómodo, e complicar a lectura do documento. Por isto, en ocasións pode ser moi útil empregar unha sección `<![CDATA[ ... ]]>`, xa que o parser XML ignorará o seu contido. Por exemplo, o XML seguinte:

```
<códigoFonte>
       &lt;H1&gt;O operador de concatenación: &amp;&lt;/H1&gt;
</códigoFonte>
```

poderíase substituír por:

```
< códigoFonte >
       <![CDATA[<H1>O operador de concatenación: &</H1>]]>
</ códigoFonte >
```

As entidades pódense clasificar nos seguintes grupos non excluíntes:

- Internas e/ou externas
  - As internas referéncianse dentro do documento no que foron declaradas.
  - As externas fan referencia a un arquivo externo.
- Analizables (ou de texto) e non analizables
  - As analizables fan referencia a documentos de texto.
  - As non analizables fan referencia a documentos doutro tipo: imaxes, video, etc.

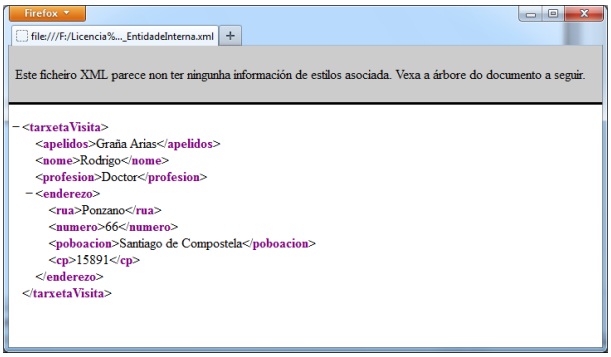
7.1 Entidades internas

Tamén coñecidas como *macros* ou *constantes* de texto, as entidades internas son as que se asocian a unha cadea de carácteres. Referéncianse única e exclusivamente dentro do documento no que foron declaradas.

Imos ver un exemplo de uso dunha entidade interna nun documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE tarxetaVisita [
    <!ENTITY cidade "Santiago de Compostela">
  ]>
<tarxetaVisita>
  <apelidos>Graña Arias</apelidos>
  <nome>Rodrigo</nome>
  <profesion>Doctor</profesion>
  <enderezo>
    <rua>Ponzano</rua>
    <numero>66</numero>
    <poboacion>&cidade;</poboacion>
    <cp>15891</cp>
  </enderezo>
</tarxetaVisita>
```

No navegador veremos que se ten substituído a referencia á entidade interna, polo seu contido:



7.2 Entidades externas

As entidades externas permiten vincular un documento XML a outro documento a través da súa URL. Este documento pode ser de tipo XML ou de calquera outro tipo.

As entidades externas poden ser analizadas ou non. Unha entidade externa analizada debe conter texto e marcado XML válidos, e o seu obxectivo e permitir compartir texto entre varios documentos.

As entidades non analizadas permiten incluir contido que non é XML (como por exemplo imaxes), proporcionando un mecanismo para asociar os datos coa ferramenta adecuada, para o que se usa a palabra `NDATA`. Para asociar a entidade coa ferramenta apropiada teremos que facer uso das notacións (`NOTATION`). No apartado seguinte vemos un exemplo disto.

Unha entidade externa pode declararse empregando a palabra reservada `SYSTEM` para identificar un arquivo do sistema local ou duna rede. Por exemplo:

```
<!ENTITY doc SYSTEM "http:/localhost/docsxml/outrodoc.xml”>
```

As referencias á entidade `doc` no documento XML serán reemprazadas automaticamente polo contido do documento ao que fai referencia.

Podemos facer que a entidade faga referencia a un arquivo binario (por exemplo, unha imaxe), co cal a entidade non se substitúe polo contido do arquivo, engadindo a palabra clave NDATA e indicando o tipo de arquivo. Por exemplo (Vemos o exemplo completo no apartado de notacións:

```
<!ENTITY imaxe SYSTEM "imaxe.gif" NDATA gif>
```

Por exemplo, se temos definida a entidade *capitulo1* cunha referencia ao documento capitulo1.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE entidadelibro [
   <!ENTITY capitulo1 SYSTEM "capitulo1.xml">
]>
<entidadelibro>
    <tema>Exemplo de entidade externa</tema>
    &capitulo1;
</entidadelibro>
```

E o contido do arquivo capitulo1.xml é:

```
<?xml version="1.0" encoding="UTF-8"?>
<capitulo>
    <para>Este é o primeiro capitulo</para>
</capitulo>
```

Esto sería equivalente a :

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE entidadelibro [
   <!ENTITY capitulo1 SYSTEM "capitulo1.xml">
]>
<entidadelibro>
    <tema>Exemplo de entidade externa</tema>
    <capitulo>
        <para>Este é o primeiro capitulo</para>
     </capitulo>
</entidadelibro>
```

A palabra clave `SYSTEM` indica que o recurso é privado. Se a substituímos pola palabra reservada `PUBLIC` estaremos indicando que o recurso está nun URI (Unique Resource IDentifier), accesible para calquera.

7.3 Entidades de parámetro

Están deseñadas para conter listas de atributos e modelos de contido o que nos permiten modularizar a DTD. Por exemplo, poderiamos crear unha entidade que almacene unha lista de subelementos que se comparten entre varios elementos. A continuación, fariamos referencia a ela tantas veces como sexa necesario. Deste xeito, se temos que modificar estes elementos, so o teriamos que facer nun único lugar.

As entidades de parámetros teñen un identificador especial na súa declaración: o símbolo de tanto por cento `%`, que irá precedendo ao nome. Imos ver un exemplo de como cambiaría a declaración dos seguintes atributos sen usar parámetros no seguinte documento XML:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE envios SYSTEM "prueba.dtd">

<envios>
  <orixen rua="Gran vía" numero="25" poboacion="Marín" />
  <destino rua="Gran vía" numero="25" poboacion="Marín" />
</envios>
```

A DTD sen utilizar entidades de parámetro sería:

```
<!ELEMENT envios (orixen,destino)>
<!ELEMENT orixen EMPTY>
<!ELEMENT destino EMPTY>
<!ATTLIST orixen
        rua CDATA #REQUIRED
        numero CDATA #IMPLIED
        poboacion CDATA #REQUIRED >
<!ATTLIST destino
        rua CDATA #REQUIRED
        numero CDATA #IMPLIED
        poboacion CDATA #REQUIRED >
```

E utizando unha entidade de parámetros para os atributos comunes aos elementos orixen e destino:

```
<!ELEMENT envios (orixen,destino)>
<!ENTITY % enderezo
    "rua CDATA #REQUIRED
     numero CDATA #IMPLIED
     poboacion CDATA #REQUIRED">
```

```
<!ELEMENT orixen EMPTY>
<!ELEMENT destino EMPTY>
<!ATTLIST orixen %enderezo;>
<!ATTLIST destino %enderezo;>
```

8. Notacións

As **declaracións de notación** proveen un nome para unha notación que despois podemos usar cando declaramos atributos. A continuación vemos un exemplo:

```
<?xml version= "1.0" encoding= "ISO-8859-1" standalone= "yes"?>
<!DOCTYPE horario [
<!ELEMENT horario (#PCDATA)>
<!NOTATION L SYSTEM "Luns">
<!NOTATION M SYSTEM "Martes">
<!NOTATION R SYSTEM "Mércores">
<!NOTATION X SYSTEM "Xoves">
<!NOTATION V SYSTEM "Venres">
<!ATTLIST horario
 dia NOTATION (L | M | R | X | V) #REQUIRED
 hora CDATA #REQUIRED>
]>
<horario dia="L" hora="8:30">Bases de datos</horario>
```

Pero además, as notacións proporcionan a información necesaria para procesar as entidades externas non analizadas. Esto é, permiten asociar unha aplicación a un formato de entidade ignorado polo procesador XML. A notación debe ser declarada antes de ser empregada por unha entidade. Por exemplo, a seguinte notación indica que o programa a usar para visulizar un arquivo de imaxe GIF é o "editorGIF.exe":

```
<!NOTATION GIF SYSTEM "editorGIF.exe">
```

Para asociar unha entidade externa non analizada a esta notación basta declarar dita entidade do seguinte modo:

```
<!ENTITY dibujo SYSTEM "imagen.gif" NDATA gif>
```

A continuación vemos un exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumno [
 <!ELEMENT alumno (nome, foto?)>
    <!NOTATION gif SYSTEM "iexplore.exe">
    <!ENTITY foto1 SYSTEM "fotoAna.gif" NDATA gif>
    <!ELEMENT nome (#PCDATA)>
    <!ELEMENT foto EMPTY>
    <!ATTLIST foto
          imaxe ENTITY #REQUIRED>
]>

<alumno>
    <nome>Ana Ferreiro</nome>
    <foto imaxe="foto1"/>
</alumno>
```

No exemplo, `gif` é a notación encargada de manexar a entidade `foto1`. Na práctica non se usa o procedemento anterior, xa que as aplicacións, por sí sólas, saben ou non saben manexar os datos XML. Se non saben facelo, esta declaración é demasiado simple para que o analiza-

dor invoque á aplicación capaz de manexar a entidade. O normal, sería facer a seguinte declaración e aplicación de esta á instancia, delegando na aplicación ou analizador xml a capacidade de manexo dos datos que non son XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumno [
 <!ELEMENT alumno (nome, foto?)>
    <!ELEMENT nome (#PCDATA)>
 <!ELEMENT foto EMPTY>
    <!ATTLIST foto
          imaxe CDATA #REQUIRED>
]>

<alumno>
 <nome>Ana Ferreiro</nome>
 <foto imaxe="foto_Ana.gif"/>
</alumno>
```

9. Seccións condicionais

En ocasións pode resultar útil ocultar algunhas partes da declaración da DTD. As palabras clave `INCLUDE` e `IGNORE` permiten, respectivamente, incluír ou ignorar seccións de declaracións dentro dunha DTD. Na seguinte táboa vemos como sería a sintaxe e un exemplo do seu uso:

Palabra clave	Sintaxe	Exemplo
INCLUDE	<![INCLUDE [declaracións visibles]]>	<!INCLUDE[<!ELEMENT pseudo #PCDATA]]>
IGNORE	<![IGNORE[declaracións a ocultar]>	<!IGNORE[<!ELEMENT contrasinal #PCDATA]]>

Cando poñemos `INCLUDE`, o contido da declaración é visible na DTD. Mentres que coa palabra clave `IGNORE` podemos ocultar e ignorar algúns elementos da DTD. El contido ignorado ou incluído será o que aparece entre corchetes xusto despois das palabras clave `IGNORE` ou `INCLUDE`.

Estas seccións condicionais só están permitidas en DTD externas (non nas internas), e deben estar compostas por seccións de marcado completas (non son válidos os fragmentos).

Unha boa práctica é usar unha entidade de parámetros como base dunha sección condicional como a seguinte:

```
<!ENTITY % DEBUG "INCLUDE">
```

Esto permite empregar a entidade de parámetros `DEBUG` para encerrar bloques de documento. Así, unicamente modificando a entidade de parámetros da seguinte maneira:

```
<!ENTITY % DEBUG "IGNORE">
```

estes bloques de documento quedarían desactivados.

A continuación vemos un exemplo do seu uso:

```
<!ENTITY % corto "INCLUDE">
<!ENTITY % longo "IGNORE">
<![%longo;
     [
      <!ELEMENT axenda (nome, mobil+, fixo?, fax?, correo*)>
      <!ELEMENT nome  (#PCDATA)>
      <!ELEMENT mobil  (#PCDATA)>
```

```
      <!ELEMENT fixo  (#PCDATA)>
      <!ELEMENT fax  (#PCDATA)>
      <!ELEMENT correo  (#PCDATA)>
]]>
<![%corto;
     [
      <!ELEMENT axenda (nome, mobil, fixo?)>
      <!ELEMENT nome  (#PCDATA)>
      <!ELEMENT mobil  (#PCDATA)>
      <!ELEMENT fixo  (#PCDATA)>
]]>
```

Este DTD validaría o seguinte XML, tanto se o modificamos para incluir o formato longo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE axenda SYSTEM "Exemplo_condicional.dtd">
<axenda>
  <nome>Bieito Carballo</nome>
  <mobil>699999999</mobil>
</axenda>
```

Pero unicamente validaría o seguinte XML se temos incluído o formato longo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE axenda SYSTEM "Exemplo_condicional.dtd">
<axenda>
  <nome>Bieito Carballo</nome>
  <mobil>699999999</mobil>
  <correo>bieito@meucorreo.com</correo>
</axenda>
```

Na tarefa 6 resolveremos os exercizos propostos de construción de documentos XML e DTD incluíndo entidades.