# Material didáctico para exposición teórica

| | | |
|---|---|---|
| Familia profesional | IFC | Informática e comunicacións |
| Ciclo formativo | CSIFC03<br>CSIFC02<br>CSIFC01 | Desenvolvemento de aplicacións web<br>Desenvolvemento de aplicacións multiplataforma<br>Administración de sistemas informáticos en rede |
| Grao | | Superior |
| Módulo profesional | MP0373 | Linguaxes de marcas e sistemas de xestión de información |
| **Unidade didáctica** | **UD03** | **Validación de documentos** |
| **Actividade** | **A01** | **DTD** |
| Profesoras | | María del Carmen Fernández Lameiro<br>María Elena Goy López |

# Índice

# 1. Validación de documentos XML

A gramática, vocabulario e estrutura dun documento XML, pódese describir utilizando unha linguaxe específica, para que un ou varios usuarios ou procesos poidan crear documentos XML e certificar que cumpren esa estrutura. Existen varios métodos diferentes de facer estas descrición: DTD, Schemas, RELAX NG ou Schematron; os máis utilizados son os esquemas e os DTD, aínda que se poden realizar descricións máis complexas e detalladas nos esquemas.

Un documento XML que respecta a gramática especificada por W3C é un documento ben formado. Un documento XML ben formado que respecta a gramática descrita nunha DTD (*Document Type Definition*) ou nun esquema é un documento *válido*. Ao proceso de comparar un documento XML cun destes documentos de regras chámaselle validación.

A descrición do vocabulario, gramática e estrutura dun XML contén, por exemplo:

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

Por exemplo, supoñamos que se queren crear apuntes dun módulo didáctico entre varios profesores e en formato XML. O módulo constará de varios temas e cada tema ten que ter a mesma estrutura:

- 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 parágrafos de texto. Ademais poden conter parágrafos con código e imaxes.

Para asegurase de forma automática que cada tema cumpre os requisitos anteriores, debería de crear un DTD ou un Schema que permita validar cada tema, e validar cada tema utilizando unha ferramenta de validación e o DTD ou o Schema.

# 2. DTD

## 2.1 Vincular unha DTD aos datos XML

Existen dúas maneiras de vincular a información DTD á información XML: mediante a inserción da DTD dentro do arquivo XML ou mediante a integración dunha referencia a un arquivo DTD. Utilízase a etiqueta *DOCTYPE*.

### 2.1.1 DTD interna

Consiste en definir a gramática no interior do documento XML. Para facer isto engadirase xusto despois do prólogo:

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

Onde, *nomeElementoRaiz* é o nome do elemento raiz e *declaracions* é o conxunto de declaracións dos elementos e atributos do documento.

As DTD internas son útiles cando non se sabe se é posible o acceso a un DTD externo, polo que se coloca embebido no propio documento. Neste caso o documento XML é autosuficiente e o valor do seu atributo *standalone* é *yes*.

Exemplo:

```
<?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>
```

### 2.1.2 DTD externa

A principal avantaxe da utilización das DTD externas é a utilización do mesmo DTD para a validación de varios documentos XML. Na validación participan dous arquivos: o arquivo XML e o arquivo DTD e ten que existir unha instrución xusto despois do prólogo e antes dos datos XML para indicar o arquivo que contén a definición da gramática DTD coa sintaxe seguinte:

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

Onde, *nomeElementoRaiz* é o nome do elemento raiz; e *nomeArquivo.dtd* é o nome do arquivo externo coas declaracións dos elementos e atributos.

Utilizando o mesmo exemplo que no apartado anterior, o *arquivo exemplo212_tema.dtd* podería ser:

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

E o arquivo XML podería ser:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tema SYSTEM "exemplo212_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.

### 2.1.3 DTD 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">
```

## 2.2 Definir elementos

Os elementos son os nodos da árbore dun documento XML. As declaracións de tipo de elemento deben comezar con `<!ELEMENT` seguidas polo nome do elemento que se declara. A continuación debe incluírse 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 especificacións-de-contido>
```

### 2.2.1 Especificacións de contido

Poden ser:

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

| DTD | XML |
|---|---|
| `<!ELEMENT br EMPTY>` | `<br />` |

- ANY – Permite que o elemento conteña calquera cousa: datos textuais, outros elementos, etc. Non é recomendable o seu uso.
- PCDATA – Úsase para indicar que o contido do elemento son datos de texto. Deberá ir precedido do símbolo # e entre parénteses. Por 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:

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

*Obxecto* podería conter cero ou máis ocorrencias 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 tal e como se ve no apartado seguinte.

### 2.2.2 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)>
```

XML

```
<ciclo>
  <codigo>CSIFC03</codigo>
  <grao>Superior</grao>
</ciclo>
```

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

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

### 2.2.3 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).

DTD

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

XML

```
<maquinas>
</maquinas
```

Tarefa 1. Crear e modificar DTD empregando só declaración de elementos. Modificar XML para que poida ser validado con DTD.

## 2.3 Definir atributos

Ata agora, os exemplos XML que se viron constaban unicamente de elementos, pero ás veces, os atributos permítennos facer cousas que non poderiamos facer empregando so elementos, como:

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

Os atributos decláranse empregando a etiqueta *ATTLIST*. Se hai que declarar máis dun atributo pode facerse cunha soa etiqueta *ATTLIST* ou con varias.

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

Exemplo cun só atributo:

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

Estes son datos XML conformes á anterior declaración:

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

Exemplo con varios atributos:

```
<!—- O elemento ciclo posúe os atributos codigo e grao ->
<!ATTLIST ciclo
  codigo CDATA #REQUIRED
  grao CDATA #REQUIRED>
```

Exemplo equivalente ao anterior:

```
<!—- ATTLIST repetido para crear os dous atributos do elemento ->
<!ATTLIST ciclo codigo CDATA #REQUIRED>
<!ATTLIST ciclo grao CDATA #REQUIRED >
```

### 2.3.1 IMPLIED, REQUIRED e FIXED

Os atributos poden ser opcionais *#IMPLIED* ou obrigatorios *#REQUIRED*. Ademais, 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 vese 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 vese 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">
```

### 2.3.2 Tipos de atributos

Detállanse a continuación os tipos de atributos máis habituais.

#### CDATA, NMTOKEN e NMTOKENS

Os atributos *CDATA* (*characterdata*) son os máis habituais cando queremos declarar un atributo que conten 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). Os atributos de tipo *NMTOKENS* poden conter unha ou varias palabras cos carácteres permitidos por *NMTOKEN* separadas por espazos en branco.

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

#### Enumeracións

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

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

#### ID, IDREF ou IDREFS

Un atributo pode empregarse como identificador dun elemento declarándoo como *ID*. Este identificador debe ser único no documento, debe comezar por unha letra ou polo carácter de subliñado e non pode ser *#FIXED*.

```
<!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 vese a DTD completa que especifica a gramática dun libro de cociña en XML:

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

Empregarase o atributo *ref* para saber as receitas que levan ese ingrediente. A continuación móstrase 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>
```

### 2.3.3 Exemplo de documento XML con DTD interna que describe atributos

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE cursos [
<!ELEMENT cursos (nomeEmpresa, curso*,alumnos)>
<!ELEMENT nomeEmpresa (#PCDATA)>
<!ELEMENT alumnos (alumno+)>
<!ELEMENT curso EMPTY>
<!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>
```

Tarefa2. Crear e modificar DTD empregando declaracións de elementos e de atributos. Modificar XML para que poida ser validado con DTD.

## 2.4 Notacións

O elemento *NOTATION* permite definir un nome que despois se pode usar ao declarar atributos.

Declaración da notación:

```
  <!NOTATION nome SYSTEM "cadea">
```

A notación pode usarse na declaración dun atributo indicando *NOTATION* despois do nome do atributo, seguido dunha lista de posibles notacións entre parénteses.

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>
```

## 2.5 Entidades

### 2.5.1 Entidades utilizadas no DTD

As entidades de parámetro están pensadas para conter listas de atributos e modelos de contido o que nos permiten modularizar a DTD. Por exemplo, poderíase definir unha entidade paramétrica que almacene unha lista de subelementos que se comparten entre varios elementos, e utilizala tantas veces como sexa necesario. Deste xeito, se hai que modificar estes elementos, só se tería 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.

Por exemplo, supoñamos o seguinte XML, no que dous elementos de diferente nome teñen a mesma descrición.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE envios SYSTEM "exemplo251.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 utilizando unha entidade de parámetros para os atributos comúns aos elementos orixe 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;>
```

## 2.5.2 Entidades utilizadas no XML

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. As entidades poden ser:

- Predefinidas.
- Definibles
  - Internas. Referéncianse dentro do documento no que foron declaradas.
  - Externas. Fan referencia a un arquivo externo.
    - As analizables fan referencia a documentos de texto.
    - As non analizables fan referencia a documentos doutro tipo como imaxes ou vídeos.

Os navegadores procesan e visualizan as entidades predefinidas internas pero non o resto das entidades.

### Entidades predefinidas

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

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

As entidades pódense clasificar nos seguintes grupos que non se exclúen entre si:

### Entidades internas

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

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

Para usalas no documento XML hai que poñer:

```
&nomeEntidad;
```

Exemplo de definición:

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

Exemplo de uso:

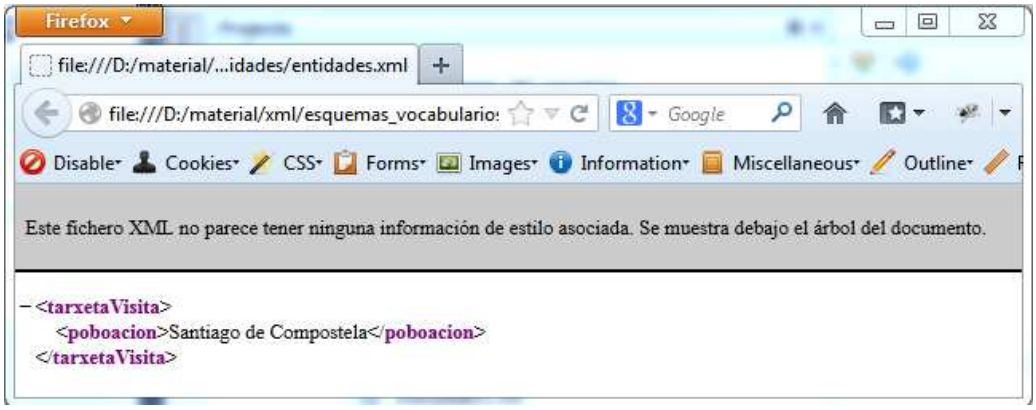
```
&cidade;
```

Exemplo completo:

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

```
<!DOCTYPE tarxetaVisita [
<!ELEMENT tarxetaVisita (poboacion)>
<!ELEMENT poboacion (#PCDATA)>
<!ENTITY cidade "Santiago de Compostela">
]>
<tarxetaVisita>
  <poboacion>&cidade;</poboacion>
</tarxetaVisita>
```

No navegador verase que se substitúe a referencia á entidade interna, polo seu contido:




## Entidades externas

Unha entidade externa defínense no DTD e utilízanse no XML de forma parecida ás entidades internas, colocando a palabra reservada *SYSTEM* despois o nome da entidade para identificar un recurso privado (arquivo do sistema local ou duna rede) ou a palabra *PUBLIC* para indicar que o recurso está nun URI (Unique Resource IDentifier), accesible para calquera. Por exemplo:

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

As referencias á entidade externa no documento XML serán substituídas automaticamente polo contido do documento ao que fai referencia.

### Entidades externas analizadas

As entidades externas analizadas permiten vincular un documento XML a outro documento que debe conter texto e marcado XML válido, a través da súa URL. O seu obxectivo é permitir compartir texto entre varios documentos.

No seguinte exemplo, as entidades externas analizadas son documentos de texto sen etiquetas XML. Ao procesar o arquivo XML, aparecerá o contido de *texto1.txt* e de *texto2.txt* dentro das etiquetas *tema*:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE temas [
<!ELEMENT temas (tema)+>
<!ELEMENT tema (#PCDATA)>
<!ENTITY apartado1 SYSTEM "texto1.txt">
<!ENTITY apartado2 SYSTEM "texto2.txt">
]>
<temas>
  <tema>&apartado1;</tema>
  <tema>&apartado2;</tema>
</temas>
```

No seguinte exemplo, as entidades externas analizadas son documentos de texto con etiquetas XML. O DTD deberá de conter a descrición das etiquetas dos documentos XML externos. Ao procesar o arquivo XML principal, aparecerá o contido de *apartado1.xml* e de *apartado2.xml* dentro das etiquetas *tema*:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- documento xml principal -->
<!DOCTYPE temas [
<!ELEMENT temas (tema)+>
<!ELEMENT tema (titulo,contido)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT contido (#PCDATA)>
<!ENTITY apartado1 SYSTEM "apartado1.xml">
<!ENTITY apartado2 SYSTEM "apartado2.xml">
]>
<temas>
  <tema>&apartado1;</tema>
  <tema>&apartado2;</tema>
</temas>
```

O arquivo apartado1.xml pode ter un contido como o do seguinte:

```
<!-- documento apartado1.xml parecido a apartado2.xml -->
<titulo>Apartado 1</titulo>
<contido>Contido do apartado 1</contido>
```

### Entidades externas non analizadas

As entidades externas non analizada permiten incluír contido que non é XML, como por exemplo imaxes de forma similar á das entidades externas non analizadas, e ademais utilízase NDATA xunto con NOTATION para asociar o contido externo coa ferramenta adecuada.

Un exemplo básico que permite asociar a aplicación iexplore.exe para visualizar o arquivo fotoAna.gif, pode ser o seguinte:

```
<?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>
```

Na práctica non se usa o procedemento anterior, senón que se deixa ao analizador que resolva cal é a aplicación que pode manexar o arquivo externo que non é 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>
```

Tarefa3. Crear  DTD con entidades.

## 2.6 Seccións condicionais

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

| Sintaxe | Exemplo |
|---|---|
| `<![INCLUDE[declaracións visibles]]>` | `<!INCLUDE[<!ELEMENT pseudo #PCDATA]]>` |
| `<![IGNORE[declaracións a ocultar]]>` | `<!IGNORE[<!ELEMENT contrasinal #PCDATA]]>` |

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 cando se necesita ter a posibilidade de permutar entre dúas descricións diferentes, pode ser usar unha entidade de parámetros como base dunha sección condicional como a seguinte:

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

Isto permite empregar a entidade de parámetros *DEBUG* para encerrar bloques de documento. Así, estes bloques de documento quedarían desactivados  unicamente modificando a entidade de parámetros da seguinte maneira:

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

Por exemplo este DTD:

```
<!ENTITY % curto "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)>
]]>
<![%curto;
[
<!ELEMENT axenda (nome, mobil, fixo?)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT mobil (#PCDATA)>
<!ELEMENT fixo (#PCDATA)>
]]>
```

Validaría:

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

E non validaría:

```
<?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>
```

Os dous XML anteriores quedarían validados se no documento DTD se intercambian as entidades para incluír o formato longo:

```
<!ENTITY % curto "IGNORE">
<!ENTITY % longo "INCLUDE">
```

Tarefa4. Crear  DTD con seccións condicionais.

# 3. Documentos de apoio ou referencia


- PIN RODRIGUEZ, Margarita, LOURIDO ESTÉVEZ, Víctor M. *Unidade didáctica 4 (Validación de documentos XML), actividade 1(A necesidade de crear documentos XML válidos.* Xunta de Galicia, Consellería de Cultura, Educación e Ordenación Universitaria. 2013.