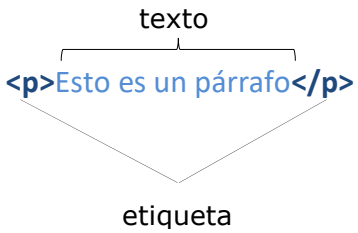
Introducción a los Lenguajes de Marcas

1 CARACTERÍSTICAS DE LOS LENGUAJES DE MARCAS

Un lenguaje de marcas es aquel que se utiliza para codificar en un documento información adicional (metainformación) mediante la incorporación en el mismo de marcas, también llamadas etiquetas o anotaciones, que permitan hacer explícita la estructura de un documento, su contenido semántico o cualquier otra información lingüística o extralingüística que se quiera hacer patente.

Un lenguaje de marcas define un conjunto de reglas que describen cómo deben incluirse las marcas, en qué condiciones se permiten y su significado.

Por ejemplo, el lenguaje HTML (*Hiper Text Markup Language*) utiliza la etiqueta siguiente para indicarle al navegador web que un determinado texto se tiene que formatear como un párrafo:



Otro uso típico de los lenguajes de marcas es la estructuración de datos para facilitar su tratamiento automatizado por parte de un programa informático.

Ejemplo de un posible formato de datos estructurado para describir un libro:

```
<libro>
      <titulo>El Aleph</titulo>
      <autor>Jorge Luis Borges</autor>
      <editor>Warner</editor>
      <precio>6.95€</precio>
      <tema>Ficción</tema>
      <isbn>0446576636</isbn>
</libro>
```

Entre las características más importantes de los lenguajes de marcas están las siguientes:

- **Se codifican mediante ficheros de texto plano, no binarios**. Por tanto, se pueden codificar los documentos con un simple editor de texto.
- **Facilitan la interoperabilidad**. El uso de ficheros de texto plano posibilita su intercambio entre programas distintos e incluso entre sistemas operativos diferentes.
- **Emplean etiquetas o marcas** que permiten marcar el texto que representa el contenido de un documento incrustándolas en el propio documento. Una etiqueta tiene la siguiente forma:

<nombre_etiqueta>contenido</nombre_etiqueta>

Existe un error común a la hora de catalogar los lenguajes de marcas como lenguajes de programación. Aunque que se pueden usar conjuntamente con lenguajes de programación, los lenguajes de marcado, en general, no poseen las características propias de éstos, como variables, operadores, sentencias condicionales, bucles, etc.

2 ÁMBITOS DE APLICACIÓN

Los lenguajes de etiquetado tienen una amplia aplicación en multitud de ámbitos. A continuación, se citan algunos de ellos:

- **Marcado de documentos de tipo general**. Por ejemplo, **Docbook** es un lenguaje de marcas que permite el etiquetado de documentos para todo tipo de material y programas informáticos. Otro ejemplo es el lenguaje de marcado para wikis denominado **Wikitexto**, del que, desgraciadamente, no existe un estándar que defina su sintaxis.

- **Tecnologías de Internet**. Sin duda, es uno de los ámbitos de aplicación donde más se usan los lenguajes de marcas. De hecho, la evolución de la World Wide Web está directamente ligada a lenguajes de marcas como **SGML** (*Standard Generalized Markup Language*), **HTML**, **XML**, etc.

- **Lenguajes de marcación especializados**. Se aplican a ámbitos específicos como las matemáticas (**OpenMath**), modelado de realidades virtuales (**VRML**) o los videojuegos (**BulletML**), entre otros.

Uno de los ámbitos de aplicación donde los lenguajes de marcas tienen un papel muy relevante (en especial el lenguaje XML) es en las tecnologías web y en especial en la **web semántica**. En este sentido, la evolución de los lenguajes de marcas determinó la World Wide Web, tal y como la conocemos hoy en día.

3 CLASIFICACIÓN

Los lenguajes de marcas se agrupan en tres categorías no excluyentes. De hecho, existen lenguajes como HTML que podrían formar parte de más de una de ellas:

- **Lenguajes de marcado de presentación**. El marcado de presentación se utiliza para indicar el formato del texto y normalmente el usuario no es consciente del marcado, ya que se emplean programas de alto nivel para realizarlo. Aunque este tipo de marcado es útil para maquetar la presentación de un documento para su lectura, resulta insuficiente para el procesamiento automático de la información. El marcado de presentación resulta complicado de mantener o modificar, por lo que su uso se ha ido reduciendo en proyectos grandes en favor de otros tipos de marcado más estructurados.

- **Lenguajes de marcado procedimental**. El marcado procedimental también está enfocado hacia la presentación del documento, y, sin embargo, es visible para el usuario que edita el texto. Se ha usado extensivamente en aplicaciones de edición profesional, manipulado por tipógrafos calificados, ya que puede llegar a ser extremadamente complejo. Algunos ejemplos de marcado de procedimientos son **ntroff, troff, TeX**.

- **Lenguajes de marcado descriptivos**. En esta categoría se encuentran **SGML** y **XML**. Ambos son estándares del W3C, que es el organismo encargado de la estandarización de las tecnologías sobre las que se sustenta la Web. Estos lenguajes no indican que hacer con el contenido del documento, sino que indican qué es dicho contenido. Como su propio nombre indica, lo describen. La idea clave es que las marcas no determinan el procesamiento del documento de manera fija (como en los procedimentales), ya que dicho procesamiento se determina a partir de las necesidades concretas, y se beneficia de la estructura lógica del documento caracterizada a través de sus marcas.

Conviene mencionar los principales lenguajes de marcas en el campo de las tecnologías Web:

- **SGML** (*Standard Generalized Markup Language*). Es un metalenguaje, es decir, se usa para crear otros lenguajes de marcas. Por ejemplo, la gramática del HTML se creó a partir del SGML. Es muy potente, pero también complejo de utilizar, motivo por el que se creó XML.

- **XML** (*eXtensible Markup Language*). Es un subconjunto de SGML. Por tanto, es también un metalenguaje, pero mucho más sencillo de usar. Su aplicación va más allá de Internet, siendo un estándar para el intercambio de información estructurada entre diferentes plataformas.

- **HTML** (*HyperText Markup Language*). Es un lenguaje de marcas más conocido ya que se usa para convertir documentos de texto en páginas web. Está bastante limitada ya que se pensó inicialmente para representar únicamente texto en un navegador web. Para solventar sus carencias se creó XHTML.

- **XHTML**. Es una reformulación de HTML en XML. En otras palabras, emplea el mismo vocabulario que HTML, pero con reglas sintácticas tomadas de XML, mucho más estrictas.

4 LENGUAJE XML: ESTRUCTURA Y SINTAXIS

El lenguaje de marcas extensible (*eXtensible Markup Language* o XML) es un lenguaje de marcas independiente del sistema operativo, que se diseñó para transportar y almacenar datos entre aplicaciones. Por eso simplifica la compartición de datos entre programas y la realización de cambios en las configuraciones de los sistemas. Permite también crear otros lenguajes a partir de él, como el lenguaje HTML, por lo que se dice que es un metalenguaje.

Aunque tiene parecidos con HTML sus aplicaciones son bien distintas. De hecho, el lenguaje HTML se creó simplemente para la visualización de datos a través de la web y no para su tratamiento automatizado. XML se centra en lo **que son los datos** (es un lenguaje descriptivo), mientras que HTML se encarga de cómo se tienen que visualizar esos datos. Por tanto, XML no es un sustituto de HTML, sino un complemento.

En XML **no existen etiquetas predefinidas**, sino que es el usuario quien las crea. Por ejemplo, una nota de Juan a Elvira puede almacenarse como un fichero XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nota>
   <destinatario>Fernando</destinatario>
   <remitente>Elvira</remitente>
   <cabecera>Recuerda</cabecera>
   <cuerpo>Llámame el fin de semana!</cuerpo>
</nota>
```

Las etiquetas del ejemplo anterior, como `<destinatario>` o `<remitente>` no están definidas en ningún estándar XML. Son marcas inventadas por el autor del documento XML. En cambio, los documentos HTML están construidos mediante etiquetas predefinidas en el estándar, como `<p>`, `<h1>`, etc. El usuario define las etiquetas XML, así como la estructura del documento.

Los ficheros XML, en sí mismos, no hacen nada más que describir los datos. Son ficheros de texto plano editables con cualquier editor. El ejemplo anterior es bastante descriptivo. Podemos leer que hay un destinatario y un remitente de la nota, así como una cabecera y un cuerpo del mensaje, pero no son más que datos estructurados mediante etiquetas. Sin embargo, cualquiera podría escribir un programa para el envío de notas y usar ficheros XML como éste para recibirlas y mostrarlas en la pantalla del usuario.

Desde que XML se convirtió en una recomendación del W3C, en febrero de 1998, su popularidad ha ido creciendo hasta convertirse en la actualidad uno de los pilares de la web. Así mismo, su uso fuera de las tecnologías web está muy extendido.

4.1 Estructura

En el ejemplo anterior, la primera línea es la **declaración XML**, en la que se define la versión y la codificación a emplear (ISO-8859-1 = Latin-1/Conjunto de caracteres de Europa Occidental). La versión por defecto es la 1.0, aunque desde el año 2006 ya está disponible la versión 1.1.

La segunda línea describe el **elemento raíz** del documento (algo así como decir "este documento es una nota").

Las siguientes 4 líneas describen 4 **elementos fijos** del elemento raíz (`destinatario`, `remitente`, `cabecera`, `cuerpo`).

La última línea contiene el cierre del elemento raíz.

Los documentos XML forman una **estructura en árbol** que comienza en la raíz y se bifurca en las hojas. El elemento raíz es el padre del resto de los elementos. A su vez, un elemento hijo puede tener otros subelementos (elementos hijos):

```
<raiz>
  <hijo>
    <subhijo>…</subhijo>
  </hijo>
</raiz>
```

Los términos padre, hijo y hermano se usan para describir las relaciones entre elementos. Los elementos padres tienen hijos y los hijos del mismo nivel son hermanos.

4.2 Etiquetas

Existen una serie de reglas que se han de tener en cuenta a la hora de escribir los nombres de las etiquetas:

- **Pueden** contener letras, números y otros caracteres, pero deben empezar con una letra.
- **No pueden** empezar con las letras xml (o XML, Xml, etc.). A excepción de esta palabra, puede usarse cualquier otra, ya que no existen palabras reservadas.
- **No pueden** contener espacios.

Se pueden escribir **etiquetas sin contenido,** pero hay que cerrarlas (`<etiqueta></etiqueta>`). Normalmente se usa una forma abreviada de escribir estas etiquetas incluyendo el símbolo `/` al final de la etiqueta y omitiendo el cierre (`<etiqueta />`).

Consejos de nomenclatura

Es conveniente tener en cuenta las indicaciones siguientes cuando escribimos los nombres de las etiquetas:

- **Usar nombres descriptivos, cortos y simples**: `<titulo>` en lugar de `<el_titulo_del_libro>`.
- **Sustituir los espacios por el guión bajo**: `<primer_apellido>`, `<segundo_apellido>`.

- **Non usar el carácter "-"**. Algunos programas pueden interpretar que `primer-apellido` se trata de una resta de dos valores.
- **Non usar el carácter "."**. Algunos programas pueden interpretar que `primer.apellido` indica que `apellido` es una propiedad del objeto `primer`.
- **Non usar el carácter ":"** Este carácter se usa en los espacios de nombres de los que se hablará más adelante.
- Los caracteres no ingleses, como **`éòá`**, son legales en XML, pero pueden tener problemas de compatibilidad.

4.3 Atributos

Los elementos pueden tener **atributos** (como en HTML). Un atributo proporciona información adicional sobre un elemento:

```
<modulo numHoras="12">Planificación y Administración de Redes</modulo>
```

Un atributo también se puede representar como un elemento hijo:

```
<modulo>
   <numHoras>12</numHoras>
   <nombre>Planificación y Administración de Redes</nombre>
</modulo>
```

Non existen reglas sobre cuándo usar atributos o elementos, pero, en general, es preferible usar elementos en lugar de atributos ya que:

- Los atributos no pueden contener múltiples valores (los elementos sí).
- Los atributos no tienen estructura de árbol (los elementos sí).
- Los atributos son difíciles de mantener (para cambios futuros en el fichero XML).

Sin embargo, existe una excepción en la que sí se aconseja el uso de atributos: cuando se usan para representar metadatos. Por ejemplo, en ocasiones hay que asignar un identificador único (ID) a un elemento:

```
<facturacion>
   <factura id="501">
      <fecha>01/01/2010</fecha>
   </factura>
   <factura id="502">
      <fecha>01/01/2010</fecha>
   </factura>
</facturacion>
```

Los atributos `id` se usan para identificar facturas, pero no son parte de la misma. Es decir, los datos se representan mediante elementos y los datos sobre los propios datos suelen representarse como atributos.

4.4 Sintaxis

Existen una serie de reglas sintácticas que se deben recordar a la hora de escribir XML:

- **Todos los elementos XML deben de tener una etiqueta de cierre.**
- **XML distingue entre mayúsculas y minúsculas**. La etiqueta <Dispositivo> es diferente de la etiqueta <dispositivo>. Por tanto, las etiquetas de apertura y cierre deben ser idénticas.

Incorrecto
```
<Mensaje>…</mensaje>
```

Correcto
```
<mensaje>…</mensaje>
```

- **Los elementos XML tienen que estar correctamente anidados.** El siguiente ejemplo es erróneo:

Incorrecto
```
<perro>
<raza>Boxer</perro>
</raza>
```

- **Los documentos XML deben tener siempre un único elemento raíz.**
- **Los atributos XML deben ir entre comillas, simples o dobles**. Por ejemplo:

Incorrecto
```
<factura id=501>
<fecha>01/01/2010</fecha>
</factura>
```

Correcto
```
<factura id="501">
<fecha>01/01/2010</fecha>
</factura>
```

- **Caracteres especiales**. Existen caracteres que tienen un significado especial para el lenguaje XML. Por ejemplo, el carácter < se puede confundir con el principio de una etiqueta. Para evitarlo se usan **entidades de referencia**. Las entidades de referencia predefinidas son:

Carácter	Referencia
&	&amp;
<	&lt;
>	&gt;

"	&quot;
'	&apos;

- **Comentarios**. La sintaxis para escribir un comentario dentro de un fichero XML es la misma que la del lenguaje HTML:

```
<!-- Esto es un comentario -->
```

5 HERRAMIENTAS DE EDICIÓN

Son programas que nos facilitan la edición de documentos XML, su validación, etc.

Como los ficheros XML son documentos de texto plano pueden crearse con cualquier programa de edición, por simple que este sea. No obstante, es más fácil crear los documentos con editores específicos para este propósito.

El uso de editores no específicos puede tener sentido si se quiere hacer una edición rápida de un documento y, además, tienen la ventaja de que se encuentran en todos los sistemas operativos. Sin embargo, tienen carencias importantes ya que no reconocen el documento XML como tal y, por tanto, no nos pueden dar soporte.

Para escribir documentos XML sin errores se necesita un editor específico que nos ayudará a validar XML contra una DTD o un esquema y forzar así estructuras XML válidas. Un editor XML debe ser capaz de proporcionarnos las siguientes funcionalidades:

- **Añadir etiquetas de cierre automáticamente**.
- **Forzar la escritura de un documento XML bien formado**, es decir, haciendo cumplir las reglas sintácticas de XML.
- **Verificar el código XML contra un DTD o un esquema**.
- **Resaltar las etiquetas y sintaxis de XML mediante colores**.

Algunas herramientas libres para la edición de documentos XML son las siguientes: XML Copy Editor, Quanta Plus o Bluefish.

6 ELABORACIÓN DE DOCUMENTOS XML BIEN FORMADOS

Un documento XML **sintácticamente correcto** se dice que está "bien formado". Las reglas de sintaxis ya se describieron:

- Los documentos XML deben tener un único elemento raíz.
- Hay que cerrar todas las etiquetas.
- Se distingue entre mayúsculas y minúsculas.
- Las etiquetas deben estar correctamente anidadas.
- Los atributos deben ir entrecomillados.

6.1 Documentos XML Válidos

Un documento XML válido es un documento "bien formado" que, además, se ajusta a las reglas de un DTD (*Document Type Definition*).

6.1.1 Declaración del DTD

El objetivo de un DTD es definir la estructura de un documento XML, es decir, establecer restricciones sobre los elementos de XML. El DTD puede especificarse **mediante un fichero externo** que contiene las restricciones sobre los datos. Por ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE nota SYSTEM "notas.dtd">
<nota>
   <destinatario>Fernando</destinatario>
   <remitente>Elvira</remitente>
   <cabecera>Recuerda</cabecera>
   <cuerpo>Llámame el fin de semana!</cuerpo>
</nota>
```

La declaración DOCTYPE es una referencia a un fichero DTD externo, llamado nota.dtd, junto con el nombre del elemento raíz del documento. La ubicación del fichero se especifica mediante un URI (es una cadena de texto que identifica un recurso de forma única en Internet o en cualquier otro sistema), por tanto, podría ser una dirección web:

```
<!DOCTYPE nota SYSTEM "http://www.servidor.org/dtd/notas.dtd">
```

Independientemente de su ubicación física, el fichero contendrá información como la siguiente:

```
<!ELEMENT nota (destinatario,remitente,cabecera,cuerpo)>
<!ELEMENT destinatario (#PCDATA)>
<!ELEMENT remitente (#PCDATA)>
<!ELEMENT cabecera (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
```

La información del DTD también puede incluirse directamente en el fichero XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE nota [
   <!ELEMENT nota (destinatario,remitente,cabecera,cuerpo)>
   <!ELEMENT destinatario (#PCDATA)>
   <!ELEMENT remitente (#PCDATA)>
   <!ELEMENT cabecera (#PCDATA)>
   <!ELEMENT cuerpo (#PCDATA)>
]>
<nota>
   <destinatario>Fernando</destinatario>
   <remitente> No obstante </remitente>
   <cabecera>Recuerda</cabecera>
   <cuerpo>Llámame el fin de semana!</cuerpo>
</nota>
```

En los dos casos, el DTD anterior tiene el siguiente significado:

`!DOCTYPE nota` indica que el elemento raíz de este documento es `nota`.

`!ELEMENT nota` indica que el elemento nota contiene cuatro elementos: `destinatario`, `remitente`, `cabecera` y `cuerpo`.

`!ELEMENT destinatario` indica que el elemento destinatario es de tipo `#PCDATA`, es decir, texto.

`!ELEMENT remitente` indica que el elemento remitente es de tipo `#PCDATA`.

`!ELEMENT cabecera` indica que el elemento cabecera es de tipo `#PCDATA`.

`!ELEMENT cuerpo` indica que el elemento remitente es de tipo `#PCDATA`.

6.1.2 Caracteres de frecuencia

La especificación de elementos de contenido puede incorporar un símbolo de frecuencia, de acuerdo con la siguiente tabla de significados:

Carácter	Descripción
+	El elemento aparece una o más veces
*	El elemento aparece cero o más veces
?	El elemento aparece cero o una vez

Así, al escribir, por ejemplo:

```
<!ELEMENT aviso (titulo?, (parrafo+, grafico)*)>
```

Se especifica que aviso puede tener un `titulo` o no (pero solo uno), y tener cero o más conjuntos cada uno de ellos formado por uno o más elementos `parrafo` seguidos de un elemento `grafico`.

6.1.3 O exclusivo

Se usa el carácter (|) para separar las distintas opciones, con significado de "o exclusivo". Por ejemplo:

```
<!ELEMENT sobremesa ( helado | pastel )>
```

Indica que sobremesa puede contener o bien un elemento `helado` o bien un elemento `pastel`.

El número de opciones no está limitado a dos, y se pueden agrupar usando paréntesis:

```
<!ELEMENT sobremesa (sorbete, (helado | pastel))>
```

6.1.4 Por qué usar DTD

Resumiendo, usamos DTD porque así:

- Cada fichero XML incorpora una descripción de su propio formato.
- Diferentes instituciones pueden ponerse de acuerdo en usar un DTD estándar para el intercambio de datos.
- Un programa puede usar un DTD estándar para verificar que los datos externos que recibe de otro programa son válidos.
- Nosotros mismos podemos verificar la validez de nuestros propios datos.

6.1.5 Esquemas XML

El W3C apoya una alternativa a los DTD llamada esquemas XML. Es otra forma de validar documentos con la ventaja de que los esquemas están basados también en XML y permiten especificar **tipos de datos** para los elementos. Los DTD son anteriores los esquemas XML y previsiblemente irán siendo sustituidos por éstos, aunque de momento, conviven las dos tecnologías.

Un fichero de esquema XML para el documento que almacena notas podría llamarse notas.xsd y tener el siguiente aspecto:

```
<xs:element name="nota">
<xs:complexType>
   <xs:sequence>
      <xs:element name="destinatario" type="xs:string"/>
      <xs:element name="remitente" type="xs:string"/>
      <xs:element name="cabecera" type="xs:string"/>
      <xs:element name="cuerpo" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
</xs:element>
```

Los programas de ayuda para la edición de documentos XML, como XML Copy Editor, incluyen herramientas para validarlos, bien a través de DTD o de esquemas XML.

7 UTILIZACIÓN DE ESPACIOS DE NOMBRES EN XML

Los espacios de nombres XML (*XML Namespaces* o xmlns) permiten evitar conflictos entre elementos XML que se llamen igual. Esto ocurre cuando mezclamos documentos XML que provienen de diferentes aplicaciones. Por ejemplo, podemos tener un fichero XML de países:

```
<?xml version="1.0" encoding="UTF-8"?>
<pais nombre="Francia">
   <capital>Paris</capital>
</pais>
```

Y otro fichero de inversiones con elementos que se llamen igual:

```
<?xml version="1.0" encoding="UTF-8"?>
<inversión>
   <capital>1200</capital>
</inversión>
```

Si mezclamos ambos ficheros en un único documento XML de inversiones tendremos problemas para diferenciar la capital de un país del capital invertido:

```
<inversiones>
   <pais nombre="Francia">
      <capital>Paris</capital>
      <capital>1200</capital>
   </pais>
</inversiones>
```

Para evitar estos problemas, asignamos un espacio de nombres a cada uno de los elementos o atributos de un documento XML y así, en caso de combinarlos, no colisionarán. Podremos

tener una misma etiqueta llamada **<capital>** para países e inversiones y no se confundirían, puesto que cada una estará precedida por el identificador de su espacio de nombres.

7.1 Espacios de nombres con prefijo

El espacio de nombres se define mediante el atributo **xmlns** con la sintaxis **xmlns:prefijo="URI del espacio de nombres"**, y se aplica escribiendo los nombres de las etiquetas o de los atributos del prefijo seguido del símbolo **':'**. Veámoslo con un ejemplo:

```
<b:inversiones xmlns:b="http://www.bolsa.com" xmlns:g="http://www.geografia.es">
   <g:pais g:nombre=”Francia”>
       <g:capital>Paris</g:capital>
       <b:capital>1200</b:capital>
   </g:pais>
</b:inversiones>
```

El prefijo es solo una forma corta de hacer referencia al identificador (o URI), que es lo que realmente identifica el espacio de nombres.

7.2 Espacio de nombres por defecto

También se define mediante el atributo **xmlns**. El espacio de nombres se aplicará al elemento en el que se incluye este atributo y a todo su contenido (siempre que no esté a su vez asociado a otro espacio de nombres). Veamos un ejemplo:

```
<inversiones xmlns="http://www.bolsa.com">
   <g:país xmlns:g="http://www.geografia.es" g:nombre="Francia">
      <g:capital>Paris</g:capital>
      <capital>1200</capital>
   </g:pais>
</inversiones>
```

Podemos observar que en las líneas 1, 4 y 6 se aplica el espacio de nombres por defecto definido en la línea 1. En las líneas 2, 3 y 5 se aplica mediante prefijo el espacio de nombres definido en la línea 2.