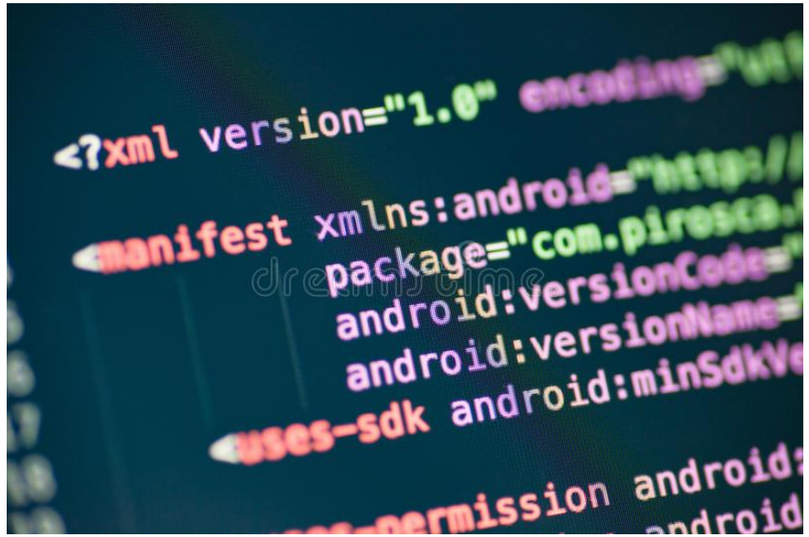
Lenguajes de marcas y sistemas de gestión de la información



UT04 – XML

1 – Introducción – Estructura – Sintaxis

IES Clara del Rey – Madrid

Qué es XML

XML: E**x**tensible **M**arkup **L**anguage

Lenguaje de marcas (que **no de programación**) desarrollado en 1998 por el W3C.

Basado en SGML., pero simplificándolo, haciéndolo más fácil de usar.

Metalenguaje: permite definir otros lenguajes con un propósito específico (SVG, XAML, MathML, XHTML, etc.)

Versiones: 1.0 y 1.1. Para mayor compatibilidad, se recomienda usar la 1.0. La 1.1 simplemente añade mejor tratamiento de ciertos caracteres. Sólo se recomienda cuando sea realmente necesario.

Ejemplo de documento XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<centro>
    <nombre>IES Clara del Rey</nombre>
    <direccion>
        <tipoVia>Calle</tipoVia>
        <nombreVia>C/ Padre Claret</nombreVia>
        <numero>8</numero>
        <codigoPostal>28004</codigoPostal>
        <provincia>Madrid</provincia>
        <ciudad>Madrid</ciudad>
    </direccion>
    <departamentos>
        <departamento>
            <nombre>Informatica y comunicaciones</nombre>
            <numeroProfesores>23</numeroProfesores>
        </departamento>
        <departamento>
            <nombre>Formación y orientación laboral</nombre>
            <numeroProfesores>5</numeroProfesores>
        </departamento>
    </departamentos>
</centro>
```

XML vs HTML

Objetivo:

- HTML, además de la información, puede incluir directrices sobre la forma de presentación (no recomendado desde HTML 5 – Usar CSS).
- XML está diseñado exclusivamente para almacenar y transferir información. Se centra en la información, no en la presentación.

Conjunto de etiquetas:

- HTML tiene un conjunto definido de etiquetas / elementos, cada una diseñada para representar un tipo de información específica.
- En XML las etiquetas / elementos no están predefinidos. Se puede usar cualquier palabra como etiqueta, siempre que cumpla ciertas reglas de sintaxis.

Estructura de un documento XML

Un documento XML tiene una estructura en árbol.

El árbol XML comienza en un elemento raíz, con ramas formadas por otros elementos, que a su vez pueden tener más ramas.

Algo de terminología:

- Nodo (node)
- Elemento (element)
- Nodo / elemento raíz
  (root node / element)
- Hijo / hijos (child / children)
- Hermano (sibling)
- Atributo (attribute)

```
<?xml version="1.0" encoding="UTF-8"?>
<raiz>
    <hijo nombre="">
        <nieto>...</nieto>
        <nieto>...</nieto>
    </hijo>
    <hijo nombre="">
        <nieto>...</nieto>
        <nieto>...</nieto>
    </hijo>
</raiz>
```

La declaración XML

Un documento XML puede comenzar con una declaración o prólogo (opcional, pero recomendable). No tiene tag de cierre.

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

Si se incluye, debe ser lo primero en el documento, y debe incluir el atributo “version“.

Atributos opcionales:

- encoding: indica la codificación del texto
- standalone (yes/no): Lo veremos más adelante.

En la declaración XML los atributos versión, encoding y standalone deben mantener este orden.

Declaración de tipo de documento (DTD)

Opcional, permite definir las reglas que debe cumplir el documento XML.

Puede ser: una referencia a un documento DTD:

```
<!DOCTYPE nombreNodoRaiz SYSTEM "file.dtd">
```

Puede ser la propia definición de la DTD:

```
<!DOCTYPE nombreNodoRaiz [
    <!-- Declaraciones DTD -->
]>
```

Y otras opciones que veremos cuando estudiemos la validación de documentos.

Elementos XML

Un elemento XML tiene la forma:

```
<etiqueta atributo="[valor]">[Contenido]</etiqueta>
```

El elemento está formado por el conjunto de la etiqueta (con tag de apertura y de cierre), sus atributos, y su contenido.

Es recomendable que las etiquetas y atributos tengan nombre descriptivo.

El contenido de un atributo puede ser texto, otro elemento, o varios elementos, y puede o no tener atributos.

Si un elemento no tiene contenido puede, opcionalmente, presentarse de forma abreviada. Estas dos representaciones son equivalentes:

```
<elemento-sin-contenido id="id"></elemento-sin-contenido>
<elemento-sin-contenido id="id" />
```

Nombres de atributos y nodos

Todos los nombres, tanto de elementos como de atributos, son "case sensitive". Se distingue entre mayúsculas y minúsculas.

Los nombres deben comenzar por una letra o un guion bajo ("_")

Pueden contener letras, números, puntos, guiones medios y bajos.

No pueden contener espacios.

Pueden contener los dos puntos (“:”), pero su uso queda reservado para espacios de nombres (namespaces), que veremos más adelante.

Aunque las tildes y otros caracteres especiales están permitidos, se recomienda no utilizarlos.

Atributos

Cualquier elemento XML puede tener atributos.

Los atributos siempre se colocan en el tag de apertura.

El atributo aporta información adicional sobre el elemento.

Un atributo debe ser único, no puede repetirse en el mismo elemento.

```
<!-- Mal, se repite el atributo -->
<persona aficion="musica" aficion="deporte">...</persona>

<!-- Bien, el atributo es único -->
<persona edad="30">...</persona>
```

Comentarios

Igual que HTML.

Empiezan con "<!--" y se cierran con "-->".

No se pueden anidar ni pueden contener dos guiones seguidos ("--").

No se pueden colocar dentro de otra etiqueta.

```
<!-- Esto es un comentario -->
<!-- Este comentario -- está mal -->
<elemento <!-- Aquí no puede ir -->></elemento>
```

Elementos vs atributos

¿Usar elementos hijos o atributos? No hay reglas concretas.

Es igual de válido guardar información en atributos o en elementos. Sí hay que respetar que los atributos no pueden repetirse en un elemento.

```
<!-- Mal, se repite el atributo aficion -->
<persona aficion="música" aficion="deporte">
    <nombre>José Luis</nombre>
</persona>
<!-- Bien, aficiones como nodos hijos -->
<persona>
    <nombre>José Luis</nombre>
    <aficion>música</aficion>
    <aficion>deporte</aficion>
</persona>
```

Entidades

Hay ciertos caracteres que tienen un uso muy específico en XML:

- < y > para escritura de elementos.
- " (comilla doble) y ' (comilla simple) para delimitar atributos.
- & (ampersand) para indicar referencia a entidad.

Estos símbolos no se pueden usar libremente en elementos o atributos.

Hay que escaparlos, según la siguiente tabla:

Referencias a entidades en XML

Carácter	*Entidad*	*Referencia a entidad*
< (*menor que*)	lt (*less than*)	<
> (*mayor que*)	gt (*greater than*)	>
" (*comilla doble*)	quot (*quotation mark*)	"
' (*comilla simple*)	apos (*apostrophe*)	'
& (*ampersand*)	amp (*ampersand*)	&

Caracteres especiales

Se puede hacer referencia a caracteres Unicode con “&#”, seguido del valor decimal o hexadecimal del carácter, y terminando con “;”

```
<elemento>Gato sonriente: &#128572;</elemento>
<elemento>Perro guía: &#129454;</elemento>
<elemento>Perro guía (hex): &#x1f9ae;</elemento>
```

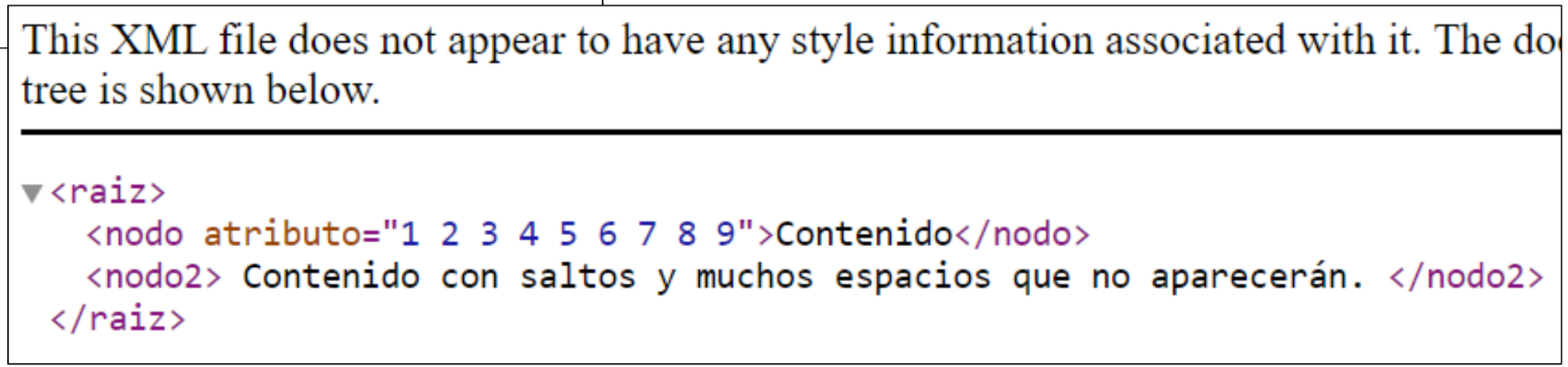
Editores modernos ya tienen soporte para estos caracteres, pero hay buscadores para localizarlos:

https://www.mclean.net.nz/ucf

Espacios en blanco

Al igual que en HTML, los espacios en blanco, tabulaciones y saltos de línea, cuando están seguidos, se interpretan como un solo espacio.

```
<nodo atributo="1 2 3
                4 5 6
                    7 8 9">Contenido</nodo>
<nodo2>
    Contenido con
    saltos      y muchos espacios
    que         no aparecerán.
</nodo2>
```



Respetar espacios en blanco

Usando el atributo xml:space=“preserve” se deberían respetar los espacios al mostrar el XML en una aplicación que lo procese. Pero muy pocas lo hacen. Por ejemplo, en los navegadores no se suele respetarse.

```
<elemento xml:space="preserve">
    Esto debería                mantener

    los espacios

    pero casi seguro que no lo hace.
</elemento>
```

Sección CDATA

CDATA significa "Character DATA"

Permite escribir texto que no debe ser interpretado como XML.

Se abre con <![CDATA[y se cierra con]]>. No puede incluir dentro]]>.

En una sección CDATA se puede escribir XML.

```
<datos>
<![CDATA[En este texto se pueden escribir caracteres
    que normalmente están reservados para XML, como
    < o >, o las comillas ". No se pueden escribir dos
    cierres de corchete juntos seguidos de un símbolo
    >, porque es el cierre de CDATA.]]>
</datos>
```

En CDATA no es necesario (y no se debe) escapar ningún carácter.

Instrucciones de procesamiento

Sirve para dar información al programa que procesará el XML.

Comienzan con "<?" y terminan con "?>". No tienen cierre.

Aunque la declaración XML se escribe también entre "<?" y "?>", no es una instrucción de procesamiento propiamente dicha.

Las más habituales son las usadas para que se utilice una hoja de estilos CSS, o para aplicar una transformación XSLT.

Incluir una hoja de estilos CSS:

```
<?xml-stylesheet href="estilos.css" type="text/css"?>
```

Aplicar una plantilla de transformación XSLT:

```
<?xml-stylesheet href="plantilla.xsl" type="text/xsl"?>
```

Documentos bien formados

Un documento bien formado cumple las reglas sintácticas de XML.

No confundir un XML bien formado con un XML válido.

XML es un metalenguaje con el que se definen otros lenguajes más específicos, como MathML, ChemML, BeerML, Office Open XML, etc.

https://en.wikipedia.org/wiki/List_of_XML_markup_languages

Para que un XML sea válido, debe estar bien formado, y además cumplir con la especificación para la que se diseñó.

Por lo tanto, podemos tener un documento XML bien formado, que no sea un documento válido, y el programa que debe procesarlo puede rechazarlo.

Algunas reglas generales

- Sólo puede haber un elemento raíz.
- Los elementos con contenido tienen siempre etiqueta inicial y final.
- Los elementos sin contenido pueden “compactarse” en una sola etiqueta (ej: <etiqueta/>)
- Un atributo no puede repetirse en un elemento.
- El valor de un atributo debe estar entre comillas dobles o simples.
- Respetar estructura jerárquica. Cerrar correctamente etiquetas y respetar el anidamiento. XML es sensible a mayúsculas/minúsculas. Ojo con aperturas y cierres de elementos con distinta convención.
- Escapado de caracteres especiales. Guía (en inglés): https://www.novixys.com/blog/what-characters-need-to-be-escaped-in-xml-documents/