

T4.1. ¿QUÉ ES XML?



4.1.1. ¿QUÉ ES XML Y PARA QUÉ SIRVE?

4.1.2. CONCEPTOS BÁSICOS.

- 4.1.2.1. Ejemplo de documento XML.
- 4.1.2.2. Conceptos y vocabulario.
- 4.1.2.3. Documentos bien formados.
- 4.1.2.4. Documentos válidos.

4.1.3. EJERCICIOS SOBRE DOCUMENTOS BIEN FORMADOS.

4.1.1. ¿Qué es XML y para qué sirve?

XML son las siglas de «*eXtensible Markup Language*». Los archivos XML son una forma de almacenar datos para que estos sean leídos fácilmente todos los navegadores de forma universal.

Los archivos XML se utilizan principalmente para exportar e importar datos de bases de datos de forma universal. Esto se debe a que principalmente este lenguaje está basado en un lenguaje universal (XML), simplificado y adaptado a Internet.

Además, no pertenece a ninguna compañía y esto provoca que los archivos XML sean de uso universal.

Sin embargo, y en contra de lo que pudiera parecer, el XML no es un lenguaje de marcado, ni una versión mejorada de HTML, ni sirve para diseñar páginas web.

Un documento XML sirve para incluir cualquier flujo de datos basado únicamente en texto como puede ser un artículo de una revista, un resumen de cotizaciones de bolsa, un conjunto de registros de una base de datos, etc.

¿Para qué sirve XML?

XML sirve para presentar información estructurada en una página web de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa por diversas aplicaciones y dispositivos.

Es decir, el XML es un archivo web que sirve principalmente para la transmisión de datos agregados (En bases de datos) entre diferentes aplicaciones y plataformas de forma universal.

RESUMEN:

- XML Almacena y transporta datos en un formato que sea totalmente legible para los usuarios y para la máquina.
- NO es una base de datos, NO es un lenguaje de programación, es un modo totalmente estructurado para presentar los datos y para ello presenta etiquetas o tags como hace html.
- Presenta un entorno muy familiar para quien sabe HTML.
- No existe una lista maestra de tags, nosotros creamos las etiquetas que queramos.
- No tiene un formato.
- XML solo describe los datos que contiene, no hace referencia a lo que se muestra en la pantalla.

4.1.2. CONCEPTOS BÁSICOS.

4.1.2.1. Ejemplo de documento XML.

Un documento XML es similar a una página web, salvo que los nombres de las etiquetas y atributos no son los del HTML.

```
<?xml version="1.0" encoding="UTF-8"?>
<países>
  <pais>
    <nombre>España</nombre>
    <capital>Madrid</capital>
  </pais>
  <pais>
    <nombre>Francia</nombre>
    <capital>París</capital>
  </pais>
</países>
```

Por supuesto, la misma información se puede guardar con otra estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
<países>
  <pais nombre="España" capital="Madrid" />
  <pais nombre="Francia" capital="París" />
</países>
```

4.1.2.2.

Conceptos y vocabulario.

Documento XML

Un documento XML es un documento de texto plano (sin formato).

Procesador XML (*XML processor*) y aplicación (*application*)

Cuando una aplicación necesita leer un documento XML, la aplicación recurre a un procesador XML. El procesador XML (o analizador XML, en inglés *XML parser*) es el que lee el documento, analiza el contenido y le pasa la información en un formato estructurado a la aplicación. La recomendación XML especifica lo que debe hacer el procesador, pero no entra en lo que hace después la aplicación con esa información.

Caracteres (*characters*)

Los documentos XML pueden estar codificados en distintos juegos de caracteres (UTF-8, ISO-8859-1, etc).

¿Cuál es la diferencia entre UTF-8 e ISO-8859-1?

<https://www.desarrollo-web-br-bd.com/es/utf-8/cual-es-la-diferencia-entre-utf-8-e-iso-8859-1/940246725/>

Marcas (*mark-up*) y contenido (*content*)

El texto que contiene un documento XML se divide en marcas y contenido. Las marcas pueden ser de dos tipos: etiquetas o referencias a entidades. Todo lo que no son marcas es contenido.

Elementos (*elements*)

Un elemento es un componente lógico de un documento que o bien comienza por una etiqueta de apertura y termina por la etiqueta de cierre correspondiente o que consiste en una única etiqueta vacía. El contenido de un elemento es todo lo que se encuentra entre las etiquetas de apertura y cierre, incluso si estos son también elementos en cuyo caso se llaman elementos hijos.

Etiquetas (*tags*)

Una etiqueta es un identificador que empieza por el carácter `<` y termina por `>`. Existen tres tipos de etiquetas:

- las etiquetas de apertura (*start-tag*), que empiezan por el carácter `<` y terminan por `>`. Por ejemplo: `<apartado>`
- las etiquetas de cierre (*end-tag*), que empiezan por los caracteres `</` y terminan por `>`. Por ejemplo: `</apartado>`
- las etiquetas vacías (*empty tag*), que empiezan con el carácter `<` y terminan por `/>`. Por ejemplo: `<linea />`

Atributos (*attributes*)

Un atributo es un componente de las etiquetas que consiste en una pareja nombre (*name*) / valor (*value*). Se puede encontrar en las etiquetas de apertura o en las etiquetas vacías, pero no en las de cierre. En una etiqueta no puede haber dos atributos con el mismo nombre. La sintaxis es siempre `nombreAtributo="valorAtributo"`. Por ejemplo:

```
<profesor nombre="Angela" apellidos="Estornell Huguet" />
```

Comentarios (*comments*)

Un comentario es una etiqueta que comienza por `<!--` y acaba por `-->`. Los comentarios no pueden estar dentro de otras marcas y no pueden contener los caracteres `--`. Dentro de un comentario las entidades de carácter no se reconocen, es decir, sólo se pueden utilizar los caracteres del juego de caracteres del documento. Por ejemplo:

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

Declaración XML (*XML declaration*)

La declaración XML es una etiqueta que comienza por `<?xml` y termina por `?>` y que proporciona información sobre el propio documento XML. Aunque no es obligatoria es conveniente que aparezca, y debe aparecer siempre al principio del documento. No es una instrucción de procesamiento, pero tiene la misma sintaxis (empieza por `<?` y acaba por `?>`).

La declaración xml indica el juego de caracteres del documento. El juego de caracteres que se utiliza en este curso es UTF-8:

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

En XML, el nombre del juego de caracteres se debe escribir en mayúsculas (UTF-8 en vez de utf-8).

La declaración xml también indica la versión XML utilizada. Aunque existe la versión XML 1.1, la versión más común sigue siendo la versión XML 1.0.

Se pueden utilizar otros juegos de caracteres, como ISO-8859-1 (Europeo occidental):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Es importante que el juego de caracteres que aparece en la declaración sea el juego de caracteres en que realmente está guardado el documento, porque si no el procesador XML puede tener problemas leyendo el documento.

Definición de tipo de documento (DTD, *Document Type Definition*)

Una DTD es un documento que define la estructura de un documento XML: los elementos, atributos, entidades, notaciones, etc, que pueden aparecer, el orden y el número de veces que pueden aparecer, cuáles pueden ser hijos de cuáles, etc. El procesador XML la utiliza para verificar si un documento es válido, es decir, si el documento cumple las reglas del DTD.

Declaración de tipo de documento (DOCTYPE, *Document type declaration*)

Una declaración de tipo de documento es una etiqueta que comienza por `<!DOCTYPE` y acaba por `>` y que indica la(s) DTD(s) que debe utilizar el procesador XML para validar el documento. La DTD puede estar incluida en el propio documento o ser un documento externo. Por ejemplo, el siguiente ejemplo muestra la declaración de tipo de documento que se debe incluir en los documentos XHTML 1.0 de tipo strict (en este caso, la DTD es un documento externo):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Instrucciones de procesamiento (PI, *processing instruction*)

Una instrucción de procesamiento es una etiqueta que empieza por `<?>` y acaba por `?>` y que contiene instrucciones dirigidas a las aplicaciones que leen el documento. Pueden aparecer en cualquier lugar del documento. Por ejemplo:

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




Entidades de carácter

En XML se utilizan varias entidades de carácter de HTML, para poder escribir en cadenas de texto los caracteres que delimitan las marcas o las cadenas de texto :

Referencia a entidad	Carácter
<	<
>	>
&	&
'	'
"	"

Secciones CDATA (CDATA section)

Una sección CDATA es una etiqueta que comienza por **<![CDATA[** y termina por **]]>** y cuyo contenido el procesador XML no interpreta como marcas sino como texto. Es decir, que si aparecen los caracteres especiales (< & " ') en una sección CDATA, el procesador XML no interpreta que empieza una marca, sino que lo considera un carácter más. Se suele utilizar en documentos en los que aparecen muchas veces esos caracteres especiales para no tener que estar utilizando las referencias a entidades (< y &) que dificultan la lectura del documento.

 <pre><?xml version="1.0" encoding="UTF-8"?> <prueba> <texto>Los caracteres < y & no pueden escribirse si no es como comienzo de marcas</texto> </prueba></pre>	<div>This page contains the following errors: error on line 3 at column 26: Starttag: invalid element name Below is a rendering of the page up to the first error.</div> <div>Los caracteres</div>
 <pre><?xml version="1.0" encoding="UTF-8"?> <prueba> <texto>Los caracteres &lt; y &amp; no pueden escribirse si no es como comienzo de marcas</texto> </prueba></pre>	<div>Los caracteres < y & no pueden escribirse si no es como comienzo de marcas</div>
 <pre><?xml version="1.0" encoding="UTF-8"?> <prueba> <texto><![CDATA[Los caracteres < y & no pueden escribirse si no es como comienzo de marcas]]></texto> </prueba></pre>	<div>Los caracteres < y & no pueden escribirse si no es como comienzo de marcas</div>

4.1.2.3. Documentos bien formados.

Un documento XML debe estar bien formado, es decir debe cumplir las reglas de sintaxis de la recomendación XML. Para que un documento esté bien formado, al menos debe cumplir los siguientes puntos:

- El documento contiene únicamente caracteres Unicode válidos.
- Hay un elemento raíz que contiene al resto de elementos.
- Los nombres de los elementos y de sus atributos no contienen espacios.
- El primer carácter de un nombre de elemento o de atributo puede ser una letra, dos puntos (:) o subrayado (_).
- El resto de caracteres pueden ser también números, guiones (-) o puntos (.).
- Los caracteres "<" y "&" sólo se utilizan como comienzo de marcas.
- Las etiquetas de apertura, de cierre y vacías están correctamente anidadas (no se solapan) y no falta ni sobra ninguna etiqueta de apertura o cierre.
- Las etiquetas de cierre coinciden con las de apertura (incluso en el uso de mayúsculas y minúsculas).
- Las etiquetas de cierre no contienen atributos.
- Ninguna etiqueta tiene dos atributos con el mismo nombre.
- Todos los atributos tienen algún valor.
- Los valores de los atributos están entre comillas (simples o dobles).
- No existen referencias en los valores de los atributos.

Si un documento XML no está bien formado, no es un documento XML. Los procesadores XML deben rechazar cualquier documento que contenga errores.

4.1.2.4. Documentos válidos.

Un documento XML bien formado puede ser válido. Para ser válido, un documento XML debe:

- incluir una referencia a una gramática,
- incluir únicamente elementos y atributos definidos en la gramática,
- cumplir las reglas gramaticales definidas en la gramática.

Existen varias formas de definir una gramática para documentos XML, las más empleadas son :

- DTD (Document Type Definition = Definición de Tipo de Documento). Es el modelo más antiguo, heredado del SGML.
- XML Schema. Es un modelo creado por el W3C como sucesor de las DTDs.
- Relax NG. Es un modelo creado por OASIS, más sencillo que XML Schema.

4.1.3. EJERCICIOS SOBRE DOCUMENTOS BIEN FORMADOS.

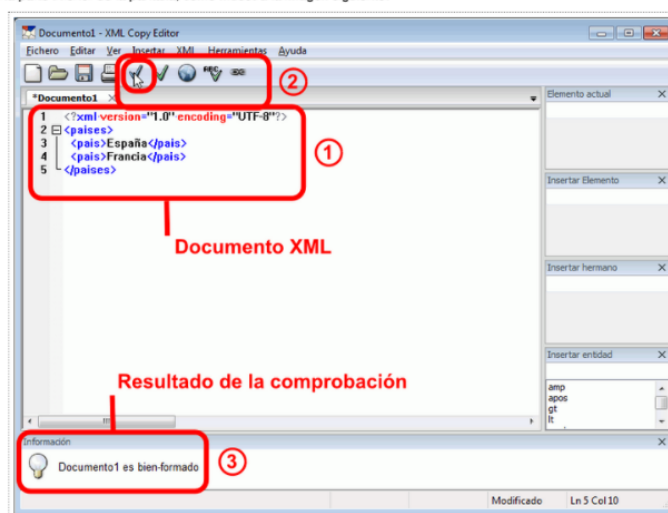
Para hacer estos ejercicios utilizaremos **XML Copy Editor** que es un editor de documentos XML libre (GPL 2.0) y multiplataforma cuya página web es <https://xml-copy-editor.sourceforge.io/> y ahí es donde podremos descargarlo.

Una vez descargado e instalado, ¿Cómo saber en XML Copy Editor si un documento está bien formado

Comprobar que un documento está bien formado

Para comprobar si un documento está bien formado, se puede elegir el menú **XML > Comprobar Bien-Formado**, hacer clic en el botón correspondiente, o pulsar la tecla **F2**.

- En caso de que el documento esté bien formado, el programa lo indica en la parte inferior de la pantalla, como muestra la imagen siguiente:




Los siguientes 7 documentos no están bien formados porque cada uno contiene dos errores (dos errores del mismo tipo cuentan como uno sólo). Corrija los errores y compruebe con XML Copy Editor que ya son documentos bien formados (si para corregir algún error hay que inventarse una etiqueta o atributo, utilice un nombre que tenga relación con la información contenida en el documento).

Aunque en XML Copy Editor no es necesario guardar los documentos para comprobar que están bien formados, se recomienda guardarlos para poder consultarlos en el futuro. El nombre del archivo podría ser `wdf_XX.xml`, donde `XX` es el número del ejercicio. **Como ejemplo ya tenéis resuelto el ejercicio 1.**

WFD - Ejercicio 1 – Deportistas

```
<?xml version="1.0" encoding="UTF-8"?>
<deportistas>
  <deportista>
    <deporte Atletismo />
    <nombre>Jesse Owens</nombre>
  </deportista>
  <deportista>
    <deporte Natación />
    <nombre>Mark Spitz</nombre>
  </deportista>
</deportistas>
```

- En las etiquetas `<deporte>` (líneas 4 y 7) aparecen los términos Atletismo y Natación sueltos.

 Error at line 4, column 24: not well-formed (invalid token)


Se podría corregir escribiendo esos términos como valores de un atributo:

```
<deporte nombre="Atletismo" />
...
<deporte nombre="Natación" />
```

o como texto dentro de la etiqueta:

```
<deporte>Atletismo</deporte>
...
<deporte>Natación</deporte>
```

- La primera etiqueta `<deportista>` no está cerrada.

 Error at line 10, column 3: mismatched tag

Se podría corregir cerrándola:

```
<deportista>
  <deporte Atletismo />
  <nombre>Jesse Owens</nombre>
</deportista>
<deportista>
...
```

- Una posible solución sería entonces:

```
<?xml version="1.0" encoding="UTF-8"?>
<deportistas>
  <deportista>
    <deporte nombre="Atletismo" />
    <nombre>Jesse Owens</nombre>
  </deportista>
  <deportista>
    <deporte nombre="Natación" />
    <nombre>Mark Spitz</nombre>
  </deportista>
</deportistas>
```

WFD - Ejercicio 2 - Películas

```
<?xml version="1.0" encoding="UTF-8"?>
<pelicula>
  <titulo>Con faldas y a lo loco</titulo>
  <director>Billy Wilder</director>
</pelicula>
<pelicula>
  <director>Leo McCarey</director>
  <titulo>Sopa de ganso</titulo>
</pelicula>
<autor />barto</autor>
```

WFD - Ejercicio 3 - Texto

```
<?xml version="1.0" encoding="UTF-8"?>
<texto>
  <Titulo>XML explicado a los niños</titulo>
  <párrafo>El <abreviatura>XML</abreviatura>define cómo crear
  lenguajes de marcas.</párrafo>
  <párrafo>Las marcas se añaden a un documento de texto
  para añadir información.</párrafo>
  <http://>www.example.org</http://>
</texto>
```

WFD - Ejercicio 4 - Información geográfica

```
<?xml version="1.0" encoding="UTF-8"?>
<geografia mundial>
  <pais>
    <pais>España</pais>
    <continente>Europa</continente>
    <capital></capital nombre="Madrid">
  </pais>
</geografia mundial>
```

WFD - Ejercicio 5 - Programas

```
<?xml version="1.0" encoding="UTF-8"?>
<programas>
  <programa nombre="Firefox" licencia="GPL" licencia="MPL" />
  <programa nombre="LibreOffice" licencia="LGPL" />
  <programa nombre="Inkscape" licencia=GPL />
</programas>
```

WFD - Ejercicio 6 - Mundiales de fútbol

```
<?xml version="1.0" encoding="UTF-8"?>
<mundiales-de-futbol>
  <mundial>
    <pais="España" />
    <1982 />
  </mundial>
</mundiales-de-futbol>
```

WFD - Ejercicio 7 - Medios de transporte

```
<?xml version="1.0" encoding="UTF-8"?>
<mediosDeTransporte>
  <bicicleta velocidad="v<100km/h" />
  <patinete velocidad maxima="50 km/h" />
</mediosDeTransporte>
```