



UTN.BA FACULTAD
REGIONAL
BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

**Centro de
e-Learning**

EXPERTO UNIVERISTARIO EN MySQL Y PHP NIVEL AVANZADO



www.sceu.frba.utn.edu.ar/e-learning



Módulo II

CONTINUAMOS CON PHP AVANZADO



PHP Y XML.



Presentación de la Unidad:

En esta unidad aprenderán el lenguaje de etiquetas XML, como debe ser la estructura de un documento XML bien formado.

Aprenderán a leer desde PHP un documento en formato XML y aprenderán a armar un documento XML desde los datos de una tabla SQL.



Objetivos:

- ❖ Conocerla estructura de un documento XML bien formado.
- ❖ Aprender a leer un documento XML mediante una de las funciones que PHP dispone para tal fin.
- ❖ Armar un XML desde PHP para que pueda ser tomado e interpretado por terceros.



Temario:

- 1 Historia
- 2 Críticas
- 3 Ventajas del XML
- 4 Estructura de un documento XML
 - 4.1 Documentos XML bien formados y control de errores
 - 4.2 Partes de un documento XML
 - 4.2.1 Prólogo
 - 4.2.2 Cuerpo
 - 4.3 Elementos
 - 4.4 Atributos
 - 4.5 Entidades predefinidas
 - 4.6 Secciones CDATA
 - 4.7 Comentarios
- 5 Validez
 - 5.1 *Document Type Definition*
 - 5.1.1 Declaraciones tipo elemento
 - 5.1.2 Modelos de contenido
 - 5.1.3 Declaraciones de lista de atributos
 - 5.1.4 Tipos de atributos
 - 5.1.5 Declaración de entidades
 - 5.1.6 Espacios de nombres
 - 5.2 XML Schemas (XSD)
 - 5.2.1 Ventajas de los Schemas frente a los DTD
- 6 Herramientas para trabajar con documentos XML
- 7 Lenguajes creados usando XML
 - 7.1 Extensible Stylesheet Language (XSL)
 - 7.2 Lenguaje de enlace XML (XLINK)
- 8 Ejemplos



CONSIGNAS PARA EL APRENDIZAJE COLABORATIVO

En esta Unidad los participantes se encontrarán con diferentes tipos de consignas que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

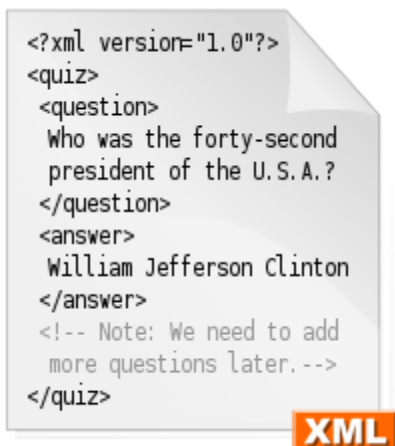
1. Los foros asociados a cada una de las unidades.
2. La Web 2.0.
3. Los contextos de desempeño de los participantes.



Es importante que todos los participantes realicen las actividades sugeridas y compartan en los foros los resultados obtenidos.

EXTENSIBLE MARKUP LANGUAGE

eXtensible Markup Language (XML)



```

<?xml version="1.0"?>
<quiz>
  <question>
    Who was the forty-second
    president of the U.S.A.?
  </question>
  <answer>
    William Jefferson Clinton
  </answer>
  <!-- Note: We need to add
    more questions later.-->
</quiz>
  
```

Desarrollador

World Wide Web Consortium

Información general

Extensión de archivo	.xml
Tipo de MIME	application/xml, text/xml
Tipo de formato	Lenguaje de marcado
Estándar(es)	1.0 (Fourth Edition) 1.1 (Second Edition)

XML, siglas en inglés de *eXtensible Markup Language* ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C). Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información.

XML no ha nacido sólo para su aplicación para Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.



XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

HISTORIA

XML proviene de un lenguaje inventado por IBM en los años setenta, llamado GML (*Generalized Markup Language*), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información. Este lenguaje gustó a la ISO, por lo que en 1986 trabajaron para normalizarlo, creando SGML (*Standard Generalized Markup Language*), capaz de adaptarse a un gran abanico de problemas. A partir de él se han creado otros sistemas para almacenar información.¹

En el año 1989 Tim Berners Lee creó la web, y junto con ella el lenguaje HTML. Este lenguaje se definió en el marco de SGML y fue de lejos la aplicación más conocida de este estándar. Los navegadores web sin embargo siempre han puesto pocas exigencias al código HTML que interpretan y así las páginas web son caóticas y no cumplen con la sintaxis. Estas páginas web dependen fuertemente de una forma específica de lidiar con los errores y las ambigüedades, lo que hace a las páginas más frágiles y a los navegadores más complejos.

Otra limitación del HTML es que cada documento pertenece a un vocabulario fijo, establecido por el DTD. No se pueden combinar elementos de diferentes vocabularios. Asimismo es imposible para un intérprete (por ejemplo un navegador) analizar el documento sin tener conocimiento de su gramática (del DTD). Por ejemplo, el navegador sabe que antes de una etiqueta <div> debe haberse cerrado cualquier <p> previamente abierto. Los navegadores resolvieron esto incluyendo lógica ad hoc para el HTML, en vez de incluir un analizador genérico. Ambas opciones, de todos modos, son muy complejas para los navegadores.

Se buscó entonces definir un subconjunto del SGML que permita:

- Mezclar elementos de diferentes lenguajes. Es decir que los lenguajes sean extensibles.
- La creación de analizadores simples, sin ninguna lógica especial para cada lenguaje.
- Empezar de cero y hacer hincapié en que no se acepte nunca un documento con errores de sintaxis.

Para hacer esto XML deja de lado muchas características de SGML que estaban pensadas para facilitar la escritura manual de documentos. XML en cambio está orientado a hacer las cosas más sencillas para los programas automáticos que

necesiten interpretar el documento.

CRÍTICAS

XML y sus extensiones han sido regularmente criticadas por su nivel de detalle y complejidad.² El mapeo del modelo de árbol básico de XML hacia los sistemas de tipos de lenguajes de programación o bases de datos puede ser difícil, especialmente cuando se utiliza XML para el intercambio de datos altamente estructurados entre aplicaciones, lo que no era su objetivo primario de diseño. Otras críticas intentan refutar la afirmación de que XML es un lenguaje autodescriptivo³ (aunque la especificación XML no hace ninguna afirmación de este tipo). Se propone a JSON y YAML frecuentemente como alternativas, centrándose ambas en la representación de datos estructurados, en lugar de documentos narrativos.

VENTAJAS DEL XML

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.
- Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

ESTRUCTURA DE UN DOCUMENTO XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman *elementos*, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde *nombre* es el nombre del elemento que se está señalando.

A continuación se muestra un ejemplo para entender la estructura de un documento XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
  <Mensaje>
    <Remitente>
      <Nombre>Nombre del remitente</Nombre>
      <Mail> Correo del remitente </Mail>
    </Remitente>
    <Destinatario>
      <Nombre>Nombre del destinatario</Nombre>
      <Mail>Correo del destinatario</Mail>
    </Destinatario>
    <Texto>
      <Asunto>
        Este es mi documento con una estructura muy sencilla
        no contiene atributos ni entidades...
      </Asunto>
      <Parrafo>
        Este es mi documento con una estructura muy sencilla
        no contiene atributos ni entidades...
      </Parrafo>
    </Texto>
  </Mensaje>
</Edit_Mensaje>
```

Aquí está el ejemplo de código del DTD del documento «Edit_Mensaje.dtd»:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Este es el DTD de Edit_Mensaje -->

<!ELEMENT Mensaje (Remitente, Destinatario, Texto)*>
<!ELEMENT Remitente (Nombre, Mail)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Mail (#PCDATA)>
<!ELEMENT Destinatario (Nombre, Mail)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Mail (#PCDATA)>
<!ELEMENT Texto (Asunto, Parrafo)>
<!ELEMENT Asunto (#PCDATA)>
<!ELEMENT Parrafo (#PCDATA)>
```

Documentos XML bien formados y control de errores

Los documentos denominados como «bien formados» (del inglés *well formed*) son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, analizarse correctamente por cualquier analizador sintáctico (*parser*) que cumpla con la norma. Se separa esto del concepto de validez que se explica más adelante.

- Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.
- Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial.
- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.
- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.
- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.
- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos «entendibles» por las personas.

Partes de un documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento así como texto de etiquetas que contiene una gran variedad de efectos positivos o negativos en la referencia opcional a la que se refiere el documento, hay que tener mucho cuidado de esa parte de la gramática léxica para que se componga de manera uniforme.

- Prólogo

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

El prólogo de un documento XML contiene:

- Una declaración XML. Es la sentencia que declara al documento como un documento XML.
- Una declaración de tipo de documento. Enlaza el documento con su DTD

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

(definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.

- Uno o más comentarios e instrucciones de procesamiento.

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

- Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener solo un elemento raíz, característica indispensable también para que el documento esté bien formado. Sin embargo es necesaria la adquisición de datos para su buen funcionamiento.

EJEMPLO:

```
<Edit_Mensaje>
  (...)
</Edit_Mensaje>
```

Elementos

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

Atributos

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.

Por ejemplo, un elemento «estudiante» puede tener un atributo «Mario» y un atributo «tipo», con valores «come croquetas» y «taleno» respectivamente.

```
<Estudiante Mario="come croquetas" tipo="taleno">Esto es un día que Mario va  
paseando...</Estudiante>
```

Entidades predefinidas

Entidades para representar caracteres especiales para que, de esta forma, no sean interpretados como marcado en el procesador XML.

Ejemplo: entidad predefinida: & carácter: &.

Secciones CDATA

Artículo principal: *Anexo:Etiquetas HTML/XHTML*.

Es una construcción en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. No confundir con 2(#PCDATA) que es para los elementos. Permite que caracteres especiales no rompan la estructura. Ejemplo:

```
<![CDATA[ contenido especial: áéíóúñ&]] >
```

Comentarios

Comentarios a modo informativo para el programador que han de ser ignorados por el procesador. Los comentarios en XML tienen el siguiente formato:

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

```
<!-- Otro comentario -->
```

VALIDEZ

Que un documento esté «bien formado» solamente se refiere a su estructura sintáctica básica, es decir, que se componga de elementos, atributos y comentarios como XML especifica que se escriban. Ahora bien, cada aplicación de XML, es decir, cada lenguaje definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento.

Esta relación entre elementos se especifica en un documento externo o definición (expresada como DTD —*Document Type Definition*, 'Definición de Tipo de Documento'— o como XSchema). Crear una definición equivale a crear un nuevo lenguaje de marcado, para una aplicación específica.

Document Type Definition

La Document Type Definition o DTD (en español "definición de tipo de documento") define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD son denominados válidos.

- Declaraciones tipo elemento

Los elementos deben ajustarse a un tipo de documento declarado en una DTD para que el documento sea considerado como válido.

- Modelos de contenido

Un modelo de contenido es un patrón que establece los subelementos aceptados, y el orden en que se aceptan.

- Declaraciones de lista de atributos

Los atributos se usan para añadir información adicional a los elementos de un documento.

- Tipos de atributos
 - Atributos CDATA y NMTOKEN
 - Atributos enumerados y notaciones
 - Atributos ID e IDREF
- Declaración de entidades

XML hace referencia a objetos que no deben ser analizados sintácticamente según las reglas XML, mediante el uso de entidades. Las entidades pueden ser:

- Internas o externas
- Analizadas o no analizadas
- Generales o parametrizadas
- Espacios de nombres

Los espacios de nombres XML permiten separar semánticamente los elementos que forman un documento XML.

XML Schemas (XSD)

Un Schema es algo similar a un DTD. Define qué elementos puede contener un documento XML, cómo están organizados y qué atributos y de qué tipo pueden tener sus elementos.

- Ventajas de los Schemas frente a los DTD
 - Usan sintaxis de XML, al contrario que los DTD.
 - Permiten especificar los tipos de datos.
 - Son extensibles.

HERRAMIENTAS PARA TRABAJAR CON DOCUMENTOS XML

De hecho cualquier procesador de texto, que sea capaz de producir archivos .txt es

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar



capaz de generar XML, aunque en los entornos de desarrollo como Eclipse o Visual Studio, se facilita, ya que reconoce los formatos y ayuda a generar un XML bien formado.

LENGUAJES CREADOS USANDO XML

Extensible Stylesheet Language (XSL)

El Lenguaje de Hoja de Estilo Extensible (*eXtensible Stylesheet Language*, XSL) es una familia de lenguajes que permiten describir como los archivos codificados en xml serán formateados (para mostrarlos) o transformados. Hay tres lenguajes en esta familia: XSL Transformations (XSLT), XSL Formatting Objects (XSL-FO) y XML Path Language.

Lenguaje de enlace XML (XLINK)

XLink es una aplicación XML que intenta superar las limitaciones que tienen los enlaces de hipertexto en HTML. XLink 1.1 es ya una recomendación W3C.

EJEMPLOS

La siguiente explicación corresponde a los ejemplos adjuntos al material de la unidad que encontrarán en la carpeta ejemplos.

Existe una función en PHP llamada `simplexml_load_file` que convierte un archivo XML en un objeto del tipo `SimpleXMLElement`. (Mas información en el manual de PHP <http://php.net/manual/es/function.simplexml-load-file.php>)

En el archivo `verxmlalumno.php` lo que hacemos es tomar el archivo `alumnos.xml` convertirlo en objeto `SimpleXMLElement` a través de la función `simplexml_load_file` y luego imprimimos el contenido del objeto resultante para ver su estructura.

En el archivo `imprimirxmlalumno.php` lo que hacemos es acceder a los elementos del objeto e imprimirlos en pantalla.

De esta misma forma podemos trabajar con cualquier xml ofrecido en la web sobre cualquier tema en particular. Por ejemplo los sitios de noticias Clarin, La Nacion, InfoBae, por nombrar solo algunos ofrecen este tipo de archivos con información.

Tomemos uno cualquiera, por ejemplo:

<http://www.clarin.com/suplementos/arquitectura/ultimo/arquitectura.xml>

Haciendo uso de este xml podemos utilizar esa información proporcionada en el mismo para mostrarla en nuestro propio sitio, siendo actualizada periódicamente sin necesidad de modificación alguna. Lo que se va actualizando es el xml del que se nutre nuestro sitio.

En el script `verxmlclarin.php` lo que hacemos es evaluar el contenido del xml a utilizar y en `imprimirxmlclarin.php` lo imprimimos en pantalla de la misma manera que lo hacíamos con el xml de alumnos.

En la carpeta `crearxml` lo que mostramos es un ejemplo sencillo de como armar un xml a partir de una base de datos para que los demás consuman. Es decir que es el proceso inverso, desde nuestro sitio vamos a crear un archivo xml con información para que los demás puedan tomarla.

Para ello hacemos uso de las funciones del sistema de archivos para crear y escribir el archivo xml que vamos a generar.

Lo único que tenemos que tener en cuenta es que las etiquetas están bien formadas y que todo el archivo este codificado en utf-8, es por ello que hacemos uso de la función `utf8_encode` de PHP.



```
$out = '<xml version="1.0" encoding="utf-8">';
$alumno=new Alumno();
$alumnos=$alumno->getalumnos();
foreach($alumnos as $unalumno){
    $out .='<alumno>';
    $out .='<nombre>'.utf8_encode($unalumno['nombre']).'</nombre>';
    $out .='<apellido>'.utf8_encode($unalumno['apellido']).'</apellido>';
    $out .='<edad>'. $unalumno['edad'].'</edad>';
    $out .='<password>'. $unalumno['password'].'</password>';
    $out .='</alumno>';
}
$out .='</xml>';

$name_file="xml/Alumnos.xml";
$file=fopen($name_file,"w+");
fwrite ($file,$out);
fclose($file);

if (file_exists($name_file)){
    echo 'Se creo el archivo xml';
}
```