UNIVERSIDAD ABIERTA
INTERAMERICANA

Lenguaje de última generación
Facultad de tecnología

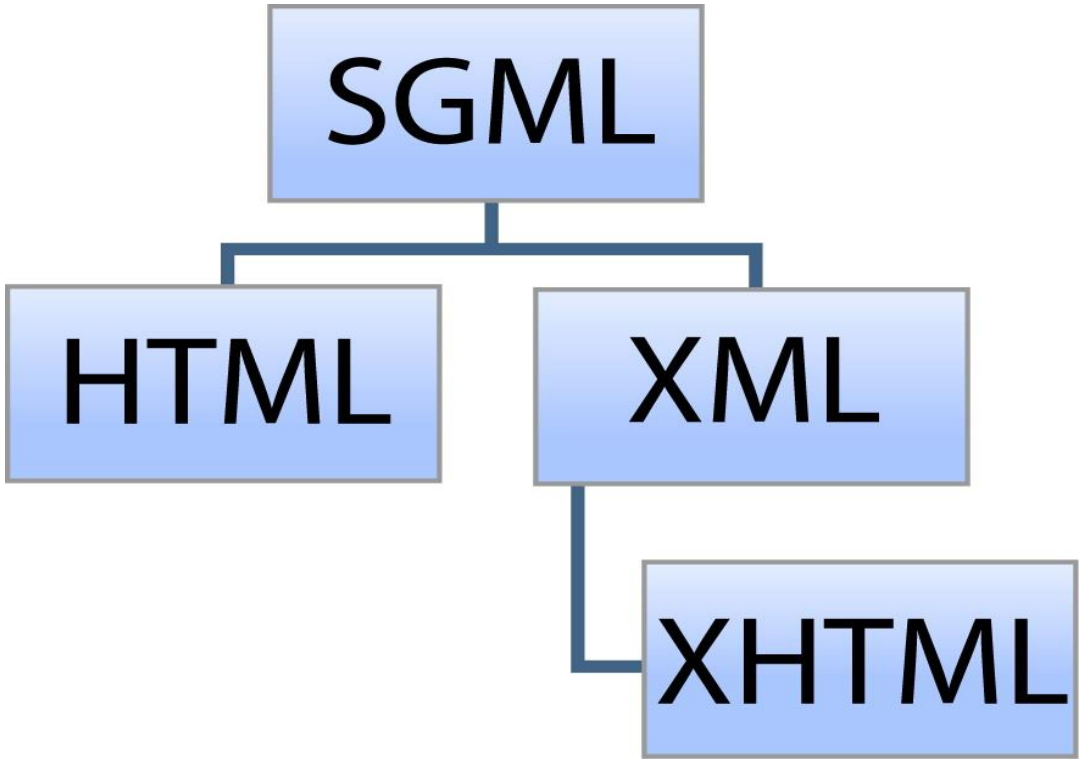
Clase - XML

Profesor Adjunto : Mauricio Prinzo

Agenda

- ✓ XML
  - Introducción
  - Utilidad
  - Ventajas
  - Características
  - Clasificación

¿Qué es XML?



¿Qué **NO** es XML?

- ✓ Un lenguaje de marcaje (markup).
  - # Porque es un estándar que especifica una sintaxis para crear lenguajes de marcado.
- ✓ Solo para Web.
  - # Puede ser usado para describir y comunicar cualquier información estructurada.
- ✓ Un invento de [x compañía].
  - # XML es un estándar creado por el W3C y soportado por compañías e instituciones de todo el mundo

Utilidad

- ✓ **XML permite representar información estructurada (en forma de documentos), de modo que pueda ser almacenada, transmitida, procesada, presentada e impresa por diferentes tipos de aplicaciones y dispositivos.**

Ventajas

- ✓ Es extensible
- ✓ Fácil de procesar/entender por software o por seres humanos
- ✓ Pensado para ser utilizado en cualquier lenguaje o alfabeto
- ✓ Separa radicalmente la información o contenido, de su presentación o formato
- ✓ Un documento en XML, puede tener varias formas de presentación (pdf, ppt, html, wml. Rtf, etc)

Características de XML

- ✓ **Permite la creación de etiquetas propias y permite asignar atributos a las etiquetas**
- ✓ **Trabaja con los llamados DTDs (Definición de Tipo de Documento)**
- ✓ **Permite la creación de nuevos DTDs.**
- ✓ **Es case sensitive**

Tipos de Archivos

- ✓ **Asociado a XML:**
  - Datos -> **XML 1.0**
  - Definición de tipo de documentos -> **DTD**
  - Definición de estilos -> **XSL = XSLT+Xpath**

Elementos

- ✓ Los **Elementos** se pueden definir como bloques de construcción de un XML, pueden comportarse como contenedores para texto, atributos o ambos.

- ✓ Cada documento XML contiene uno o más elementos, el alcance de lo que son o bien delimitado por las etiquetas inicial y final, o de elementos vacíos, de una etiqueta de elemento vacío.

Documentos

- ✓ **Los documentos XML tienen una estructura lógica y una física.**

  - **La estructura lógica formada por declaraciones, elementos, comentarios, referencias de caracteres e instrucciones de procesamiento que se indican en el documento mediante marcas concretas.**

  - **La estructura física formado por una serie de unidades llamadas entidades, comenzando por la raíz, que indican los datos que contendrá el documento.**

Descripciones

- ✓ **Los componentes de marcado XML que se soportan en XML 1.0 son**

  - **Etiquetas de elemento o nodos**
  - **Referencias de entidades**
  - **Atributos**
  - **Instrucciones de procesamiento**
  - **Declaraciones de tipo documento**
  - **Comentarios**

Etiquetas

- ✓ **Las etiquetas constituyen el componente más evidente de la sintaxis XML y se emplean para describir elementos**
- ✓ **Los elementos están delimitados por ángulos (<,>) e identifican el contenido que delimitan.**
- ✓ **Los elementos XML pueden ser vacíos, y las etiquetas de estos deben usar la barra (/) tras el nombre del elemento, que indica que están vacías.**

Etiquetas - Reglas

- ✓ **Para nombrar a los elementos hay ciertas reglas**

- **No pueden comenzar con xml**
- **Siempre deben comenzar con una letra**
- **Xml es case sentitive**

Etiquetas (I)

- ✓ Referencia a entidades

- Según Morrison: **Son bloques de construcción de XML**
- Según Gallardo: **Caracteres especiales llamados entidades**

- ✓ Las referencia a entidades comienzan con "&" y acaban con ";".

Entidad	Carácter
<	< (menor)
>	> (mayor)
&	& (ampersand)
'	' (apóstrofe)
"	“ (dobles comillas)

Ejemplo Referencia a Entidades

```
<?xml version="1.0" encoding="UTF-8"?>
<entidades>
  <menor_que>&amp;lt;</menor_que>
  <mayor_que>&amp;gt;</mayor_que>
  <comilla_doble>&amp;quot;</comilla_doble>
 <comilla_simple>&amp;apos;</comilla_simple>
  <ampersand>&amp;amp;</ampersand>
</entidades>
```

```
<?xml version="1.0" encoding="UTF-8"?>
- <entidades>
    <menor_que><</menor_que>
    <mayor_que>></mayor_que>
    <comilla_doble>"</comilla_doble>
    <comilla_simple>'</comilla_simple>
    <ampersand>&</ampersand>
  </entidades>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Companias>
  <compania>D&amp;apos;Angelo Servicios </compania>
 <compania>Rosenber&amp;apos;s Shoes &amp;amp;
Glasses</compania>
</Companias>
```

```
<?xml version="1.0" encoding="UTF-8"?>
- <Companias>
    <compania>D'Angelo Servicios </compania>
    <compania>Rosenber's Shoes & Glasses</compania>
  </Companias>
```

Etiquetas (II)

- ✓ Atributos

- Pueden aprovecharse para incluir datos.
- Se colocan siempre en la etiqueta de apertura
- Siempre se ponen comillas

- ✓ Ejemplo: **<elemento atributo="valor">**

- ✓ Ejemplo2: **<gato raza="Persa">Micifú</gato>**
  <gato raza="Persa" nombre="Micifú"/>

Etiqueta – Estructura Elemento

Etiquetas (III)

- ✓ Instrucciones de procesamiento
- ✓ Se utilizan para proporcionar información en un documento XML
- ✓ Siempre comienzan con un signo menor que y un signo de interrogación **<?** y terminan con un signo de interrogación y un signo mayor que **?>**
- ✓ Ejemplo de Prólogo:

  <?xml version=”1.0” ?>
- ✓ Además de la versión XML, la declaración puede describir el estado autónomo (yes/no), como también el encoding

  <?xml version=”1.0” standalone=yes ?>

  <?xml version=”1.0” standalone=no ?>

  <!DOCTYPE anuncio SYSTEM “anuncio.dtd“>

  <!DOCTYPE libro SYSTEM "http://www.pruebasxml/libro.dtd">

Etiquetas (IV)

- ✓ **Comentarios**
- ✓ **Los comentarios comienzan con <!-- y finalizan con -- >**
- ✓ **Ejemplo <!--Comentario -->**

Sección CDATA

- ✓ El uso de las secciones CDATA se utilizan para marcar que parte del código el analizador debe ignorar

- ✓ Se define una sección de código CDATA englobándola entre las cadenas **<![CDATA[y]]>**

- ✓ EJemplo:

  <![CDATA[<nombre>jose Lopez</nombre> <direccion>San juan 2500</direccion>]]>

  <ciudad>Capital Federal</ciudad><provincia>Buenos Aires</provincia>

  Donde nombre y dirección no serán analizados sintácticamente

Ejemplo CDATA

```
<?xml version="1.0" encoding="UTF-8"?>
<Datos>
    <![CDATA[<nombre>jose Lopez</nombre>
     <direccion>San juan 2500</direccion>]]>
    <ciudad>Capital Federal</ciudad>
    <provincia>Buenos Aires</provincia>
</Datos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
- <Datos>
  - <![CDATA[
        <nombre>jose Lopez</nombre>
         <direccion>San juan 2500</direccion>
    ]]>
    <ciudad>Capital Federal</ciudad>
    <provincia>Buenos Aires</provincia>
</Datos>
```

Juego de Caracteres

Una tabla o juego de caracteres es una lista de los caracteres que está permitido utilizar (Gallardo).

Sin embargo no utilizamos la tabla ASCII por defecto, en el caso de los archivos XML utiliza UTF-8 "Unicode Trasnformation Formal"

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

- Utf-8(unicode)
- Utf-16(unicode)
- ISO-10646-UCS-2 (Unicode)
- UNICODE-1-1-UTF-8
- UNICODE-2-0-UTF-16
- UNICODE-2-0-UTF-8
- ISO-106446-ucs-4(unicode)
- ISO-8859-1 a ISO-8859-9
- WINDOWS-1250 - WINDOWS-1258
- Shift_jis(jis x-0208-1997)

Ejemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <Biblioteca>
  - <Documento>
      <Titulo>Ingenieria del software: un enfoque prÃictico</Titulo>
      <Edicion>3a. ed</Edicion>
      <Paginas>824</Paginas>
      <Editorial>McGraw-Hill</Editorial>
      <Autor>Pressman , Roger S.</Autor>
    </Documento>
</Biblioteca>
```

```
<?xml version="1.0" encoding="SHIFT_JIS"?>
- <Biblioteca>
  - <Documento>
      <Titulo>Ingenieria del software: un enfoque prｽctico</Titulo>
      <Edicion>3a. ed</Edicion>
      <Paginas>824</Paginas>
      <Editorial>McGraw-Hill</Editorial>
      <Autor>Pressman , Roger S.</Autor>
    </Documento>
</Biblioteca>
```

```
<?xml version="1.0" encoding="UNICODE-2-0-UTF-8"?>
- <Biblioteca>
  - <Documento>
      <Titulo>Ingenieria del software: un enfoque práctico</Titulo>
      <Edicion>3a. ed</Edicion>
      <Paginas>824</Paginas>
      <Editorial>McGraw-Hill</Editorial>
      <Autor>Pressman , Roger S.</Autor>
    </Documento>
</Biblioteca>
```

```
<?xml version="1.0"?>
- <Biblioteca>
  - <Documento>
      <Titulo>Ingenieria del software: un enfoque práctico</Titulo>
      <Edicion>3a. ed</Edicion>
      <Paginas>824</Paginas>
      <Editorial>McGraw-Hill</Editorial>
      <Autor>Pressman , Roger S.</Autor>
    </Documento>
</Biblioteca>
```

Ejemplo XML

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

<Biblioteca>
    <Documento>
        <Titulo>Ingenieria del software: un enfoque práctico</Titulo>
        <Edicion>3a. ed</Edicion>
        <Paginas>824</Paginas>
        <Editorial>McGraw-Hill</Editorial>
        <Autor>Pressman , Roger S.</Autor>
     </Documento>
</Biblioteca>
```

¿Cómo validamos un archivo XML?

- ✓ Usando un DTD Interno
- ✓ Usando un DTD Externo
- ✓ Usando Schema

DTD

- ✓ La definición de tipo de documento (DTD) es una descripción de estructura y sintaxis de un documento XML.
- ✓ Una DTD describe:

- ⌗ **Elementos:** indican qué etiquetas son permitidas y el contenido de dichas etiquetas.
- ⌗ **Estructura:** indica el orden en qué van las etiquetas en el documento.
- ⌗ **Anidamiento:** indica qué etiquetas van dentro de otras.

- ✓ Ejemplo: **<! DOCTYPE agenda SYSTEM “agenda.dtd”>**

DTD II

- ✓ Describen las reglas para validar un archivo XML. En pocas palabras, permiten confrontar la estructura del contenido de un archivo XML con el acuerdo entre las partes.

- ✓ Los **nodos** se describen con una palabra reservada llamada ELEMENT.

- ✓ Los elementos están formados por un nombre y una **regla**, respetando la sintaxis ELEMENT nombre (regla). Las reglas pueden ser los nodos que contiene o el valor

DTD III – Declaración de Elementos

- ✓ **Los elementos se declaran en una DTD, de la forma:**

  <!ELEMENT ElementName Type>

- ✓ **XML soporta cuatro tipos de elementos:**
  - EMPTYT (Vacío): El elemento no tiene contenido, pero puede contener atributos.

    <!ELEMENT ElementName EMPTY>
  - Sólo elementos: El elemento solo contiene elementos secundarios.

    <!ELEMENT ElementName ContentModel>
  - Cada subelemento puede llevar, además, los siguientes símbolos:
    Sin símbolo: El elemento secundario debe aparecer una sola vez.
    Interrogación (?): Puede aparecer una o ninguna vez.(0 o 1)
    Asterisco (*): Puede aparecer cualquier número de veces. (0 a n)
    Signo más (+): Debe aparecer por lo menos una vez. (1 a n)

    <!ELEMENT planeta (nombre+, tipo, (orbita|satélite+), composición*)>
  - Mixto: El elementos contiene combinación de elementos secundarios y datos de caracteres.

    <!ELEMENT ElementName (#PCDATA | ElementList)*>
  - ANY: El elemento contiene cualquier cosa que permita la DTD

    <! ELEMENT ElementName ANY>. .

PCDATA

- ✓ **PCDATA**: Son datos de caracteres analizados sintácticamente.
  - Ejemplo:*<!ELEMENT telefono (#PCDATA)>*

Ejemplo DTD

```
<? xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE socios [
  <!ELEMENT socios (socio)*>  <!-- * 0 a n -->
  <!ELEMENT socio ((numero | clave)?, nombre, edad?)> <!-- ? 0 a 1 -->
  <!ELEMENT clave (#PCDATA)>
  <!ELEMENT edad (#PCDATA)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT numero (#PCDATA)>
]>

<socios>
  <socio>
    <nombre>Ana</nombre>
    <edad>21</edad>
  </socio>
  <socio>
    <numero>598</numero>
    <nombre>Pedro</nombre>
    <edad>19</edad>
  </socio>
  <socio>
    <clave>hY75Du</clave>
    <nombre>jose</nombre>
  </socio>
</socios>
```

Ejemplo DTD Interna

```
?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE Direccion [
  <!ELEMENT Direcciones (Direccion*)>
  <!ELEMENT Direccion (Nombre,Empresa,Movil)>
  <!ELEMENT Nombre (#PCDATA)>
  <!ELEMENT Empresa (#PCDATA)>
  <!ELEMENT Movil (#PCDATA)>
]>

<Direcciones>
<Direccion>
  <Nombre>Juan Perez</Nombre>
  <Empresa>Cualquiera S.A</Empresa>
  <Movil>(015) 123-4567</Movil>
</Direccion>
<Direccion>
  <Nombre>Lola</Nombre>
  <Empresa>Pedriel S.A</Empresa>
  <Movil>(0221)777889</Movil>
</Direccion>
</Direcciones>
```

EjemploII DTD Externa

Archivo Direccion.DTD

```
<!ELEMENT Direcciones (Direccion*)>
 <!ELEMENT Direccion (Nombre,Empresa,Movil)>
 <!ELEMENT Nombre (#PCDATA)>
 <!ELEMENT Empresa (#PCDATA)>
 <!ELEMENT Movil (#PCDATA)>
```

Archivo XML

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

<Direcciones>
<Direccion>
  <Nombre>Juan Perez</Nombre>
  <Empresa>Cualquiera S.A</Empresa>
  <Movil>(015) 123-4567</Movil>
</Direccion>
</Direcciones>
```

Xml Bien Formados

- si no se utiliza DTD, el documento debe comenzar con un Declaración de Documento Standalone
- todas las etiquetas deben estar balanceadas
- todos los valores de los atributos deben ir entrecomillados .
- cualquier elemento VACÍO deben terminar con '/>‘.
- no debe haber etiquetas aisladas (< ó &) en el texto .
- los elementos deben anidar dentro de sí sus propiedades .
- Los ficheros bien-formados sin-DTD pueden utilizar atributos en sus elementos, pero éstos deben ser todos del tipo CDATA, por defecto. El tipo CDATA (characterDATA) son caracteres..

Según la especificación de XML del W3C, un documento XML está bien formado si:

- Tomado como un todo, cumple la regla denominada "document".
- Respeta todas las restricciones de buena formación dadas en la especificicación.
- Cada una de las entidades analizadas que se referencia directa o indirectamente en el documento está bien formada.

Cumplir la regla "document" antes mencionada significa:

- Que contiene uno o más elementos.
- Hay exactamente un elemento, llamado raíz, o elemento documento, del cual ninguna parte aparece en el contenido de ningún otro elemento.
- Para el resto de elementos, si la etiqueta de comienzo está en el contenido de algún otro elemento, la etiqueta de fin está en el contenido del mismo elemento. Es decir, los elementos delimitados por etiquetas de principio y final se anidan adecuadamente mutuamente.

Analizadores

- ✓ Para comprobar si un documento está bien formado

- ✓ Un cliente Web
  - IE
- ✓ Un servicio de comprobación
  - http://validator.w3.org/

SChema

- ✓ Es Conjunto de reglas que sirven para forzar la estructura y la articulación del conjunto de documentos XML.

- ✓ **W3C**

  "XML Schema" (Esquema XML) es el nombre oficial otorgado a la recomendación del W3C, que elaboró el primer lenguaje de esquema separado de XML (la definición de tipo de documentos (DTD) forma parte de XML)

Ventajas SChema

- ✓ Se basa en XML
- ✓ Se puede Administrar como XML
- ✓ Soporta tipos de datos
- ✓ Es un modelo abierto
- ✓ Soporta espacios de nombres
- ✓ Soporta grupos de atributos y permite combinarlos

SChema

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Libro">
        <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="Título"  type="xsd:string"/>
                  <xsd:element name="Autores" type="xsd:string"
maxOccurs="10"/>
                  <xsd:element name="Editorial" type="xsd:string"/>
             </xsd:sequence>
          <xsd:attribute name="precio" ype="xsd:double"/>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

DTD y SChema

DTD	XML Schema
Basadas en una sintaxis especializada.	Basados en XML
Compactas	No compactas
Hay muchas herramientas disponibles para procesar documentos XML con las DTD	Hay muy pocas herramientas para los esquemas XML
Tratan todos los datos como cadenas o cadenas enumeradas	Soportan una serie de tipos de datos (int, flota, bolean, date, etc.)
Emplean un modelo de datos cerrados que no permite prácticamente la ampliación.	Presentan un modelo de datos abierto, lo cual permite ampliar los vocabularios y establecer relaciones de herencia entre los elementos sin invalidar a los documentos.
Sólo permiten una asociación entre un documento y su DTD a través de la declaración de tipos de documentos.	Soportan la integración de los espacios de nombres, lo que permite asociar nodos individuales en un documento con las declaraciones de tipos de un esquema.
Soportan una forma primitiva de agrupación a través de entidades de parámetros, lo que supone una gran limitación, ya que el procesador XML no sabe nada acerca del agrupamiento.	Soportan los grupos de atributos, que le permiten combinar atributos lógicamente.

UNIVERSIDAD ABIERTA INTERAMERICANA

XML II Parte

XML y VS : Espacio de Nombres

- ✓ Todas las funcionalidades de XML en el framework están agrupadas en:

  - **System.XML**

Manejo de archivos XML

- ✓ **XmlTextReader**
  - Funciona igual que el DataReader
  - Es menos potente que un analizador DOM
  - Nunca carga el contenido completo del archivo XML , la aplicación es la que se ocupa de extraer los datos de la secuencia XML

XmlTextReader (C#)

- ✓ **Creación del Objeto**

```
XmlTextReader ArchivoXML = new XmlTextReader("MiArchivo.Xml");
```

¿Cómo leer el contenido del archivo XML?

```
while (xmlText.Read) {
        //Código  de trabajo;
    }
```

- ✓ **Algunas Sobrecargas del Constructor**
  - Recupera el archivo desde una dirección url.
  - Permite cargar el texto del archivo

Identificación de nodos

- ✓ El resultado que se obtiene en la variable que abre el archivo, devuelve una lista de nodos.
- ✓ Que se acceden por medio de XmlNodeType
  - Document
  - Element
  - EndElement
  - Text
  - CData

Demo

Lectura de un archivo XML

XmlTextWriter (C#)

- ✓ Escribe un archivo XML
- ✓ Asegura que la salida es XML bien formado

```
XmlTextWriter archivoxml =  new XmlTextWriter("ejemplo2.xml",
System.Text.Encoding.UTF8);
```

Crear nodos (C#)

```
archivoxml.Formatting = Formatting.Indented;
archivoxml.Indentation = 2;
archivoxml.WriteStartDocument(true);
archivoxml.WriteStartElement("Alumno");
archivoxml.WriteAttributeString("DNI", 1);
archivoxml.WriteElementString("Apellido", "Perez");
archivoxml.WriteEndElement();
archivoxml.WriteEndDocument();
archivoxml.Close();
```

- Tabula el contenido
- Instrucción de procesamiento
- Abre elemento
- Crea un Atributo
- Crea un Nodo
- Cierre el elemento, el documento y el archivo

XPath

- ✓ **XPATH permite navegar a través de los elementos y atributos de un documento XML.**

Path absoluto es aquel donde se especifica el camino en forma completa arrancando desde el nodo Raiz.

Path relativo toma la posición donde se encuentra y completa el Path para llegar al nodo.

Demo#2

CREAR un archivo XML

DOM XML (C#)

- ✓ Permite un manejo más profundo de los archivos XML
- ✓ Permite agregar y borrar nodos
- ✓ Cada nodo se representa con un objeto
- ✓ Utiliza la clase XmlNode y Xmldocument

```
XmlDocument  doc = new XmlDocument();
doc.Load("Ejemplo.xml")
```

DOM - Nodos

- ✓ A medida que el contenido XML se lee en el DOM las partes se traducen en **nodos** que mantienen metadatos adicionales acerca de sí mismos, como su tipo y valores.
- ✓ Cuando se leen varios datos, se asigna a cada nodo un tipo. Esto es importante ya que no todos los nodos son del mismo tipo y el tipo determina las características y funcionalidad del nodo (qué acciones pueden realizarse y qué propiedades pueden establecerse o recuperarse).

Tipos de Nodo

Tipo de nodo DOM	Objeto (clase .NET)	Descripción
Document	XmlDocument	Contenedor de todos los nodos del árbol. También se conoce como la raíz del documento, que no siempre coincide con el elemento raíz.
Element	XmlElement	Representa un nodo de elemento.
Attr	XmlAttribute	Atributo de un elemento.
Comment	XmlComment	Nodo de comentario.
Text	XmlText	Texto que pertenece a un elemento o atributo.
CDATASection	XmlCDataSection	Representa CDATA.

DOM

Para recorrer nodos:

- ✓ Se usa una estructura ForEach
- ✓ La variable es de tipo XmlNode
- ✓ La propiedad HasChildNodes permite saber si tiene nodos hijos
- ✓ Attributes devuelve una lista de atributos
- ✓ innerTexte innerXml devuelve el contenido del nodo en sus respectivos formatos

Para borrar Nodos:

- ✓ **RemoveAll:** Borra todos los nodos
- ✓ **RemoveChild**: Borra un nodo
- ✓ **Attributes.Remove**: Borra un atributo con un nombre especifico

Para Agregar Nodos:

- ✓ **AppendChild**: Agrega un Elemento de tipo XmlElement

Explorando nodos (C#)

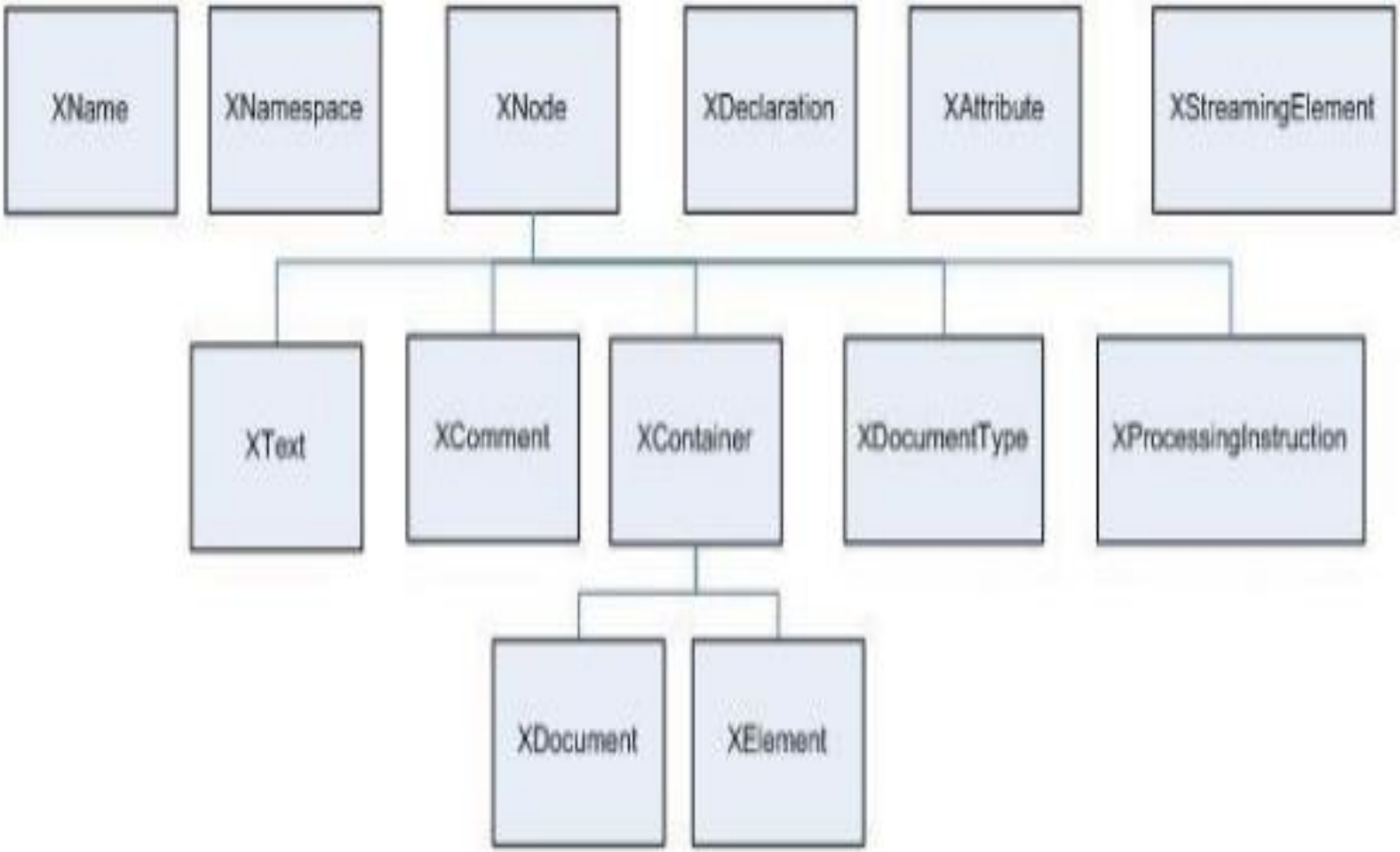
```
XmlDocument doc = new XmlDocument();
doc.Load("Ejemplo.xml");

foreach (XmlNode f in doc.DocumentElement) {
    foreach (XmlNode a in f.Attributes) {
        this.Arbol.Nodes.Add(a.InnerText);
    }

    foreach (XmlNode ff in f.ChildNodes) {
        this.Arbol.Nodes.Add(ff.InnerText);
    }

}
```

LinQ to XML

- ✓ **“LINQ a XML” no es otra cosa que proporcionar toda la potencialidad del “Lenguaje Integrado de Consultas” aplicado a la información que se encuentra contenida en XML**
- ✓ **Permite leer y modificar estos archivos de forma parecida a como lo hacíamos a través de XMLReader y XMLWritter.**
- ✓ **Notable mejora en el consumo de memoria y la cantidad de código que debemos escribir.**
- ✓ **Utiliza la librería**

  System.Xml.Linq

LinQ to XML - Jerarquías



LinQ to XML - Ventajas

- ✓ **Elementos principales**.- Los elementos básicos para trabajar con formatos XML a nuestro criterio son XElement y XAttribute.

- ✓ **Visión simplificada de los archivos XML:** se puede tratar tanto al documento como a los nodos que lo conforman como un Xelement

- ✓ **Proporciona una identificación a los nodos de los archivos XML**.- Podemos dar un nombre a cada elemento de nuestro árbol XML a través de la propiedad XName, esta utilidad será de gran ayuda al momento de realizar copias, reemplazos o al renombrar los Nodos.

- ✓ **Creación de estructuras XML.-:** Utilizando XElement o Xattributte

- ✓ **Memoria**.- Mejor uso de los recursos de memoria, lo que ayudará a mejorar el rendimiento de nuestras aplicaciones.

Forma Tradicional - Escritura

```
XmlDocument DocumentoXML = new XmlDocument();
    XmlElement nodPersona = DocumentoXML.CreateElement("Persona");
    DocumentoXML.AppendChild(nodPersona);
    XmlElement nodContacto = DocumentoXML.CreateElement("Contacto1");
    nodPersona.AppendChild(nodContacto);
    XmlElement nodCedula = DocumentoXML.CreateElement("Cédula");
    nodCedula.InnerText = "0128928212";
    nodContacto.AppendChild(nodCedula);
    XmlElement nodNombre = DocumentoXML.CreateElement("Nombre");
    nodNombre.InnerText = "Juan Carlos Pérez Maldonado";
    nodContacto.AppendChild(nodNombre);
    XmlElement nodDireccion = DocumentoXML.CreateElement("Dirección");
    nodDireccion.InnerText = "Av. De las Américas y Batán";
    nodContacto.AppendChild(nodDireccion);
            XmlElement nodTelefonos = DocumentoXML.CreateElement("Teléfonos");
            Contacto.AppendChild(nodTelefonos);
            XmlElement nodTelCasa = DocumentoXML.CreateElement("Convencional1");
            nodTelCasa.SetAttribute("Hogar", "2890654");
            nodTelefonos.AppendChild(nodTelCasa);
            XmlElement nodTelTrabajo = DocumentoXML.CreateElement("Convencional2");
            nodTelTrabajo.SetAttribute("Oficina", "2847241");
            nodTelTrabajo.SetAttribute("Desde", "09H00 AM");
            nodTelTrabajo.SetAttribute("Hasta", "19h00 PM");
            nodTelefonos.AppendChild(nodTelTrabajo);
            XmlElement nodTelCelular = DocumentoXML.CreateElement("Celular");
            nodTelCelular.SetAttribute("Celular", "098213125");
            nodTelefonos.AppendChild(nodTelCelular);
  DocumentoXML.Save("d:\\SinLINQ.xml");
```

Usando LinQ - Escritura

```
XDocument xmlDoc = XDocument.Load("Juegos.XML");

xmlDoc.Element("juegos").Add(new XElement("juego",
                        new XAttribute("id", this.txtid.Text.Trim()),
                        new XElement("nombre", txtNom.Text.Trim()),
                        new XElement("genero", txtGenero.Text.Trim()),
                        new XElement("plataforma", txtPlataforma.Text.Trim()),
                        new XElement("companiaCreadora", txtempresa.Text.Trim())));

xmlDoc.Save("Juegos.XML");
```

LinQ to XML - Lectura

```
public List<Juego> LeerXML2()
{
    //Ponemos la direccion del archivo y el nombre de los elementos que queremos obtener
    var consulta =
        from juego in XElement.Load("Juegos.XML").Elements("juego")
        select new Juego
        {
            IdJuego = Convert.ToInt32(Convert.ToString(juego.Attribute("id").Value).Trim()),
            NombreJuego = Convert.ToString(juego.Element("nombre").Value).Trim(),
            GeneroJuego = Convert.ToString(juego.Element("genero").Value).Trim(),
            PlataformaJuego = Convert.ToString(juego.Element("plataforma").Value).Trim(),
            CompaniaCreadora = Convert.ToString(juego.Element("companiaCreadora").Value).Trim()
        };//Fin de consulta.
    List<Juego> Lstjuegos = consulta.ToList<Juego>();
    return Lstjuegos;
}//Fin de metodo leerXML.
```

¿Preguntas?

Autoevaluación

✓ Responda las siguientes preguntas :

- ¿Porque es extensible?

HTML funciona con etiquetas predefinidas como <\p>, <\h1>, <\table>, etc. El lenguaje XML no tiene etiquetas predefinidas. Las etiquetas no están definidas en ningún estándar XML. Las etiquetas son "inventadas" por el autor del documento XML. Con XML, el autor debe definir tanto las etiquetas como la estructura del documento.

- ¿Cuándo utilizaría CData?

El uso de las secciones CDATA se utilizan para marcar que parte del código el analizador debe ignorar.

- ¿Cómo se valida un archivo XML?

*) Usando un DTD Interno
*) Usando un DTD Externo
*) Usando Schema

- ¿Cuándo usaría DOM?

Cuando necesite leer, manipular y modificar un documento XML mediante programación

BIBLIOGRAFÍA

- Balena,Francesco. Programación avanzada con Microsoft Visual Basic.Net.-- México, DF: McGraw-Hill; c2003.
  Capitulo 20,21 y 22
- ✓ Troelsen, Andrew. Pro C# 5.0 and Net 4.5. FrameWork. 6a. ed.—New york, DF : Apress, 2012.
- ✓ Jérôme Hugon. C# 5. Desarrolle Aplicaciones Windows Con Visual Studio 2013.— España: Eni.2013

Otro libros recomendados

- ✓ Fraguas Berasain,Santiago;Morrison,Michael. XML : al descubierto .Madrid: Prentice Hall c2000. 899 páginas *Disponibilidad: Ubicación:*CD 64 ej.1 - *Inventario* 039841/ *Ubicación:*005.13 MOR ej.1 - *Inventario* 039834
- ✓ Marchal,Benoit;Vidal Romero Elizondo,Alfonso. XML con ejemplos .México, DF: Pearson Educación de México c.2001. 504 páginas. *Ubicación:*005.13 MAR ej.2 - *Inventario* 052243/ *Ubicación:*005.13 MAR ej.1 - *Inventario* 052242
- ✓ Castro,Elizabeth. Guía de aprendizaje XML .Madrid: Pearson Educación c2001. 264 páginas *Ubicación:*005.13 CASx - *Inventario* 047170

URL

- ✓ http://www.w3c.org 2005-08-23
- ✓ http://www.w3.org/TR/CSS1 2005-08-23