Escuela de Sistemas y Tecnologías

Transparencias de **ANALISTA DE SISTEMAS**

Edición 2017 – Materia: **Aplicaciones Distribuidas**

TEMA: XML

Plantel y Contactos

- **Bedelía:**
  - Mail: bedeliasistemas@bios.edu.uy
- **Encargado de Sucursal:**
  - Pablo Castaño
  - Mail: pablocasta@bios.edu.uy

Recursos

- **Recursos Imprescindibles:**
  - Sitio Web de material (comunicarse con Bedelía por usuario/contraseña).
  - Transparencias del Curso.
  - Contar con el software necesario

Agenda

- Introducción a XML
- Ado .Net y XML
- Clases Básicas
- LinQ to XML
- XSLT

Introducción a XML

¿Qué es XML?

- e**X**tensible **M**arkup **L**anguage (lenguaje extensible de marcas).
- Se basa en una serie de recomendaciones publicadas por los grupos de trabajo W3C. Por lo tanto es apto, aunque no se ha limitado solo a esto, para utilizarse en aplicaciones basadas en Web.
- Es un formato de texto flexible, que se puede utilizar como base, para crear nuevos lenguajes de marcas. Estos podrán usarse en la publicación de documentos e intercambio de datos.

Ventajas

- **Abierto**: Permite crear nuevos elementos (nodos).
- **Extensible**: En cualquier momento se puede extender, agregando más elementos.
- **No requiere de un compilador**: No fue creado para tener que ser traducido a lenguaje de máquina.
- **Interoperable**: Es un lenguaje utilizado para el intercambio de datos entre distintas aplicaciones y/o plataformas (ya que en realidad sus archivos son de texto plano).

Características que debe cumplir (1)

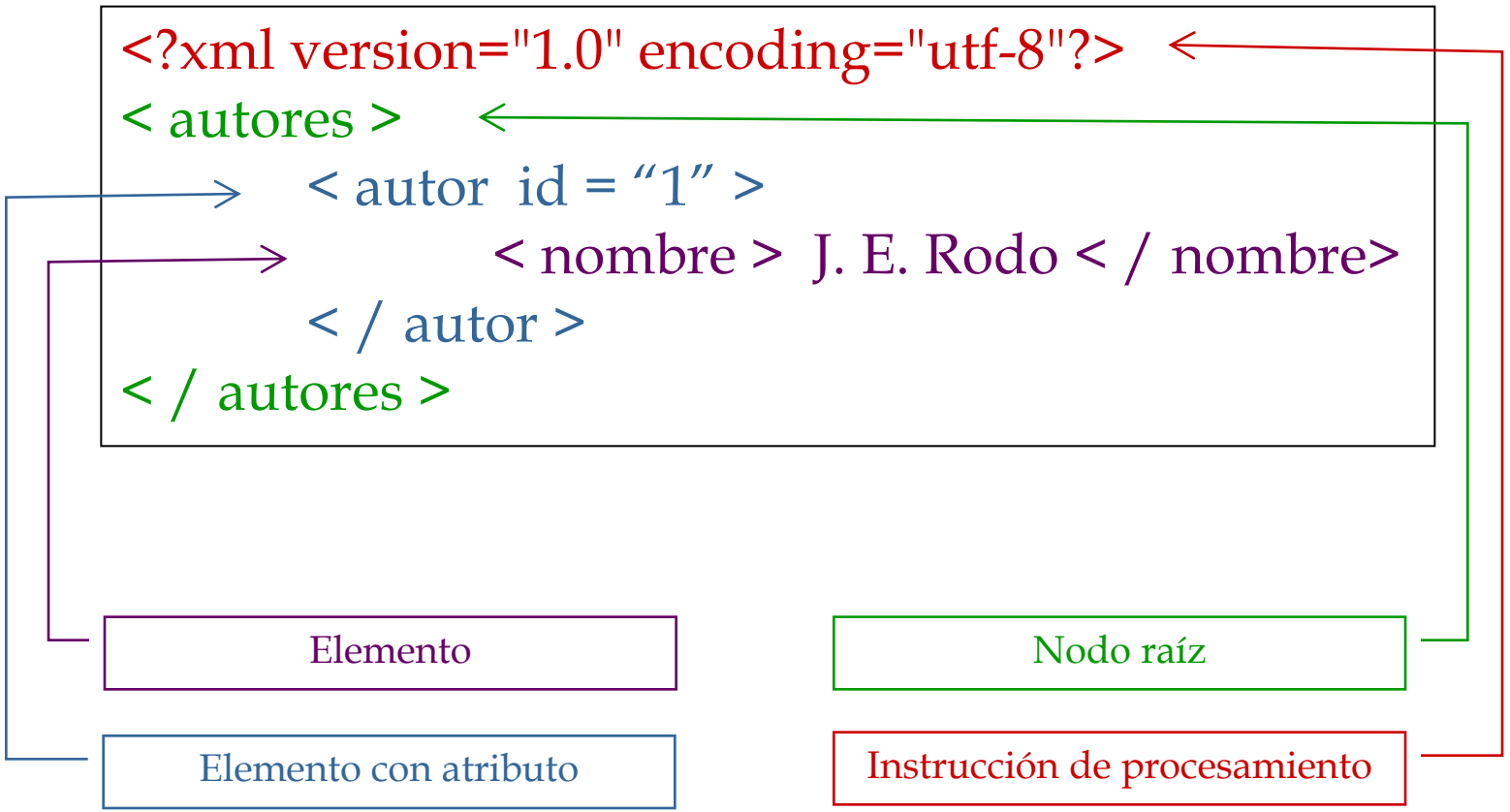
- **Bien formado**: Los documentos XML deben estar bien formados, cumpliendo con el estándar de la W3C.
- **Estructura jerárquica**: Los documentos deben seguir una estructura jerárquica de anidación.
- **Elemento raíz**: Sólo puede existir un elemento raíz (*root*) en cada documento XML.
- **Texto plano**: Se pueden crear en cualquier editor de texto, siempre que se guarde como archivo de sólo texto (sin formato).

Características que debe cumplir (2)

- Todos los elementos secundarios están anidados correctamente uno dentro de otro.
- Es *case sensitive* con el nombre de las etiquetas.

Características que debe cumplir (3)

- Formato Básico



XML en 9 puntos (1)

1. **Estructura de datos**: Conjunto de reglas para diseñar etiquetas que permiten estructurar los datos.
2. **Etiquetas**: Usa etiquetas del formato

   ```
   <nodo> valor </nodo>
   ```

   las cuales pueden tener atributos. Estas son utilizadas en la estructura de datos.
3. **Es simplemente texto**: Permite su edición, y está pensado para ser procesado por sistemas computacionales.

XML en 9 puntos (2)

4. **Es *verbose***: Los archivos XML son más grandes que los binarios. Ocupan más espacio en disco, pero existen técnicas de compresión que facilitan su manejo.
5. **Familia de tecnologías**: XML es el lenguaje estándar para definir nodos y atributos. Pero existen un conjunto de tecnologías para trabajar con documentos XML: XSLT, Esquemas XSD, XPATH, entre otros.
6. **Es relativamente nuevo**: Se desarrolló en 1996 y fue adoptado por la W3C en 1998.

XML en 9 puntos (3)

7. **Es modular**: Permite combinar dos documentos XML y formar un tercero.
8. **Es la base para otras tecnologías**: Muchas tecnologías se basan en XML para su funcionamiento (por ejemplo los Servicios Web).
9. **Es gratis**: Cualquier puede usar XML.

Ado .Net y XML

DataSet (1)

- Estructura:
  - Un objeto **DataSet** puede leer / escribir datos y esquemas como documentos XML. Los datos y esquemas pueden transportarse, a continuación, a través de HTTP y cualquier aplicación puede utilizarlos en cualquier plataforma que sea compatible con XML

DataSet (2)

- **Operaciones**:
  - **ReadXml(*"NombreArchivo" [, XmlReadMode]*) -** lee un documento XML que incluya esquema y datos; cargando el objeto DataSet con dicha información.
  - **WriteXmlSchema(*"NombreArchivo"*) –** guarda en archivo XML únicamente el esquema de los DataTables contenidos
  - **WriteXml (*"NombreArchivo" [, XmlWriteMode]*) –** guarda en archivo XML el esquema los datos de los DataTables contenidos.

Clases Básicas

XmlDocument (1)

- Permite que los documentos XML sean almacenados, recuperados y manipulados.
- **Propiedades Básicas:**
  - ***DocumentElement*** – obtiene el nodo Raíz del documento. Devuelve un objeto XmlNode
- **Métodos Básicos:**
  - ***Load*** *("nombreArchivo")* – carga en memoria el contenido de un archivo Xml bien formado
  - ***Save*** *("nombreArchivo")* – graba el contenido del objeto en un archivo XML bien formado. Si el archivo ya existe, pasa por encima a su contenido
  - ***CreateElement*** *("Nombre")* – crea un nuevo elemento con el nombre de terminado. Devuelve un objeto **XmlElement**

XmlDocument (2)

- **Métodos Básicos (cont.):**
  - ***CreateNode****(XmlNodeType, "Nombre“, “namespaceURI “)* – crea un nuevo nodo con el tipo determinado, nombre de terminado. Devuelve un objeto **XmlNode**
  - ***RemoveChild*** *(XmlNode)* – elimina el nodo referenciado por parámetro del documento Xml
  - ***CreateNavigator()*** - crea un nuevo objeto XPathNavigator para navegar por los nodos del documento

XmlNode

- Permite crear y manipular un nodo de XML.
  - **Propiedades:**
    - **Attributes[ ]** – colección de elementos XmlAttribute de un nodo Xml
    - **ChildNodes[ ]** – colección de nodos secundarios de un nodo Xml
    - **NodeType** – determina el tipo del nodo actual. Se basa en el enumerado XmlNodeType:
    - **Name** – nombre completo del nodo
    - **InnerText** – valor del nodo
  - **Método Básico:**
    - ***AppendChild (ObjetoXmlElement)*** – agrega un nuevo nodo Xml al nodo actual

XmlNodeType

- Enumerado que determina los diferentes tipos de un XmlNode.
  - **Element**
  - **Attribute**
  - **Text** (contenido de texto de un nodo)

XmlAttribute

- Permite crear y manipular un atributo de XML.
  - **Propiedades:**
    - **Value –** valor del atributo (solo texto)

Linq to XML

Introducción

- Espacio de nombres
  - ***System.Linq***
  - ***System.XML.Linq***
- Nos permite trabajar con documentos XML
- Se debe tener una variable de tipo ***XElement*** cargada con los datos de un documento XML. Dicha variable se usara de origen de datos para la consulta

XElement

- Esta clase representa un elemento XML para LinQ. Contiene un objeto ***XName***, uno o varios atributos y, opcionalmente, puede incluir contenido
- ***Load***(«archivo») →operación de clase. Carga un *XElement* desde un archivo
- ***Elements***(«NombreEtiqueta») →Devuelve una colección filtrada de elementos secundarios de este elemento. En la colección sólo se incluyen los elementos que tienen un objeto *XName* coincidente.
- ***Element*** («NombreEtiqueta») →Obtiene el primer elemento con el nombre especificado.

XSLT

¿Qué es XSLT? (1)

- e**X**tensible **S**tylesheet **L**anguage **T**ransformations: Lenguaje de transformación basado en hojas de estilo.
- Utiliza una hoja de estilos para realizar la transformación:
  - **.CSS** → Hoja de estilo para HTML
  - **.XSL** → Hoja de estilo para XML
- XSL describe cómo el documento debe ser mostrado.
- Se utiliza para transformar documentos XML en otros tipos de documentos, incluso en documentos que no son XML, como ser HTML.

¿Qué es XSLT? (2)

- Realizan la transformación basada en reglas de plantillas (*templates*) que unidas al documento XML generan un tercer documento.
- Permite cambiar la estructura de un documento XML o sus valores.
- Un documento XSLT debe empezar siempre con el nodo raíz:

```
<xsl:stylesheet>
```

ó

```
<xsl:transform>
```

¿Qué es XSLT? (3)

- Un archivo XSLT, está basado en XML, por lo tanto debe estar bien formado para su correcto funcionamiento.
- XSL está compuesto por un conjunto de reglas llamadas *templates*.
- Para construir un template se debe utilizar el nodo **`<xsl:template>`**
- El atributo **match** del nodo **`<xsl:template>`** especifica a qué porción del XML se va a aplicar dicha regla.

¿Qué es XSLT? (4)

- El siguiente ejemplo indica que el *template* será aplicado a todo el documento XML que se adjunte:

```
<xsl:template match="/">
```

- Se puede utilizar el *template* para mezclar XSL y otro tipo de lenguaje, como ser HTML:

```
<xsl:template match="/”>
  <b><xsl:value-of select=“//title"/></b>
</xsl:template>
```

Sintaxis (1)

- **<xsl:value-of select="" />**: Se utiliza para extraer el valor de un nodo (elemento), determinado por el atributo **select**.
- **<xsl:for-each select="">**: Se utiliza para recorrer un conjunto de nodos, determinados por el atributo **select**.
- **<xsl:sort select="" />**: Se utiliza para ordenar el resultado de una iteración, en función del atributo **select**.
- **<xsl:if test="">**: Se utiliza para armar una condición lógica, que se determina en el atributo **test**.

Sintaxis (2)

- **<xsl:choose>**: Utilizado conjuntamente con el elemento **<xsl:when>** permite armar un conjunto de múltiples expresiones para evaluar el valor de un nodo:

```
<xsl:choose>
    <xsl:when test="expression">
        Hacer algo...
    </xsl:when>
    <xsl:otherwise>
        Hacer otra cosa...
    </xsl:otherwise>
</xsl:choose>
```