XMLNode y XMLDocument en el aula

Nieves Carralero Colmenar.
IES Pedro Mercedes. Junta de Comunidades de Castilla-La Mancha. España.
ncarralero@edu.jccm.es

Resumen

El presente artículo se centra en ofrecer una aletrnativa para el trabajo con XML y Microsoft .NET usando el objeto XMLNode para poder leer y escribir documentos.

Según la Orden EDU/2887/2010, de 2 de noviembre, por la que se establece el currículo del ciclo formativo de Grado Superior correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Web, se define un módulo con título: "Módulo Profesional: Lenguajes de marcas y sistemas de gestión de información". En él se pretende hacer una aproximación al trabajo con XML como pieza clave en el desarrollo de aplicaciones Web.

1. Introducción

Desde su creación, XML ha despertado encontradas pasiones, y como para cualquier tema en Internet, hay gente que desde el principio se ha dejado iluminar por sus expectativas, mientras que otras muchas lo han denostado o simplemente ignorado. Desde 1999 XML se ha convertido en una realidad empresarial palpable. Los programas que lo soportan han crecido del mismo modo exponencial, y a día de hoy no hay empresa de software que se precie que no anuncie la compatibilidad de sus productos más vendidos con este nuevo estándar: Microsoft (.Net), Oracle (Oracle 10i, Web Application Server) o Lotus (Notes) son tres claros ejemplos de ello. Aún más increíble es pensar que hay empresas que se han creado en torno a él, u otras que han movido su actividad hacia su ámbito (de SGML a XML, por ejemplo, como ArborText).

2. XMLNode

Más concretamente con .NET, según la documentación oficial de MSDN, *XmlNode*: está clase implementa el W3C *Core Document Object Model* (DOM) *Level 1* y el *Core* DOM *Level 2*. DOM es la representación en forma de árbol en memoria (caché) de un documento XML. *XmlNode* es la clase base de la implementación .NET del DOM. Admite selecciones *XPath* y proporciona capacidades de edición. [MSDN]

Como se verá *XmlNode* es la manera a través de la cual el *Framework* nos permite trabajar con documentos XML por medio de DOM, cuya mayor finalidad es la de proveer un sistema adecuado para la edición de dichos documentos.

XmlNode es una clase abstracta, y esto significa que en la mayoría de los casos no se trabaja directamente con ella sino que lo vamos a hacer utilizando aquellas clases que la heredan, de las cuales las más importantes son:

- XmlAttribute: Representa un atributo de un elemento XML.
- XmlDocument: Representa un documento XML.
- XmlDocumentFragment: Representa una porción de un documento XML.

Además de estas clases hay otra clase que hereda de *XmlNode* la cual también es abstracta y es *XmlLinkedNode*. Las clases que heredan de *XmlLinkedNode* nos permiten llegar a cada elemento del documento XML, y las más importantes son:

- XmlDeclaration: Representa al nodo de declaración XML: <?xml version='1.0' ...?>.
- XmlElement: Representa un elemento XML.

Existen muchas más clases dentro de la estructura jerárquica de clases que heredan de *XmlNode*, pero la mencionadas son suficientes para este trabajo.

3. Acceso a XML con XMLNode.

En la siguiente práctica se muestra un ejemplo de lectura de documento XML usando el objeto XMLNode y XMLDocumdent. La salida será los elementos y sus etiquetas mostrados en formato secuencial.

A continuación se muestra una imagen de un documento XML que será el que se usará de ejemplo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Books>
    <Book Pages="1088">
        <Author>Delaney, Kalen</Author>
        <Title>Inside Microsoft SQL Server 2000</Title>
        <Publisher>Microsoft Press</Publisher>
    </Book>-
    <Book Pages="997">
        <Author>Burton, Kevin</Author>
        <Title>.NET Common Language Runtime</Title>
        <Publisher>Sams</Publisher>
    </Book>
    <Book Pages="392">
        <Author>Cooper, James W.</Author>
        <Title>C# Design Patterns</Title>
        <Publisher>Addison Wesley</Publisher>
    </Book>
</Books>
```

Práctica 1:

Práctica 1: Realizar una lectura del documento XML anterior usando XmlTextNode y XMLDocument. En la práctica se deben seguir los siguientes puntos:

- Usan una solución recursiva.
- Comienza desde el nodo raíz (propiedad DocumentElement de XmlDocument)
- Visita cada hijo de cada nodo y visualiza la información deseada.
- Así hasta que no hay más nodos.

- Destacar en la solución el uso de *WhitespaceHandling* (que es una propiedad de XmlTextNode) y *XmlAttributeCollection* que implementa una colección de atributos.

La salida de la práctica 1 es todo el documento XML incluidas las declaraciones y los espacios en blanco y los atributos.

Solución:

Las funciones de la clase XmlNode son:

Member	*Type*	*Description*
`AppendChild()`	Method	Adds a new child node to the end of this node's list of children
`Attributes`	Property	Returns the attributes of the node as an `XmlAttributeCollection` object
`ChildNodes`	Property	Returns all child nodes of this node
`CloneNode()`	Method	Creates a duplicate of this node
`FirstChild`	Property	Returns the first child node of this node
`HasChildNodes`	Property	Returns `true` if this node has any children
`InnerText`	Property	Specifies the value of the node and all its children
`InnerXml`	Property	Specifies the markup representing only the children of this node
`InsertAfter()`	Method	Inserts a new node after this node

`InsertBefore()`	Method	Inserts a new node before this node
`LastChild`	Property	Returns the last child node of this node
`Name`	Property	Specifies the node's name
`NextSibling`	Property	Returns the next child of this node's parent node
`NodeType`	Property	Specifies this node's type
`OuterXml`	Property	Specifies the markup representing this node and its children
`OwnerDocument`	Property	Specifies the `XmlDocument` object that contains this node
`ParentNode`	Property	Returns this node's parent
`PrependChild()`	Method	Adds a new child node to the beginning of this node's list of children
`PreviousSibling`	Property	Returns the previous child of this node's parent node
`RemoveAll()`	Method	Removes all children of this node
`RemoveChild()`	Method	Removes a specified child of this node
`ReplaceChild()`	Method	Replaces a child of this node with a new node
`SelectNodes()`	Method	Selects a group of nodes matching an XPath expression
`SelectSingleNode()`	Method	Selects the first node matching an XPath expression
`WriteContentTo()`	Method	Writes all children of this node to an `XmlWriter` object
`WriteTo()`	Method	Writes this node to an `XmlWriter` object

XmlDocument

No existe un camino directo que cree un XmlNode para representar una entidad de un documento XML. Es necesario recuperar los XmlNode desde

objetos XmlDocument. Un objeto XmlDocument representa un documento XML completo.

Los métodos y propiedades más importantes de XmlDocument son mostrados en la siguiente tabla:

Member	*Type*	*Description*
`CreateAttribute()`	Method	Creates an attribute node
`CreateElement()`	Method	Creates an element node
`CreateNode()`	Method	Creates an `XmlNode` object
`DocumentElement`	Property	Returns the root `XmlNode` object for this document
`DocumentType`	Property	Returns the node containing the DTD declaration for this document, if it has one
`ImportNode()`	Method	Imports a node from another XML document
`Load()`	Method	Loads an XML document into the `XmlDocument` object
`LoadXml()`	Method	Loads the `XmlDocument` object from a string of XML data
`NodeChanged`	Event	Occurs after the value of a node has been changed
`NodeChanging`	Event	Occurs when the value of a node is about to be changed
`NodeInserted`	Event	Occurs when a new node has been inserted
`NodeInserting`	Event	Occurs when a new node is about to be inserted
`NodeRemoved`	Event	Occurs when a node has been removed
`NodeRemoving`	Event	Occurs when a node is about to be removed

PreserveWhitespace	Property	Returns true if whitespace in the document should be preserved when loading or saving the XML
Save()	Method	Saves the XmlDocument object as a file or stream
WriteTo()	Method	Saves the XmlDocument object to an XmlWriter object

El código comentado es el siguiente:

```
public void Run(String args)
{
      StringBuilder sbNode = new StringBuilder();
// Crea un nuevo XMLTextReader con el fichero XML
      XmlTextReader xtr = new XmlTextReader(args);
      // No detecta los espacios en blanco. Otra opción es ALL( por
defecto)
      xtr.WhitespaceHandling = WhitespaceHandling.None;
      XmlDocument xd = new XmlDocument();
      // Carga el fichero XML a una XmlDocument
      xd.Load(xtr); // Otras alternativas es cargar desde stream o
desde namefile directamente
      // Obtiene el nodo ráiz de un documento
      XmlNode xnodRoot = xd.DocumentElement;
      // Ahora, va recorriendo el DOM y visualiza
      XmlNode xnodWorking; //Un Node auxiliar
      if (xnodRoot.HasChildNodes)
      {
            xnodWorking = xnodRoot.FirstChild;
            while (xnodWorking != null)
            {
```

```
                    Visualiza_Hijos(xnodWorking, 0);
                    xnodWorking = xnodWorking.NextSibling; //Sacar el
siguiente Hermano
              }
        }
        // Clean up
        xtr.Close();
}
private void Visualiza_Hijos(XmlNode xnod, Int32 intDepth)
{
        // Va visualizando cada nodo junto con sus hijos
        // intDepth controla la profundidad
        StringBuilder sbNode = new StringBuilder();
        // Solo procesa Element y Text. Observar como compara con
XmlNodeType también como XmlTextReader
        if((xnod.NodeType == XmlNodeType.Element) ||
              (xnod.NodeType == XmlNodeType.Text) )
        {
              sbNode.Length = 0;
              for(int intI=1; intI <= intDepth ; intI++)
              {
                    sbNode.Append(" ");
              }
              sbNode.Append(xnod.Name + " ");
              sbNode.Append(xnod.NodeType.ToString());

              sbNode.Append(": " + xnod.Value);
              Console.WriteLine(sbNode.ToString());
              // Ahora añade los atributos si los hay.
              //utiliza una objeto XmlAttributeCollection
```

```
            XmlAttributeCollection atts = xnod.Attributes;
            if(atts != null)
            {
                  for(int intI = 0;intI < atts.Count; intI++)
                  {
                        sbNode.Length = 0;
                        for (int intJ = 1; intJ <= intDepth + 1;intJ++)
                        {
                              sbNode.Append(" ");
                        }
                        sbNode.Append(atts[intI].Name + " "); //Accedo
a los atributos como elementos de un colección
                        sbNode.Append(atts[intI].NodeType.ToString());
                        sbNode.Append(": " + atts[intI].Value);
                        Console.WriteLine(sbNode);
                  }
            }
            // Recursivamente explorar los nodos hijo
            XmlNode xnodworking;
            if (xnod.HasChildNodes)
            {
                  xnodworking = xnod.FirstChild;
                  while (xnodworking != null)
                  {
                        Visualiza_Hijos(xnodworking, intDepth + 1);
                        xnodworking = xnodworking.NextSibling;//proximo
hermano
                  }
            }
      }
```

```
}
```

4. **Aspectos a tener en cuenta en el aula.**

Esta práctica debe entenderse como una solución factible para trabajar con XML y DOM; de manera completa. Esta solución es sin duda, junto con la escritura, la más usada en este tipo de entorno, por su potencia y flexibilidad.

5. Bibliografía

MSDN. http://msdn.microsoft.com/es-es/library/bb972196.aspx

XML desde .NET: http://zarza.usal.es/~fgarcia/docencia/poo/04-05/Trabajos/XML.NET.pdf

.NET Framework: http://msdn.microsoft.com/en-us/magazine/cc302158.aspx

XML tutorial: http://www.programacion.net/html/xml/

SOCIEDAD DE LA INFORMACION

www.sociedadelainformacion.com

Edita:

Director: José Ángel Ruiz Felipe

Jefe de publicaciones: Antero Soria Luján

D.L.: AB 293-2001

ISSN: 1578-326x