

Manual para principiantes del XML DOM.

Edward Ocando

www.mitrompo.com/ingen

Julio, 2000.

Resumen: En este artículo se explica cómo obtener acceso y manipular documentos XML mediante la implementación XML DOM, según lo expuesto por el Intérprete (parser) XML de Microsoft®.

Contenido.

Introducción.

¿Qué es DOM exactamente?

¿Cómo funciona XML DOM?

¿Cómo se carga un documento?

Solución de errores.

Recuperación de la información de un documento XML.

¿Cómo se recorre un documento?

¿Y ahora qué?

Introducción.

Usted es un programador en Visual Basic® y ha recibido algunos datos en un documento XML (lenguaje de marcado extensible). Ahora desea extraer la información del documento XML e integrar esos datos en sus soluciones de Visual Basic. Por supuesto, podría escribir código para interpretar el contenido del archivo XML, pues no deja de ser un archivo de texto. Sin embargo, esta solución no sería muy productiva y desaprovecharía una de las ventajas de XML: el ser una forma estructurada de representar datos.

Una forma mejor de recuperar información de archivos XML es utilizar un intérprete (parser) de XML. Un intérprete de XML es, a grandes rasgos, un programa que lee un archivo XML y permite disponer de sus datos. En su calidad de programador en Visual Basic, querrá utilizar un intérprete que sea compatible con el modelo de objeto de documento (DOM) de XML. DOM define un conjunto estándar de comandos que los intérpretes exponen para facilitar el acceso al contenido de los documentos HTML y XML desde sus programas. Un intérprete de XML compatible con DOM toma los datos de un documento XML y los expone mediante un conjunto de objetos que se pueden programar. Con este artículo aprenderá la forma de obtener acceso y manipular los documentos XML a través de la implementación XML DOM, según lo expuesto por el intérprete XML de Microsoft® (Msxml.dll).

Antes de seguir leyendo, debería ver un archivo XML sin procesar para hacerse una idea de la ayuda que supone un intérprete. El siguiente código muestra el contenido del archivo Cds.xml, que contiene elementos de disco compacto. Cada elemento contiene datos como el nombre del artista, el título y las pistas.

```
<?xml version="1.0"?>
<!DOCTYPE compactdiscs SYSTEM "cds.dtd">
<compactdiscs>
  <compactdisc>
    <artist type="individual">Frank Sinatra</artist>
    <title numberoftracks="4">InThe Wee Small Hours</title>
  <tracks>
    <track>In The Wee Small Hours</track>
    <track>Mood Indigo</track>
    <track>Glad To Be Unhappy</track>
    <track>I Get Along Without You Very Well</track>
  </tracks>
  <price>12,99 dólares</price>
</compactdisc>
<compactdisc>
  <artist type="band">The Offspring</artist>
  <title numberoftracks="5">Americana</title>
<tracks>
  <track>Welcome</track>
  <track>Have You Ever</track>
  <track>Staring At The Sun</track>
  <track>Pretty Fly (For A White Guy)</track>
</tracks>
  <price>12,99 dólares </price>
```

```
</compactdisc>  
</compactdiscs>
```

La segunda línea del documento anterior hace referencia a un archivo DTD (definición de tipo de documento) externo. Un DTD define el diseño y el contenido previsto de un tipo de documento XML determinado. Un intérprete de XML puede utilizar un DTD para determinar si un documento es válido. Los archivos DTD son una de las formas de ayudar al intérprete para validar los documentos. Otro método de validación de documentos cada vez más popular es el uso de esquemas XML. En lugar de utilizar los DTD, con su propia e "interesante" sintaxis, los esquemas se definen con XML.

Este código muestra el contenido del archivo Cds.dtd utilizado por Cds.xml:

```
<!ELEMENT compactdiscs (compactdisc*)>  
<!ELEMENT compactdisc (artist, title, tracks, price)>  
<!ENTITY % Type "individual | band">  
<!ELEMENT artist (#PCDATA)>  
<!ATTLIST artist type (%Type;) #REQUIRED>  
<!ELEMENT title (#PCDATA)>  
<!ATTLIST title numberoftracks CDATA #REQUIRED>  
<!ELEMENT tracks (track*)>  
<!ELEMENT price (#PCDATA)>  
<!ELEMENT track (#PCDATA)>
```

En este artículo no se tratan en profundidad ni los DTD ni los esquemas XML. El lenguaje del esquema XML (en inglés) se basa en la nota XML-Data (en inglés) enviada al W3C.

¿Qué es un DOM exactamente?

El DOM para XML es un modelo de objeto que muestra el contenido de un documento XML. La Especificación de nivel 1 del Modelo de objeto de documento (DOM) del W3C define actualmente lo que debería mostrar un DOM como propiedades, métodos y eventos. La implementación de Microsoft del modelo DOM es totalmente compatible con el estándar del W3C y tiene características adicionales que facilitan el trabajo con archivos XML desde los programas.

¿Cómo funciona XML DOM?

Para utilizar XML DOM, hay que crear una instancia de un intérprete XML. Para ello, Microsoft muestra XML DOM mediante un conjunto de interfaces COM estándar en Msxml.dll. El archivo Msxml.dll contiene la biblioteca de tipos y el código de implementación para trabajar con documentos XML. Si trabaja con un cliente de scripts, como scripts VBScript ejecutadas en Internet Explorer, para usar el DOM hay que recurrir al método CreateObject para crear una instancia del objeto Parser (intérprete).

```
Set objParser = CreateObject( "Microsoft.XMLDOM" )
```

Si trabaja con VBScript desde una página Active Server (ASP), debe utilizar Server.CreateObject.

```
Set objParser = Server.CreateObject( "Microsoft.XMLDOM" )
```

Si trabaja con Visual Basic, puede obtener acceso al DOM estableciendo una referencia a la biblioteca de tipos MSXML, incluida en Msxml.dll. Para utilizar MSXML desde Visual Basic 6.0:

Abra el cuadro de diálogo Referencias del proyecto .

En la lista de objetos COM disponibles, seleccione Microsoft XML, versión 2.0. Si no encuentra este elemento, debe conseguir la biblioteca MSXML.

Entonces podrá crear una instancia del objeto Parser.

```
Dim xDoc As MSXML.DOMDocument Set xDoc = New MSXML.DOMDocument
```

¿Dónde puede conseguir el archivo Msxml.dll? Hay dos formas de obtener la biblioteca MSXML:

Puede instalar Internet Explorer 5.0, pues el intérprete MSXML es un componente integrado en el explorador.

Otra opción es obtener una versión redistribuible del intérprete XML de Microsoft.

Una vez establecida la referencia a la biblioteca de tipos en el proyecto de Visual Basic, invoque al intérprete, cargue un documento y trabaje con los datos del documento.

Quizás se estará preguntando con qué está trabajando en realidad. Si abre la biblioteca MSXML y examina su modelo de objeto con el Examinador de objetos de Visual Basic 6.0, verá que el modelo de objeto es bastante rico. En este artículo se muestra la forma de obtener acceso a un documento XML con la clase DOMDocument y la interfase IXMLDOMNode.

¿Cómo se carga un documento?

Para cargar un documento XML, primero debe crear una instancia de la clase DOMDocument:

```
Dim xDoc As MSXML.DOMDocument  
Set xDoc = New MSXML.DOMDocument
```

Una vez obtenida una referencia válida, abra un archivo con el método Load (cargar). El intérprete MSXML puede cargar documentos XML procedentes de un disco local, de la red (mediante referencias UNC) o de una dirección URL.

Si desea cargar un documento de un disco, cree la siguiente construcción con el método Load:

```
If xDoc.Load("C:\Mis documentos\cds.xml") Then  
' El documento se ha cargado correctamente.  
' Ahora haga algo interesante.  
Else  
' No se ha cargado el documento.  
End If
```

Cuando termine con el documento, debe liberar la referencia de objeto al documento. El intérprete MSXML no muestra ningún método Close (cerrar) explícito. Lo más recomendable es establecer explícitamente la referencia a Nothing.

```
Set xDoc = Nothing
```

De forma predeterminada, el intérprete carga los archivos de manera asíncrona. Para modificar esta configuración, debe manipular la propiedad booleana Async (asíncrono) del documento. Es muy importante examinar la propiedad ReadyState (Estado: preparado) de los documentos para confirmar que está preparado antes de empezar a examinar su contenido. La propiedad ReadyState devuelve uno de los cinco valores posibles enumerados a continuación:

Estado Valor

Uninitialized (sin inicializar): el método de carga no se ha iniciado. 0

Loading (cargando): mientras se ejecuta el método de carga. 1

Loaded (cargado): cuando ha terminado el método de carga. 2

Interactive (interactivo): hay suficiente DOM disponible para un examen de sólo lectura y los datos sólo se han interpretado parcialmente. 3

Completed (finalizado): ya se han cargado e interpretado los datos, que están disponibles para operaciones de lectura y escritura. 4

El intérprete MSXML muestra eventos que se utilizan al cargar documentos de gran tamaño para realizar un seguimiento del estado del proceso de carga. Dichos eventos también son de utilidad cuando se carga, por Internet y de forma asíncrona, un documento procedente de una dirección URL.

Para abrir un archivo desde una dirección URL, debe especificar la ubicación del archivo con una dirección URL completa. Debe incluir el prefijo http:// en la ubicación del archivo.

A continuación se muestra un ejemplo de carga de un archivo desde una dirección URL:

```
xDoc.async = False  
If xDoc.Load("http://www.develop.com/hp/brianr/cds.xml") Then  
  ' El documento se ha cargado correctamente.  
  ' Ahora haga algo interesante.  
Else  
  ' No se ha cargado el documento.  
End If
```

Si establece la propiedad Async del documento como False, el intérprete no devolverá el control a su código hasta que el documento se haya cargado completamente y esté listo para su manipulación. Si deja esa propiedad establecida como True, tendrá que examinar la propiedad ReadyState antes de obtener acceso al documento, o bien utilizar los eventos del DOMDocument (documento DOM) para que su código reciba una notificación cuando el documento esté preparado.

Solución de errores.

Un documento puede no cargarse por muchos motivos. Una causa corriente es que el nombre de documento pasado al método Load no sea válido. Otro motivo posible es que el propio documento XML no sea válido.

De forma predeterminada, el intérprete MSXML validará el documento comparándolo con un DTD o un esquema si así se ha especificado en el documento. Puede indicar al intérprete que no valide el documento. Para ello, establezca la propiedad ValidateOnParse (validar en análisis) del objeto DOMDocument antes de invocar el método Load.

```

Dim xDoc As MSXML.DOMDocument
Set xDoc = New MSXML.DOMDocument
xDoc.validateOnParse = False
If xDoc.Load("C:\Mis documentos\cds.xml") Then
' El documento se ha cargado correctamente.
' Ahora haga algo interesante.
Else
' No se ha cargado el documento.
End If

```

No es aconsejable desactivar la característica de validación del intérprete en las aplicaciones de producción. Un documento incorrecto puede hacer que el programa presente errores por muchos motivos. Como mínimo, podría proporcionar datos no válidos a los usuarios.

Independientemente del tipo de error, puede recurrir al objeto ParseError (error de interpretación) para solicitar información sobre el error al intérprete. Establezca una referencia a la interfase IXMLDOMParseError del propio documento para trabajar con las propiedades del objeto ParseError. La interfase IXMLDOMParseError muestra siete propiedades que pueden servir para investigar la causa del error.

El siguiente muestra un cuadro de mensajes y toda la información disponible sobre el error procedente del objeto ParseError.

```

Dim xDoc As MSXML.DOMDocument
Set xDoc = New MSXML.DOMDocument
If xDoc.Load("C:\Mis documentos\cds.xml") Then
' El documento se ha cargado correctamente.
' Ahora haga algo interesante.
Else
' No se ha cargado el documento.
Dim strErrText As String
Dim xPE As MSXML.IXMLDOMParseError
' Obtenga el objeto ParseError
Set xPE = xDoc.parseError
With xPE
strErrText = "Your XML Document failed to load" & _
"due the following error." & vbCrLf & _
"Error #: " & .errorCode & ": " & xPE.reason & _
"Line #: " & .Line & vbCrLf & _
"Line Position: " & .linepos & vbCrLf & _
"Position In File: " & .filepos & vbCrLf & _
"Source Text: " & .srcText & vbCrLf & _
"Document URL: " & .url
End With

MsgBox strErrText, vbExclamation
End If
Set xPE = Nothing

```

La información expuesta por el objeto ParseError puede mostrarse al usuario, guardarse en un archivo de errores o servirle para corregir el error.

Recuperación de la información de un documento XML.

Después de cargar el documento, el siguiente paso consiste en recuperar la información que contiene. Si bien es importante el objeto de documento, comprobará que termina por utilizar la interfase IXMLDOMNode (nodo de XML DOM) casi siempre. La interfase IXMLDOMNode se utiliza para leer y escribir en elementos de nodo individuales. Antes de hacer nada, debe saber que el intérprete MSXML admite actualmente 13 tipos de nodo. En la tabla siguiente se enumeran algunos de los tipos de nodo más comunes.

DOM Node Type Example

NODE_ELEMENT <artist type="band">Offspring</artist>

NODE_ATTRIBUTE <artist type="band">Offspring</artist>

NODE_TEXT <artist type="band">Offspring</artist>

NODE_PROCESSING_INSTRUCTION <?xml version="1.0"?>

NODE_DOCUMENT_TYPE <!DOCTYPE compactdiscs SYSTEM "cds.dtd">

El acceso al tipo de nodo se realiza a través de dos propiedades mostradas por la interfase IXMLDOMNode. La propiedad NodeType (tipo de nodo) muestra una enumeración de los elementos DOMNodeType (tipo de nodo DOM), algunos de los cuales figuran en la tabla anterior. Además, puede utilizar NodeTypeString (cadena de tipo de nodo) para recuperar una cadena de texto para el tipo de nodo.

Cuando tenga una referencia a un documento, puede empezar a recorrer la jerarquía de nodos. Desde la referencia al documento, puede obtener acceso a la propiedad ChildNodes (nodos secundarios), que ofrece un punto de entrada de arriba a abajo para todos los nodos del documento. La propiedad ChildNodes muestra la lista de nodos XML DOM (IXMLDOMNodeList), que admite la construcción For/Each de Visual Basic. Por lo tanto, se pueden enumerar todos los nodos individuales de la propiedad ChildNodes. La propiedad ChildNodes también muestra una propiedad Level (nivel), que devuelve el número de nodos secundarios existentes.

Tanto el objeto de documento como todos los nodos individuales pueden exponer una propiedad ChildNodes. Esto, junto con la propiedad HasChildNodes (tiene nodos secundarios) de IXMLDOMNode, facilita el recorrido de la jerarquía de nodos para examinar elementos, atributos y valores.

Hay que tener siempre presente la relación primario-secundario entre un elemento de documento y el valor del elemento. En el documento XML de CDs, por ejemplo, el elemento <title> muestra el título de una canción. Para recuperar el valor real del elemento <title>;, debe buscar nodos del tipo NODE_TEXT. Cuando haya encontrado un nodo con datos de interés, puede examinar los atributos e incluso obtener acceso a su nodo primario a través de la propiedad ParentNode (nodo primario).

¿Cómo se recorre un documento?

En un documento XML, se recorre el conjunto de nodos mostrados por el objeto de documento. Dado que los documentos XML son jerárquicos por naturaleza, es bastante fácil escribir una rutina recursiva para recorrer todo el documento.

La rutina LoadDocument (cargar documento) abre un documento XML. A continuación, LoadDocument llama a otra rutina, DisplayNode (mostrar nodo), que es la que recorre el

documento en realidad. LoadDocument pasa una referencia a la propiedad ChildNodes del documento XML abierto en la forma de un parámetro y un valor entero que especifica dónde comenzar el nivel con sangría. El código emplea el parámetro Indent (sangría) para dar formato a la presentación del texto en la Ventana Inmediato de la estructura del documento en Visual Basic.

La función DisplayNode recorre el documento buscando sólo nodos del tipo NODE_TEXT. Cuando el código encuentra un nodo del tipo NODE_TEXT, recupera el texto del nodo mediante la propiedad NodeValue (valor del nodo). Además, la propiedad ParentNode del nodo actual sirve para obtener una referencia retrospectiva a un nodo del tipo NODE_ELEMENT. Los nodos del tipo NODE_ELEMENT muestran una propiedad NodeName (nombre del nodo). Se muestra el contenido de NodeName y NodeValue.

Si un nodo tiene nodos secundarios, lo cual se determina revisando la propiedad HasChildNodes, la rutina DisplayNode se llama a sí misma recursivamente hasta llegar al final del documento.

La rutina DisplayNode escribe la información en la Ventana Inmediato de Visual Basic mediante Debug.Print (depurar e imprimir):

```
Public Sub LoadDocument()  
Dim xDoc As MSXML.DOMDocument  
Set xDoc = New MSXML.DOMDocument  
xDoc.validateOnParse = False  
If xDoc.Load("C:\Mis documentos\sample.xml") Then  
  ' El documento se ha cargado correctamente.  
  ' Ahora haga algo interesante.  
  DisplayNode xDoc.childNodes, 0  
Else  
  ' No se ha cargado el documento.  
  ' Para información sobre los errores consulte el listado  
  anterior.  
End If  
End Sub  
Public Sub DisplayNode(ByRef Nodes As MSXML.IXMLDOMNodeList, _  
  ByVal Indent As Integer)  
  
  Dim xNode As MSXML.IXMLDOMNode  
  Indent = Indent + 2  
  For Each xNode In Nodes  
    If xNode.nodeType = NODE_TEXT Then  
      Debug.Print Space$(Indent) & xNode.parentNode.nodeName & _  
        ":" & xNode.nodeValue  
    End If  
    If xNode.hasChildNodes Then  
      DisplayNode xNode.childNodes, Indent  
    End If  
  Next xNode  
End Sub
```

DisplayNode utiliza la propiedad HasChildNodes para determinar si debe llamarse a sí misma otra vez. También se puede utilizar la propiedad Level del nodo para buscar un valor mayor que 0.

¿Y ahora qué?

Este artículo sólo es una introducción. Ahora puede profundizar y ampliar sus conocimientos sobre XML y el intérprete MSXML. Puede hacer muchas cosas interesantes, como actualizar valores de elementos de nodo individuales, buscar dentro de un documento, generar sus propios documentos, etc.

Edward Ocando

www.mitrompo.com/ingen