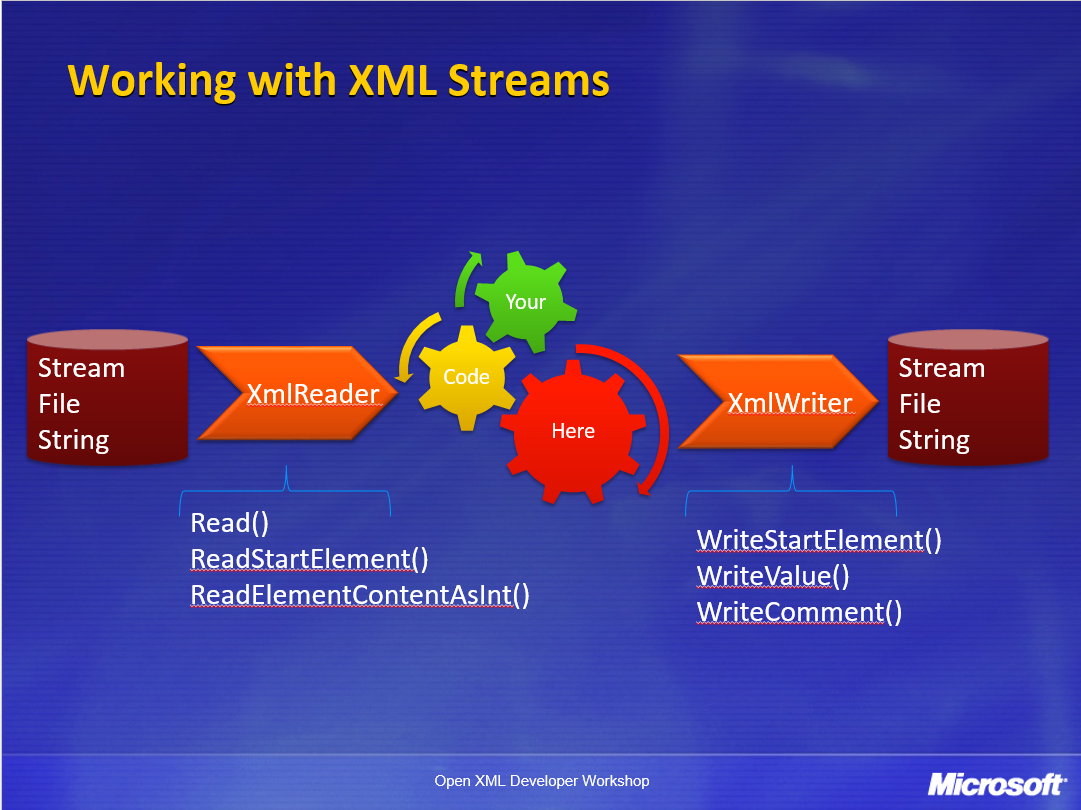
XML in VB.Net

Two (main) ways to use XML in .Net:

* XML Streams
* Working with the DOM

# Working with XML Streams



* Read XML with an XmlReader
  + Non cached, read-only, forward-only access
  + You are always at a specific location in the stream
* Write XML with an XmlWriter
  + Same architecture as the XmlReader based classes

Features of this approach

* Read and write from a Stream, String or File

Use this approach for speed and efficiency

* When working with large documents
* In data-centric processing – SOAP message parsing for instance

## Reading XML with an XmlReader

<order>

<orderItem>

<quantity>10</quantity>

<unitPrice>34.99</unitPrice>

</orderItem>

</order>

Dim myXMLReaderSettings As New System.Xml.XmlReaderSettings

myXMLReaderSettings.IgnoreComments = True

Using myXMLReader As System.Xml.XmlReader = System.Xml.XmlReader.Create("sample.xml", myXMLReaderSettings)

While myXMLReader.Read

If myXMLReader.IsStartElement And myXMLReader.LocalName = "orderItem" Then

'read quantity value (quantity child node in orderItem parent node)

myXMLReader.ReadToDescendant("quantity")

Dim quantity As Integer = myXMLReader.ReadElementContentAsInt

'read unitPrice value (unitPrice child node in orderItem parent node)

myXMLReader.ReadToNextSibling("unitPrice")

Dim unitPrice As Decimal = myXMLReader.ReadElementContentAsDecimal

Dim total As Decimal = quantity \* unitPrice

Console.WriteLine("Total: " & total.ToString)

End If

End While

End Using

XmlReader is an abstract class: you use the Create method to create an instance from an XmlReaderSettings.

Notice the forward only behavior:

* while loop
* Read() method
* ReadToDescendant() methods.

Best practice: always use a USING statement to assure that resources are released at the end of your processing loop.

## Writing XML with an XmlWriter

Using myXMLWriter As Xml.XmlWriter = Xml.XmlWriter.Create("sample.xml", myXMLWriterSetting)

myXMLWriter.WriteStartDocument(True)

myXMLWriter.WriteStartElement("order")

myXMLWriter.WriteStartElement("orderItem")

myXMLWriter.WriteElementString("quantity", Xml.XmlConvert.ToString(10))

myXMLWriter.WriteElementString("unitPrice", Xml.XmlConvert.ToString(34.99))

myXMLWriter.WriteEndElement() 'End "orderItem" node

myXMLWriter.WriteEndElement() 'End "order" node

End Using

Typically you’d have nested loops that write out many repeated nodes.

Best practice: use settings. Indent to make your generated XML more human-readable.

# Working with the DOM

The DOM is a heavier, more memory-intensive framework than XmlReader/XmlWriter approach.

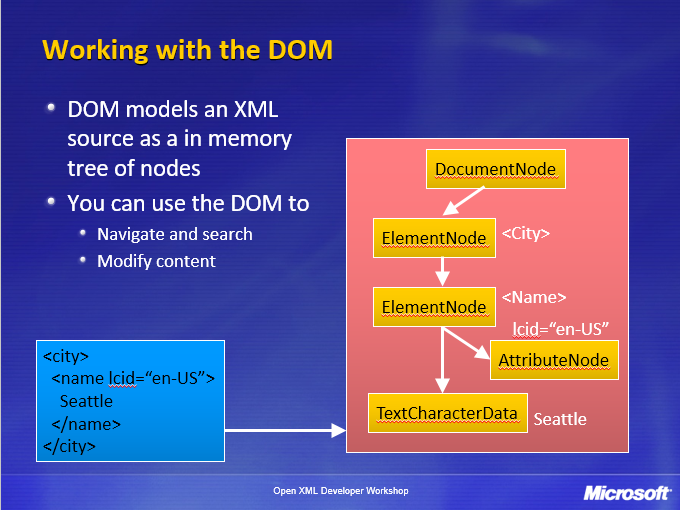
DOM is compliant with the DOM specification, W3C programming interface for XML

The DOM allows loading/saving from arbitrary locations, the manipulation and validation of elements, and the usage of Xpath to query for content.

DOM is more flexible, but its performance doesn’t scale for huge documents or large numbers of simultaneous documents (e.g., busy server scenarios).

DOM models an XML source as an in memory tree of nodes.

You can use the DOM to navigate, search or modify the content in the in memory tree of nodes.



## Reading XML with an XDocument -DOM

'Load an XML file into memory

Dim myXMLDoc As New Xml.XmlDocument

myXMLDoc.Load("sample.xml")

## Writing XML with an XDocument -DOM

'Save an XML file to disk

myXMLDoc.PreserveWhitespace = False 'auto-indents the output to make it readable

myXMLDoc.Save("books2.xml")

## Working with XML with an XDocument

The XmlNode class serves as the base class for various DOM elements

* XmlElement and XmlAttribute are samples

<?xml version="1.0" encoding="utf-8" ?>

<library>

<book isbn="123456789">

<title>XML.NET</title>

<price>19.99</price>

</book>

</library>

'Load an XML file into memory

Dim myXMLDoc As New Xml.XmlDocument

myXMLDoc.Load("books.xml")

'Get Document Node

Dim library As Xml.XmlNode = myXMLDoc.DocumentElement

'Get first child node of document - i.e. get book node

Dim book As Xml.XmlNode = library.FirstChild

'Get second child node in book node - i.e. get price node

Dim priceNode As Xml.XmlNode = book.ChildNodes(1) 'ChildNodes is 0-based

'Get "isbn" attribute of document node

Dim isbnNode As Xml.XmlNode = book.Attributes("isbn")

Dim price As String = priceNode.FirstChild.Value

Dim isbn As String = isbnNode.Value

The XmlDocument allows creation of all DOM elements

Dim comments As Xml.XmlComment = myXMLDoc.CreateComment("Some comment")

To remove nodes, use the parentXmlNode

book.RemoveChild(priceNode)

book.Attributes.Remove(isbnNode)

# Appendix – Code examples

## Sample XML Data file “MyFriends.XML”

<?xml version="1.0" encoding="utf-8" ?>

<my\_friend\_list>

<friend>

<family\_name>Simpson</family\_name>

<given\_name>Homer</given\_name>

<dob>

<day>19</day>

<month>4</month>

<year>1987</year>

</dob>

<good\_looking>False</good\_looking>

<address>742 Evergreen Terrace</address>

<surburb>Springfield</surburb>

</friend>

<friend>

<family\_name>Nahasapeemapetilon</family\_name>

<given\_name>Apu</given\_name>

<dob>

<day>25</day>

<month>2</month>

<year>1990</year>

</dob>

<good\_looking>False</good\_looking>

<address>Kwik-E-Mart</address>

<surburb>Springfield</surburb>

</friend>

<friend>

<family\_name>Duffman</family\_name>

<given\_name>Barry</given\_name>

<dob>

<day>21</day>

<month>9</month>

<year>1997</year>

</dob>

<good\_looking>True</good\_looking>

<address>Duffless</address>

<surburb>Keysborough</surburb>

</friend>

</my\_friend\_list>

## Stream Reader code

Structure MyFriend

Dim FamilyName As String

Dim GivenName As String

Dim FriendDob As Dob

Structure Dob

Dim DobDay As Integer

Dim DobMonth As Integer

Dim DobYear As Integer

End Structure

Dim GoodLooking As Boolean

Dim Address As String

Dim Surburb As String

End Structure

Dim AllMyFriends(2) As MyFriend

''' <summary>

''' This sub reads in an XML file via a stream reader. This will process each node from the file and then outputs the content via stringbuilder.

''' </summary>

''' <param name="fileName">The full path and filename of the XML file</param>

''' <param name="ignoreComments">Flag to determine whether to reading comments or not</param>

''' <returns>String containing the content of the XML with one line for each item</returns>

Function ReadFromXMLFileViaStream(fileName As String, ignoreComments As Boolean) As String

Dim myXMLReaderSettings As New System.Xml.XmlReaderSettings

If ignoreComments Then

myXMLReaderSettings.IgnoreComments = True

End If

Dim xmlString As New System.Text.StringBuilder

Dim indentLevel As Integer = 0

Using myXmlReader As System.Xml.XmlReader = XmlReader.Create(fileName, myXMLReaderSettings)

While myXmlReader.Read

Dim whiteSpace As Integer = indentLevel \* 5

Select Case myXmlReader.NodeType

Case XmlNodeType.Element

Console.Write("<{0}>", myXmlReader.Name)

indentLevel += 1

whiteSpace = indentLevel \* 5

xmlString.Append(" "c, whiteSpace)

xmlString.AppendFormat("<{0}", myXmlReader.Name)

If myXmlReader.HasAttributes Then

While myXmlReader.MoveToNextAttribute

xmlString.AppendFormat(" {0}='{1}'", myXmlReader.Name, myXmlReader.Value)

End While

End If

xmlString.Append(">")

xmlString.AppendLine()

Case XmlNodeType.Text

indentLevel += 1

whiteSpace = indentLevel \* 5

xmlString.Append(" "c, whiteSpace)

xmlString.Append(myXmlReader.Value)

xmlString.AppendLine()

indentLevel -= 1

whiteSpace = indentLevel \* 5

Case XmlNodeType.CDATA

xmlString.Append("<![CDATA[{0}]]>", myXmlReader.Value)

xmlString.AppendLine()

Case XmlNodeType.ProcessingInstruction

xmlString.AppendFormat("<?{0} {1}?>", myXmlReader.Name, myXmlReader.Value)

Case XmlNodeType.Comment

xmlString.Append("<!--{0}-->", myXmlReader.Value)

xmlString.AppendLine()

Case XmlNodeType.XmlDeclaration

xmlString.AppendFormat("<?xml {0}?>", myXmlReader.Value.ToString)

xmlString.AppendLine()

Case XmlNodeType.Document

Case XmlNodeType.DocumentType

xmlString.AppendFormat("<!DOCTYPE {0} [{1}]", myXmlReader.Name, myXmlReader.Value)

Case XmlNodeType.EntityReference

xmlString.Append(myXmlReader.Name)

xmlString.AppendLine()

Case XmlNodeType.EndElement

xmlString.Append(" "c, whiteSpace)

xmlString.AppendFormat("</{0}>", myXmlReader.Name)

xmlString.AppendLine()

indentLevel -= 1

whiteSpace = indentLevel \* 5

End Select

End While

End Using

Return xmlString.ToString

End Function

''' <summary>

''' This sub reads in an XML file with the "myFriends" structure via a stream reader.

''' This will store each "friend" node into the "AllMyFriends". "friend" contains all child nodes which corresponds to the record type "MyFriend".

''' </summary>

''' <param name="inputFilename">The full path and filename of the XML file</param>

Sub ReadFromXMLFileViaStream(inputFilename)

Dim myXMLReaderSettings As New System.Xml.XmlReaderSettings

myXMLReaderSettings.IgnoreComments = True

Dim friendNumber As Integer = -1

Dim xmlString As New System.Text.StringBuilder

Using myXmlReader As System.Xml.XmlReader = XmlReader.Create(inputFilename, myXMLReaderSettings)

myXmlReader.ReadToDescendant("friend")

Do

friendNumber += 1

myXmlReader.ReadToDescendant("family\_name")

myXmlReader.Read()

AllMyFriends(friendNumber).FamilyName = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadToFollowing("given\_name")

myXmlReader.Read()

AllMyFriends(friendNumber).GivenName = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadToFollowing("dob")

myXmlReader.ReadToDescendant("day")

myXmlReader.Read()

AllMyFriends(friendNumber).FriendDob.DobDay = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadStartElement("month")

AllMyFriends(friendNumber).FriendDob.DobMonth = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadStartElement("year")

AllMyFriends(friendNumber).FriendDob.DobYear = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadEndElement()

myXmlReader.ReadStartElement("good\_looking")

AllMyFriends(friendNumber).GoodLooking = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadStartElement("address")

AllMyFriends(friendNumber).Address = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadStartElement("surburb")

AllMyFriends(friendNumber).Surburb = myXmlReader.Value

myXmlReader.Read()

myXmlReader.ReadEndElement()

myXmlReader.ReadEndElement()

Loop While myXmlReader.ReadToNextSibling("friend")

End Using

End Sub

''' <summary>

''' This sub writes contents of an Array of Records to an XML file

''' </summary>

''' <param name="outputFileName"></param>

Sub WriteToXMLFileViaStream(outputFileName As String)

Dim myXMLWriterSettings As New System.Xml.XmlWriterSettings

myXMLWriterSettings.CloseOutput = True

myXMLWriterSettings.Indent = True

Using myXMLWriter As System.Xml.XmlWriter = XmlWriter.Create(outputFileName, myXMLWriterSettings)

myXMLWriter.WriteStartDocument(True)

myXMLWriter.WriteStartElement("my\_friend\_list")

For Each person As MyFriend In AllMyFriends

myXMLWriter.WriteStartElement("friend")

myXMLWriter.WriteElementString("family\_name", person.FamilyName.ToString)

myXMLWriter.WriteElementString("given\_name", person.GivenName.ToString)

'dob

myXMLWriter.WriteStartElement("dob")

myXMLWriter.WriteElementString("day", person.FriendDob.DobDay.ToString)

myXMLWriter.WriteElementString("month", person.FriendDob.DobMonth.ToString)

myXMLWriter.WriteElementString("year", person.FriendDob.DobYear.ToString)

myXMLWriter.WriteEndElement()

'end dob

myXMLWriter.WriteElementString("good\_looking", person.GoodLooking.ToString)

myXMLWriter.WriteElementString("address", person.Address.ToString)

myXMLWriter.WriteElementString("surburb", person.Surburb.ToString)

myXMLWriter.WriteEndElement()

Next

myXMLWriter.WriteEndElement()

End Using

End Sub

Private Function GetFileName() As String

Dim returnFileName As String = String.Empty

'Create a "OpenFileDialog" object to use

Dim openFileDialog1 As New OpenFileDialog()

'Sets the file extensions that can be selected to filter the list

openFileDialog1.Filter = "txt files (\*.xml)|\*.xml|All files (\*.\*)|\*.\*"

'Sets the initial file extensions filter that is applied

openFileDialog1.FilterIndex = 1

openFileDialog1.RestoreDirectory = True

'Open/Display the control AND if the "OK" button is pressed to close the control

'then do the next section of code

If openFileDialog1.ShowDialog() = System.Windows.Forms.DialogResult.OK Then

Dim fileNameAndFullPath As String

fileNameAndFullPath = openFileDialog1.FileName

returnFileName = fileNameAndFullPath

Else

returnFileName = String.Empty

End If

Return returnFileName

End Function

## XDocument - DOM code

Dim doc As XmlDocument

Sub ReadXMLIntoDOM(FilenameAndPath As String)

Try

doc = New XmlDocument()

doc.Load(FilenameAndPath)

Catch xmlex As XmlException ' Handle the Xml Exceptions here

Console.WriteLine("{0}", xmlex.Message)

Catch ex As Exception ' Handle the generic Exceptions here

Console.WriteLine("{0}", ex.Message)

End Try

End Sub

Sub SelectAllFriendNodes()

' 1. Select all the Friends family\_name by using an XPath query.

Dim nodeList As XmlNodeList = doc.SelectNodes("//friend/family\_name")

Dim node As XmlNode

Console.WriteLine("{0}", "TITLES LIST: ")

For Each node In nodeList

'Console.WriteLine("{0}", node.InnerText)

txtOutput.Text &= node.InnerText.ToString & vbCrLf

Next

End Sub

Sub SelectComments()

' 2. Read the XmlDeclartion values.

Dim decl As XmlDeclaration = CType(doc.FirstChild, XmlDeclaration)

Console.WriteLine("{0}", vbNewLine & "XML DECLARTION:")

Console.WriteLine("{0}", "Version " & "= " & decl.Version)

Console.WriteLine("{0}", "Encoding " & "= " & decl.Encoding)

Console.WriteLine("{0}", "Standalone " & "= " & decl.Standalone)

End Sub

Sub SelectFirstNodesAttributes()

' 3. Move to the first node of DOM and get all of its attributes.

Dim root As XmlElement = doc.DocumentElement

Dim node As XmlNode

node = root.FirstChild

Dim attr As XmlAttribute

Console.WriteLine("{0}", vbNewLine & "ATTRIBUTES OF THE FIRST CHILD:")

For Each attr In node.Attributes

Console.WriteLine("{0}", attr.Name & " = " & attr.InnerText)

Next

End Sub

Sub SelectChildNodes()

' 4. Navigate to the child nodes of the first Friend node.

Dim root As XmlElement = doc.DocumentElement

Dim node As XmlNode

node = root.FirstChild

Dim cNode As XmlNode

Console.WriteLine("{0}", vbNewLine & "FIRST NODE'S CHILDREN:")

If node.HasChildNodes Then

For Each cNode In node.ChildNodes

Console.WriteLine("{0}", cNode.OuterXml)

Next

End If

End Sub

Sub MoveToNextSiblingNode()

Dim root As XmlElement = doc.DocumentElement

Dim node As XmlNode

node = root.FirstChild

' 5. Navigate to the next sibling of the first Book node.

node = node.NextSibling

Console.WriteLine("{0}", vbNewLine & "NEXT SIBLING:")

If Not node Is Nothing Then

Console.WriteLine("{0}", node.OuterXml)

End If

End Sub