

VB.NET - XML PROCESSING

The Extensible Markup Language *XML* is a markup language much like HTML or SGML. This is recommended by the World Wide Web Consortium and available as an open standard.

The **System.Xml** namespace in the .Net Framework contains classes for processing XML documents. Following are some of the commonly used classes in the System.Xml namespace.

S.N	Class	Description
1	XmlAttribute	Represents an attribute. Valid and default values for the attribute are defined in a document type definition <i>DTD</i> or schema.
2	XmlCDATASection	Represents a CDATA section.
3	XmlCharacterData	Provides text manipulation methods that are used by several classes.
4	XmlComment	Represents the content of an XML comment.
5	XmlConvert	Encodes and decodes XML names and provides methods for converting between common language runtime types and XML Schema definition language <i>XSD</i> types. When converting data types, the values returned are locale independent.
6	XmlDeclaration	Represents the XML declaration node <code><?xml version='1.0'...?></code> .
7	XmlDictionary	Implements a dictionary used to optimize Windows Communication Foundation <i>WCF</i> 's XML reader/writer implementations.
8	XmlDictionaryReader	An abstract class that the Windows Communication Foundation <i>WCF</i> derives from <i>XmlReader</i> to do serialization and deserialization.
9	XmlDictionaryWriter	Represents an abstract class that Windows Communication Foundation <i>WCF</i> derives from <i>XmlWriter</i> to do serialization and deserialization.
10	XmlDocument	Represents an XML document.
11	XmlDocumentFragment	Represents a lightweight object that is useful for tree insert operations.
12	XmlDocumentType	Represents the document type declaration.
13	XmlElement	Represents an element.
14	XmlEntity	Represents an entity declaration, such as <code><!ENTITY... ></code> .
15	XmlEntityReference	Represents an entity reference node.
16	XmlAttribute	Returns detailed information about the last exception.
17	XmlImplementation	Defines the context for a set of <i>XmlDocument</i> objects.
18	XmlLinkedNode	Gets the node immediately preceding or following this node.
19	XmlNode	Represents a single node in the XML document.

20	XmlNodeList	Represents an ordered collection of nodes.
21	XmlNodeReader	Represents a reader that provides fast, non-cached forward only access to XML data in an XmlNode.
22	XmlNotation	Represents a notation declaration, such as <!NOTATION...>.
23	XmlParserContext	Provides all the context information required by the XmlReader to parse an XML fragment.
24	XmlProcessingInstruction	Represents a processing instruction, which XML defines to keep processor-specific information in the text of the document.
25	XmlQualifiedName	Represents an XML qualified name.
26	XmlReader	Represents a reader that provides fast, noncached, forward-only access to XML data.
27	XmlReaderSettings	Specifies a set of features to support on the XmlReader object created by the Create method.
28	XmlResolver	Resolves external XML resources named by a Uniform Resource Identifier <i>URI</i> .
29	XmlSecureResolver	Helps to secure another implementation of XmlResolver by wrapping the XmlResolver object and restricting the resources that the underlying XmlResolver has access to.
30	XmlSignificantWhitespace	Represents white space between markup in a mixed content node or white space within an xml:space='preserve' scope. This is also referred to as significant white space.
31	XmlText	Represents the text content of an element or attribute.
32	XmlTextReader	Represents a reader that provides fast, non-cached, forward-only access to XML data.
33	XmlTextWriter	Represents a writer that provides a fast, non-cached, forward-only way of generating streams or files containing XML data that conforms to the W3C Extensible Markup Language <i>XML</i> 1.0 and the Namespaces in XML recommendations.
34	XmlUrlResolver	Resolves external XML resources named by a Uniform Resource Identifier <i>URI</i> .
35	XmlWhitespace	Represents white space in element content.
36	XmlWriter	Represents a writer that provides a fast, non-cached, forward-only means of generating streams or files containing XML data.
37	XmlWriterSettings	Specifies a set of features to support on the XmlWriter object created by the XmlWriter.Create method.

XML Parser APIs

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

- **Simple API for XML SAX** : Here, you register callbacks for events of interest and then let the parser proceed through the document. This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk, and the entire file is never

stored in memory.

- **Document Object Model DOM API** : This is World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical *tree - based* form to represent all the features of an XML document.

SAX obviously can't process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other there is no reason why you can't use them both for large projects.

For all our XML code examples, let's use a simple XML file `movies.xml` as an input:

```
<?xml version="1.0"?>
<collection shelf="New Arrivals">
<movie title="Enemy Behind">
  <type>War, Thriller</type>
  <format>DVD</format>
  <year>2003</year>
  <rating>PG</rating>
  <stars>10</stars>
  <description>Talk about a US-Japan war</description>
</movie>
<movie title="Transformers">
  <type>Anime, Science Fiction</type>
  <format>DVD</format>
  <year>1989</year>
  <rating>R</rating>
  <stars>8</stars>
  <description>A schientific fiction</description>
</movie>
  <movie title="Trigun">
    <type>Anime, Action</type>
    <format>DVD</format>
    <episodes>4</episodes>
    <rating>PG</rating>
    <stars>10</stars>
    <description>Vash the Stampede!</description>
  </movie>
  <movie title="Ishtar">
    <type>Comedy</type>
    <format>VHS</format>
    <rating>PG</rating>
    <stars>2</stars>
    <description>Viewable boredom</description>
  </movie>
</collection>
```

Parsing XML with SAX API

In SAX model, you use the **XmlReader** and **XmlWriter** classes to work with the XML data.

The **XmlReader** class is used to read XML data in a fast, forward-only and non-cached manner. It reads an XML document or a stream.

Example 1

This example demonstrates reading XML data from the file `movies.xml`.

Take the following steps:

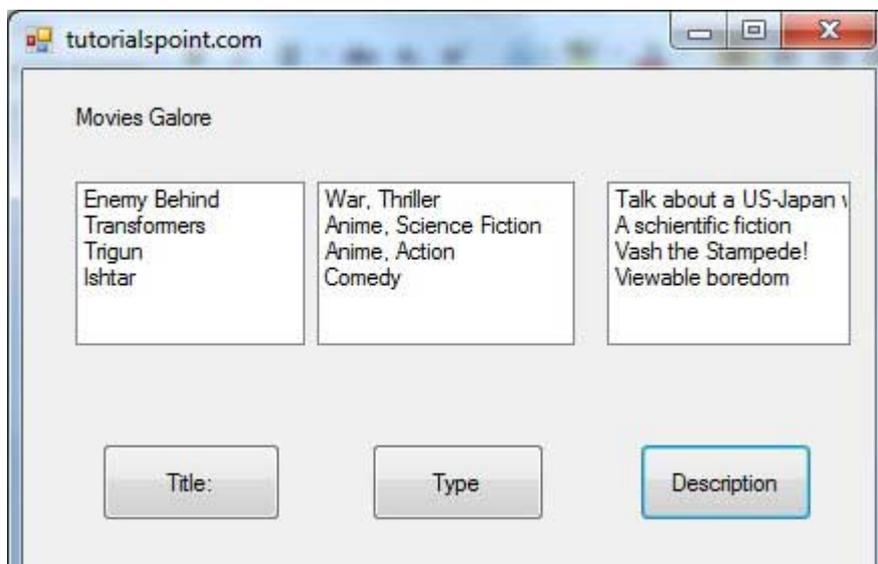
- Add the `movies.xml` file in the `bin\Debug` folder of your application.
- Import the `System.Xml` namespace in `Form1.vb` file.

- Add a label in the form and change its text to 'Movies Galore'.
- Add three list boxes and three buttons to show the title, type and description of a movie from the xml file.
- Add the following code using the code editor window.

```
Imports System.Xml
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "tutorialspoint.com"
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        ListBox1().Items.Clear()
        Dim xr As XmlReader = XmlReader.Create("movies.xml")
        Do While xr.Read()
            If xr.NodeType = XmlNodeType.Element AndAlso xr.Name = "movie" Then
                ListBox1.Items.Add(xr.GetAttribute(0))
            End If
        Loop
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        ListBox2().Items.Clear()
        Dim xr As XmlReader = XmlReader.Create("movies.xml")
        Do While xr.Read()
            If xr.NodeType = XmlNodeType.Element AndAlso xr.Name = "type" Then
                ListBox2.Items.Add(xr.ReadElementString)
            Else
                xr.Read()
            End If
        Loop
    End Sub
    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        ListBox3().Items.Clear()
        Dim xr As XmlReader = XmlReader.Create("movies.xml")
        Do While xr.Read()
            If xr.NodeType = XmlNodeType.Element AndAlso xr.Name = "description" Then
                ListBox3.Items.Add(xr.ReadElementString)
            Else
                xr.Read()
            End If
        Loop
    End Sub
End Class
```

Execute and run the above code using **Start** button available at the Microsoft Visual Studio tool bar. Clicking on the buttons would display, title, type and description of the movies from the file.



The **XmlWriter** class is used to write XML data into a stream, a file or a TextWriter object. It also works in a forward-only, non-cached manner.

Example 2

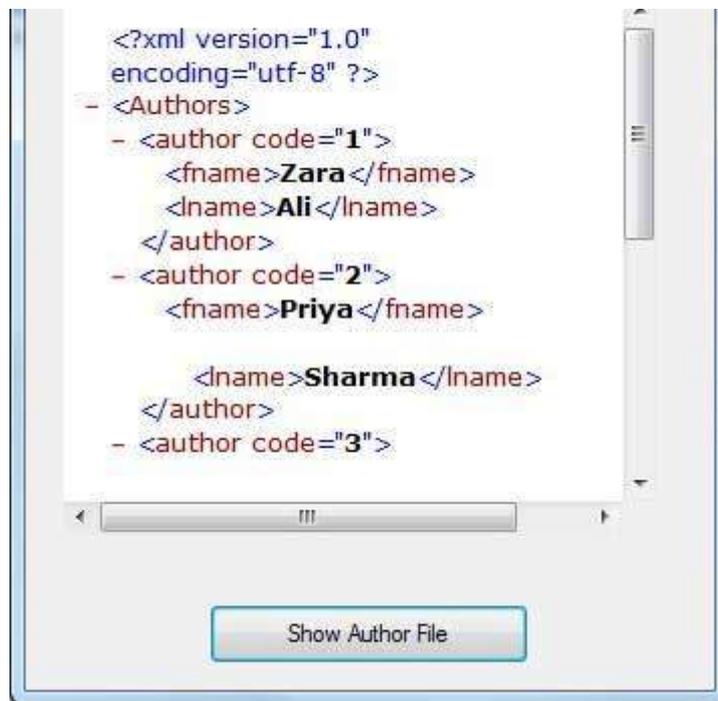
Let us create an XML file by adding some data at runtime. Take the following steps:

- Add a WebBrowser control and a button control in the form.
- Change the Text property of the button to Show Authors File.
- Add the following code in the code editor.

```
Imports System.Xml
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "tutorialspoint.com"
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim xws As XmlWriterSettings = New XmlWriterSettings()
        xws.Indent = True
        xws.NewLineOnAttributes = True
        Dim xw As XmlWriter = XmlWriter.Create("authors.xml", xws)
        xw.WriteStartDocument()
        xw.WriteStartElement("Authors")
        xw.WriteStartElement("author")
        xw.WriteAttributeString("code", "1")
        xw.WriteElementString("fname", "Zara")
        xw.WriteElementString("lname", "Ali")
        xw.WriteEndElement()
        xw.WriteStartElement("author")
        xw.WriteAttributeString("code", "2")
        xw.WriteElementString("fname", "Priya")
        xw.WriteElementString("lname", "Sharma")
        xw.WriteEndElement()
        xw.WriteStartElement("author")
        xw.WriteAttributeString("code", "3")
        xw.WriteElementString("fname", "Anshuman")
        xw.WriteElementString("lname", "Mohan")
        xw.WriteEndElement()
        xw.WriteStartElement("author")
        xw.WriteAttributeString("code", "4")
        xw.WriteElementString("fname", "Bibhuti")
        xw.WriteElementString("lname", "Banerjee")
        xw.WriteEndElement()
        xw.WriteStartElement("author")
        xw.WriteAttributeString("code", "5")
        xw.WriteElementString("fname", "Riyan")
        xw.WriteElementString("lname", "Sengupta")
        xw.WriteEndElement()
        xw.WriteEndElement()
        xw.WriteEndDocument()
        xw.Flush()
        xw.Close()
        WebBrowser1.Url = New Uri(AppDomain.CurrentDomain.BaseDirectory + "authors.xml")
    End Sub
End Class
```

Execute and run the above code using **Start** button available at the Microsoft Visual Studio tool bar. Clicking on the Show Author File would display the newly created authors.xml file on the web browser.





Parsing XML with DOM API

According to the Document Object Model *DOM*, an XML document consists of nodes and attributes of the nodes. The **XmlDocument** class is used to implement the XML DOM parser of the .Net Framework. It also allows you to modify an existing XML document by inserting, deleting or updating data in the document.

Following are some of the commonly used methods of the **XmlDocument** class:

S.N Method Name & Description

- 1 **AppendChild**
Adds the specified node to the end of the list of child nodes, of this node.
- 2 **CreateAttributeString**
Creates an XmlAttribute with the specified Name.
- 3 **CreateComment**
Creates an XmlComment containing the specified data.
- 4 **CreateDefaultAttribute**
Creates a default attribute with the specified prefix, local name and namespace URI.
- 5 **CreateElementString**
Creates an element with the specified name.
- 6 **CreateNodeString, String, String**

Creates an XmlNode with the specified node type, Name, and NamespaceURI.

7

CreateXmlNode(XmlNodeType, String, String)

Creates an XmlNode with the specified XmlNodeType, Name, and NamespaceURI.

8

CreateXmlNode(XmlNodeType, String, String, String)

Creates a XmlNode with the specified XmlNodeType, Prefix, Name, and NamespaceURI.

9

CreateProcessingInstruction

Creates an XmlProcessingInstruction with the specified name and data.

10

CreateSignificantWhitespace

Creates an XmlSignificantWhitespace node.

11

CreateTextNode

Creates an XmlText with the specified text.

12

CreateWhitespace

Creates an XmlWhitespace node.

13

CreateXmlDeclaration

Creates an XmlDeclaration node with the specified values.

14

GetElementById

Gets the XmlElement with the specified ID.

15

GetElementsByTagName(String)

Returns an XmlNodeList containing a list of all descendant elements that match the specified Name.

16

GetElementsByTagName(String, String)

Returns an XmlNodeList containing a list of all descendant elements that match the specified LocalName and NamespaceURI.

17

InsertAfter

Inserts the specified node immediately after the specified reference node.

18	InsertBefore Inserts the specified node immediately before the specified reference node.
19	LoadStream Loads the XML document from the specified stream.
20	LoadString Loads the XML document from the specified URL.
21	LoadTextReader Loads the XML document from the specified TextReader.
22	LoadXmlReader Loads the XML document from the specified XmlReader.
23	LoadXml Loads the XML document from the specified string.
24	PrependChild Adds the specified node to the beginning of the list of child nodes for this node.
25	ReadNode Creates an XmlNode object based on the information in the XmlReader. The reader must be positioned on a node or attribute.
26	RemoveAll Removes all the child nodes and/or attributes of the current node.
27	RemoveChild Removes specified child node.
28	ReplaceChild Replaces the child node oldChild with newChild node.
29	SaveStream Saves the XML document to the specified stream.

30

SaveString

Saves the XML document to the specified file.

31

SaveTextWriter

Saves the XML document to the specified TextWriter.

32

SaveXmlWriter

Saves the XML document to the specified XmlWriter.

Example 3

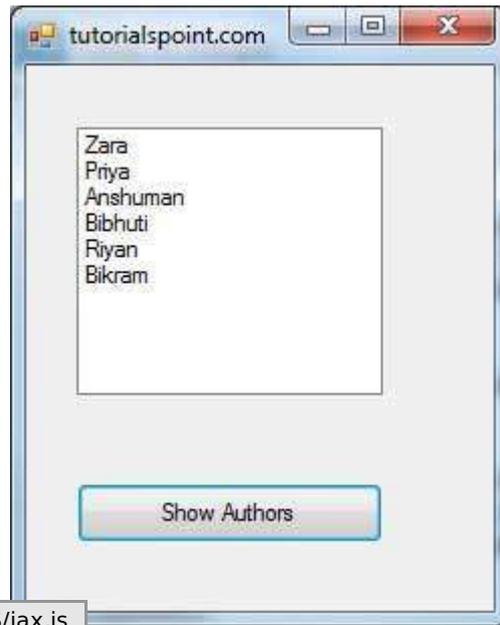
In this example, let us insert some new nodes in the xml document authors.xml and then show all the authors' first names in a list box.

Take the following steps:

- Add the authors.xml file in the bin/Debug folder of your application
it should be there if you have tried the last example
- Import the System.Xml namespace
- Add a list box and a button control in the form and set the text property of the button control to Show Authors.
- Add the following code using the code editor.

```
Imports System.Xml
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Set the caption bar text of the form.
        Me.Text = "tutorialspoint.com"
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        ListBox1.Items.Clear()
        Dim xd As XmlDocument = New XmlDocument()
        xd.Load("authors.xml")
        Dim newAuthor As XmlElement = xd.CreateElement("author")
        newAuthor.SetAttribute("code", "6")
        Dim fn As XmlElement = xd.CreateElement("fname")
        fn.InnerText = "Bikram"
        newAuthor.AppendChild(fn)
        Dim ln As XmlElement = xd.CreateElement("lname")
        ln.InnerText = "Seth"
        newAuthor.AppendChild(ln)
        xd.DocumentElement.AppendChild(newAuthor)
        Dim tr As XmlTextWriter = New XmlTextWriter("movies.xml", Nothing)
        tr.Formatting = Formatting.Indented
        xd.WriteContentTo(tr)
        tr.Close()
        Dim n1 As XmlNodeList = xd.GetElementsByTagName("fname")
        For Each node As XmlNode In n1
            ListBox1.Items.Add(node.InnerText)
        Next node
    End Sub
End Class
```

Execute and run the above code using **Start** button available at the Microsoft Visual Studio tool bar. Clicking on the Show Author button would display the first names of all the authors including the one we have added at runtime.



Loading [MathJax]/jax/output/HTML-CSS/jax.js