**How to XML in C#**

**XML** is a general purpose tag based language and very easy to transfer and store data across applications. Like HTML , XML is a subset of **SGML** (Standard Generalized Markup Language). XML is a platform independent language, so the information formatted in XML can be used in any other platforms (Operating Systems). **XML** is a self describing language and it gives the data as well as the rules to identify what information it contains.

**XML** files are made up of tags that contains data. Generally a start tag and end tag to hold the data. For example, if you want to create an XML tag name "Header" , the start tag is like **< Header >** and the end tag is like **< /Header >** . We can fill our information between these tags.



While creating an XML file , some important points have to remember :

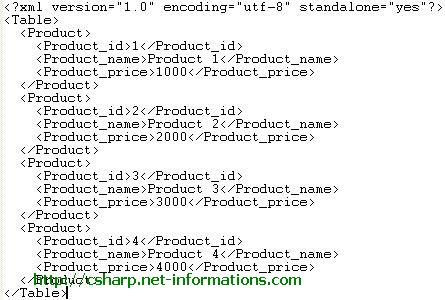
**\* XML is case sensitive**

ex: **< Header > is not same as < HeadeR >** .

**\* Tags must be closed in the reverse order that they were opened**

ex : **< first-tag >< second-tag > Data here < /second-tag > < /first-tag >** .

**Sample XML File**



The .Net technology is widely supported XML file format. The [.Net Framework](http://vb.net-informations.com/framework/what_is_net_framework.htm) provides the Classes for read, write, and other operations in XML formatted files . These classes are stored in the namespaces like System.Xml, System.Xml.Schema, System.Xml.Serialization, System.Xml.XPath, System.Xml.Xsl etc. The Dataset in **ADO.NET** uses XML as its internal storage format.

You can use any text editor to create an **XML** file . More over XML files are readable by humans as well as computers. From the following links you can see how to use XML in C# Programming Language.

**How to create an XML file in C#**

**XML** is a platform independent language, so the information formatted in XML can be used in any other platforms (Operating Systems). Once we create an XML file in one platform it can be used in other platforms also.

In order to creating a new XML file in C#, we are using **XmlTextWriter Class** . The class takes FileName and Encoding as argument. Also we are here passing formatting details . The following C# source code creating an XML file product.xml and add four rows in the file.

using System;

using System.Data;

using System.Windows.Forms;

using System.Xml;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

XmlTextWriter writer = new XmlTextWriter("product.xml", System.Text.Encoding.UTF8);

writer.WriteStartDocument(true);

writer.Formatting = Formatting.Indented;

writer.Indentation = 2;

writer.WriteStartElement("Table");

createNode("1", "Product 1", "1000", writer);

createNode("2", "Product 2", "2000", writer);

createNode("3", "Product 3", "3000", writer);

createNode("4", "Product 4", "4000", writer);

writer.WriteEndElement();

writer.WriteEndDocument();

writer.Close();

MessageBox.Show("XML File created ! ");

}

private void createNode(string pID, string pName, string pPrice, XmlTextWriter writer)

{

writer.WriteStartElement("Product");

writer.WriteStartElement("Product\_id");

writer.WriteString(pID);

writer.WriteEndElement();

writer.WriteStartElement("Product\_name");

writer.WriteString(pName);

writer.WriteEndElement();

writer.WriteStartElement("Product\_price");

writer.WriteString(pPrice);

writer.WriteEndElement();

writer.WriteEndElement();

}

}

}

**How to read XML file from C#**

using System;

using System.Data;

using System.Windows.Forms;

using System.Xml;

using System.IO;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

XmlDataDocument xmldoc = new XmlDataDocument();

XmlNodeList xmlnode ;

int i = 0;

string str = null;

FileStream fs = new FileStream("product.xml", FileMode.Open, FileAccess.Read);

xmldoc.Load(fs);

xmlnode = xmldoc.GetElementsByTagName("Product");

for (i = 0; i <= xmlnode.Count - 1; i++)

{

xmlnode[i].ChildNodes.Item(0).InnerText.Trim();

str = xmlnode[i].ChildNodes.Item(0).InnerText.Trim() + " " + xmlnode[i].ChildNodes.Item(1).InnerText.Trim() + " " + xmlnode[i].ChildNodes.Item(2).InnerText.Trim();

MessageBox.Show (str);

}

}

}

}

**Reading Xml with XmlReader**

**XmlReader** opens and parses **XML** files. It is a faster and less memory consuming alternative. It lets you run through the XML string one element at a time, while allowing you to look at the value, and then moves on to the next **XML** element. It provides a lower-level abstraction over the XML file structure.

C# XML Handling

XmlReader xReader = XmlReader.Create(new StringReader(xmlNode));

while (xReader.Read())

{

switch (xReader.NodeType)

{

case XmlNodeType.Element:

listBox1.Items.Add("<" + xReader.Name + ">");

break;

case XmlNodeType.Text:

listBox1.Items.Add(xReader.Value);

break;

case XmlNodeType.EndElement:

listBox1.Items.Add("");

break;

}

}

**How to search in a XML file**

**XML** files are made up of tags that contains information. The .Net technology is widely supported XML file format. Also the Dataset in **ADO.NET** uses XML format as its internal storage format.

The following C# source code shows how to search an item in a XML file using Dataset . Here Dataset using an XmlReader for read the content of the file. Locate the XML file using **XmlReader** and pass the XmlReader as argument of Dataset. By using the Dataset , search the product Product2 in the file Product.XML with the help of DataView.

using System;

using System.Data;

using System.Windows.Forms;

using System.Xml;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

XmlReader xmlFile ;

xmlFile = XmlReader.Create("Product.xml", new XmlReaderSettings());

DataSet ds = new DataSet();

DataView dv ;

ds.ReadXml(xmlFile);

dv = new DataView(ds.Tables[0]);

dv.Sort = "Product\_Name";

int index = dv.Find("Product2");

if (index == -1)

{

MessageBox.Show ("Item Not Found");

}

else

{

MessageBox.Show(dv[index]["Product\_Name"].ToString() + " " + dv[index]["Product\_Price"].ToString());

}

}

}

}

**XML** is a platform independent language, so the information formatted in XML can be used in any other platforms (Operating Systems). Once we create an XML file in one platform it can be used in other platforms also. The .Net technology is widely supported XML file format. Also the Dataset in **ADO.NET** uses XML format as its internal storage format.

Here we are going to filter an XML file content and store the result in a newly created XML file through C#. We have an XML file Product.XML , and it has a field Product\_Price. We are giving a search criteria like the **Product\_Price >= 3000** and store the result in a newly created XML file **Result.XML**

using System;

using System.Data;

using System.Windows.Forms;

using System.Xml;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

XmlReader xmlFile ;

xmlFile = XmlReader.Create("Product.xml", new XmlReaderSettings());

DataSet ds = new DataSet();

DataView dv ;

ds.ReadXml(xmlFile);

dv = new DataView(ds.Tables[0], "Product\_price > = 3000", "Product\_Name", DataViewRowState.CurrentRows);

dv.ToTable().WriteXml("Result.xml");

MessageBox.Show ("Done");

}

}

}

**How to insert data from XML to database**

**XML** is a general purpose tag based language and very easy to transfer and store data across applications. The .Net technology is widely supported XML file format. The [.Net Framework](http://vb.net-informations.com/framework/what_is_net_framework.htm) provides the Classes for read, write, and other operations in XML formatted files . Moreover the Dataset in **ADO.NET** uses XML format as its internal storage format.

Here we are going to insert the values of an XML file to a Database Table using **SQL Insert Command** . Here the Dataset using an XmlReader for read the content of the XML file - Product.XML . Locate the XML file using XmlReader and pass the **XmlReader** as argument of Dataset. Also establish a connection to the Database using a connectionstring . After getting the data from XML file to the **Dataset** , we can loop through the dataset values and use insert command to add the values to the Product table in the Databse.

using System;

using System.Data;

using System.Windows.Forms;

using System.Xml;

using System.Data.SqlClient;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string connetionString = null;

SqlConnection connection;

SqlCommand command ;

SqlDataAdapter adpter = new SqlDataAdapter();

DataSet ds = new DataSet();

XmlReader xmlFile ;

string sql = null;

int product\_ID = 0;

string Product\_Name = null;

double product\_Price = 0;

connetionString = "Data Source=servername;Initial Catalog=databsename;User ID=username;Password=password";

connection = new SqlConnection(connetionString);

xmlFile = XmlReader.Create("Product.xml", new XmlReaderSettings());

ds.ReadXml(xmlFile);

int i = 0;

connection.Open();

for (i = 0; i <= ds.Tables[0].Rows.Count - 1; i++)

{

product\_ID = Convert.ToInt32(ds.Tables[0].Rows[i].ItemArray[0]);

Product\_Name = ds.Tables[0].Rows[i].ItemArray[1].ToString();

product\_Price = Convert.ToDouble(ds.Tables[0].Rows[i].ItemArray[2]);

sql = "insert into Product values(" + product\_ID + ",'" + Product\_Name + "'," + product\_Price + ")";

command = new SqlCommand(sql, connection);

adpter.InsertCommand = command;

adpter.InsertCommand.ExecuteNonQuery();

}

connection.Close();

MessageBox.Show("Done .. ");

}

}

}