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

*<Books>*

*<Book>*

*<Title>Programming Microsoft ASP.NET 4</Title>*

*<ISBN>9780735643383</ISBN>*

*<Author>Dino Esposito</Author>*

*</Book>*

*<Book>*

*<Title>Microsoft Visual C# 2010 Step by Step</Title>*

*<ISBN>0735626707</ISBN>*

*<Author>John Sharp</Author>*

*</Book>*

*<Book>*

*<Title>Programming Microsoft ADO.NET Core Reference</Title>*

*<ISBN>073562206X</ISBN>*

*<Author>David Sceppa</Author>*

*</Book>*

*</Books>*

Here is the C# code to read the above XML file which is stored in the root application folder in the attached example and display the data on the asp.net page.

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Web;*

*using System.Web.UI;*

*using System.Web.UI.WebControls;*

*using System.Xml;*

*using System.Text;*

*public partial class \_Default : System.Web.UI.Page*

*{*

*protected void Page\_Load(object sender, EventArgs e)*

*{*

*readXML();*

*}*

*private void readXML()*

*{*

*XmlDocument doc = new XmlDocument();*

*//Load XML from the file into XmlDocument object*

*doc.Load(Server.MapPath("~/books.xml"));*

*XmlNode root = doc.DocumentElement;*

*StringBuilder sb = new StringBuilder();*

*//Select all nodes with the tag Book*

*XmlNodeList nodeList = root.SelectNodes("Book");*

*//Loop through each node under the node “Book”*

*foreach (XmlNode node in nodeList)*

*{*

*sb.Append("Title: ");*

*//Select the text from a single node, “Title” in this case*

*sb.Append(node.SelectSingleNode("Title").InnerText);*

*sb.Append("  
ISBN: ");*

*sb.Append(node.SelectSingleNode("ISBN").InnerText);*

*sb.Append("  
Author: ");*

*sb.Append(node.SelectSingleNode("Author").InnerText);*

*sb.Append("  
  
 ");*

*}*

*Response.Write(sb.ToString());*

*}*

*}*

* <videos>
* <video>
* <title>The Distinguished Gentleman</title>
* <director>Jonathan Lynn</director>
* <length>112 Minutes</length>
* <format>DVD</format>
* <rating>R</rating>
* </video>
* <video>
* <title>Her Alibi</title>
* <director>Bruce Beresford</director>
* <length>94 Mins</length>
* <format>DVD</format>
* <rating>PG-13</rating>
* </video>
* <video>
* <title>Other People's Money</title>
* <director>Alan Brunstein</director>
* <length>114 Minutes</length>
* <format>VHS</format>
* <rating>PG-13</rating>
* </video>
* </videos>

* + [Reply](http://forums.asp.net/post/set/43/1322024/2630042)

[](http://forums.asp.net/members/nikki_doer_of_things.aspx)

[**nikki\_doer\_o...**](http://forums.asp.net/members/nikki_doer_of_things.aspx)

Contributor

6863 Points

[1097 Posts](http://forums.asp.net/search?q=authors%3A(nikki_doer_of_things)&s=postdate&d=dsc)

**Re: Read XML file and Update C#. How to do it simple?**

Sep 18, 2008 01:57 PM|nikki\_doer\_of\_things|[LINK](http://forums.asp.net/post/2630042.aspx)

This is just sample code. For a real system, you'd want error checking, a formview or something for data entry, etc.

Example ASP Page

    <form id="form1" runat="server">  
    <div>  
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False">  
            <Columns>  
                <asp:BoundField HeaderText="Title" DataField="innerText" />  
            </Columns>  
        </asp:GridView>  
        <br />  
        <br />  
        Add Video<br />  
        Title&nbsp;  
        <asp:TextBox ID="txtTitle" runat="server"></asp:TextBox><br />  
        Director&nbsp;  
        <asp:TextBox ID="txtDirector" runat="server"></asp:TextBox><br />  
        Length&nbsp;  
        <asp:TextBox ID="txtLength" runat="server"></asp:TextBox><br />  
        Format&nbsp;  
        <asp:TextBox ID="txtFormat" runat="server"></asp:TextBox><br />  
        Rating&nbsp;  
        <asp:TextBox ID="txtRating" runat="server"></asp:TextBox><br />  
        <asp:Button ID="btnAdd" runat="server" OnClick="btnAdd\_Click" Text="Add" /></div>  
    </form>

Code behind

using System.Xml;  
  
public partial class Default2 : System.Web.UI.Page  
{  
    XmlDocument vidDocument = new XmlDocument();  
    protected void Page\_Load(object sender, EventArgs e)  
    {          
        vidDocument.Load(Server.MapPath("~/Videos.xml"));  
  
        // this is how you get all the video nodes, perhaps to bind them or iterate over them  
        XmlNodeList videos = vidDocument.SelectNodes("/videos/video");  
  
        // this would select all title elements  
        XmlNodeList titles = vidDocument.GetElementsByTagName("title");  
  
        // this binds the gridview to the title text to display  
        GridView1.DataSource = titles;  
        GridView1.DataBind();  
          
    }  
    protected void btnAdd\_Click(object sender, EventArgs e)  
    {  
        // the easiest way to create a new node to add to a document is to just clone an existing node  
        XmlNode node = vidDocument.SelectNodes("/videos/video")[0];  
        XmlNode newnode = node.CloneNode(true);  
        // update element values for the new node  
        newnode.SelectSingleNode("title").InnerText = txtTitle.Text;  
        newnode.SelectSingleNode("director").InnerText = txtDirector.Text;  
        newnode.SelectSingleNode("length").InnerText = txtLength.Text;  
        newnode.SelectSingleNode("format").InnerText = txtFormat.Text;  
        newnode.SelectSingleNode("rating").InnerText = txtRating.Text;  
        // append the new node to the document  
        vidDocument.DocumentElement.AppendChild(newnode);  
  
        // update the gridview to refresh display  
        XmlNodeList titles = vidDocument.GetElementsByTagName("title");  
        GridView1.DataSource = titles;  
        GridView1.DataBind();  
  
    }  
}

This article describes how to use the **XmlTextReader** class to read Extensible Markup Language (XML) from a file. **XmlTextReader** provides direct parsing and tokenizing of XML and implements the XML 1.0 specification as well as the namespaces in the XML specification from the World Wide Web Consortium (W3C). This article provides fast, tokenized stream access to XML rather than using an object model such as the XML Document Object Model (DOM).

Requirements

The following list outlines the recommended hardware, software, network infrastructure, and service packs that you need:

* Microsoft Visual Studio 2005 or Microsoft Visual Studio .NET

This article assumes that you are familiar with the following topics:

* XML terminology
* Creating and reading an XML file

How to read XML from a file

This example uses a file named Books.xml. You can create your own Books.xml file or use the sample file that is included with the .NET Software Development Kit (SDK) QuickStarts in the following folder:

\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Samples\QuickStart\Howto\Samples\Xml\Transformxml\Cs

You must copy Books.xml to the \Bin\Debug folder, which is located under the folder in which you create this project. Books.xml is also available for download. See to the "[References](http://support.microsoft.com/kb/307548#6)" section for the download location.

1. Start Visual Studio 2005 or Visual Studio .NET.
2. Create a new Visual C# Console Application. You proceed directly to the "[Complete code listing](http://support.microsoft.com/kb/307548#4)" section or continue through these steps to build the application.
3. Make sure that the project contains a reference to the **System.Xml.dll** assembly.
4. Specify the **using** directive on the **System.Xml** namespace so that you are not required to qualify **XmlTextReader** declarations later in your code. You must use the **using** directive before any other declarations.
5. using System.Xml;

1. Create an instance of an **XmlTextReader** object, and populate it with the XML file. Typically, the **XmlTextReader** class is used if you need to access the XML as raw data without the overhead of a DOM; thus, the **XmlTextReader** class provides a faster mechanism for reading XML. The **XmlTextReader** class has different constructors to specify the location of the XML data. The following code creates an instance of the **XmlTextReader** class and loads the Books.xml file. Add the following code to the Main procedure of Class1.
2. XmlTextReader reader = new XmlTextReader ("books.xml");

1. Read through the XML. (Note that this step demonstrates an outer "while" loop, and the next two steps demonstrate how to use that loop to read the XML.) After you create the **XmlTextReader** object, use the **Read** method to read the XML data. The **Read** method continues to move through the XML file sequentially until it reaches the end of the file, at which point the **Read** method returns a value of "False."
2. while (reader.Read())
3. {
4. // Do some work here on the data.
5. Console.WriteLine(reader.Name);
6. }
7. Console.ReadLine();

1. Inspect the nodes. To process the XML data, each record has a node type that can be determined from the **NodeType** property. The **Name** and **Value** properties return the node name (the element and attribute names) and the node value (the node text) of the current node (or record). The **NodeType** enumeration determines the node type. The following sample code displays the name of the elements and the document type. Note that this sample ignores element attributes.
2. while (reader.Read())
3. {
4. switch (reader.NodeType)
5. {
6. case XmlNodeType.Element: // The node is an element.
7. Console.Write("<" + reader.Name);
8. Console.WriteLine(">");
9. break;
10. case XmlNodeType.Text: //Display the text in each element.
11. Console.WriteLine (reader.Value);
12. break;
13. case XmlNodeType. EndElement: //Display the end of the element.
14. Console.Write("</" + reader.Name);
15. Console.WriteLine(">");
16. break;
17. }
18. }

1. Inspect the attributes. Element node types can include a list of attribute nodes that are associated with them. The **MovetoNextAttribute** method moves sequentially through each attribute in the element. Use the **HasAttributes** property to test whether the node has any attributes. The **AttributeCount** property returns the number of attributes for the current node.
2. while (reader.Read())
3. {
4. switch (reader.NodeType)
5. {
6. case XmlNodeType.Element: // The node is an element.
7. Console.Write("<" + reader.Name);
8. while (reader.MoveToNextAttribute()) // Read the attributes.
9. Console.Write(" " + reader.Name + "='" + reader.Value + "'");
10. Console.WriteLine(">");
11. break;
12. case XmlNodeType.Text: //Display the text in each element.
13. Console.WriteLine (reader.Value);
14. break;
15. case XmlNodeType. EndElement: //Display the end of the element.
16. Console.Write("</" + reader.Name);
17. Console.WriteLine(">");
18. break;
19. }
20. }

1. Save and close your project.

Complete code listing

using System;

using System.Xml;

namespace ReadXMLfromFile

{

/// <summary>

/// Summary description for Class1.

/// </summary>

class Class1

{

static void Main(string[] args)

{

XmlTextReader reader = new XmlTextReader ("books.xml");

while (reader.Read())

{

switch (reader.NodeType)

{

case XmlNodeType.Element: // The node is an element.

Console.Write("<" + reader.Name);

Console.WriteLine(">");

break;

case XmlNodeType.Text: //Display the text in each element.

Console.WriteLine (reader.Value);

break;

case XmlNodeType.EndElement: //Display the end of the element.

Console.Write("</" + reader.Name);

Console.WriteLine(">");

break;

}

}

Console.ReadLine();

}

}

}

Sample output

<bookstore>

<book>

<title>

The Autobiography of Benjamin Franklin

</title>

<author>

<first-name>

Benjamin

</first-name>

<last-name>

Franklin

</last-name>

</author>

<price>

8.99

</price>

</book>

<book>

<title>

The Confidence Man

</title>

<author>

<first-name>

Herman

</first-name>

<last-name>

Melville

</last-name>

</author>

<price>

11.99

</price>

</book>

<book>

<title>

The Gorgias

</title>

<author>

<name>

Plato

</name>

</author>

<price>

9.99

</price>

</book>

</bookstore>