

Chapter No.8

XML

In

ASP.Net



Today we will learn,

- ✓ What is XML
- ✓ Difference between XML and HTML
- ✓ Filling the GridView Control from XML
- ✓ Searching the XML File
- ✓ Adding New Record in XML File
- ✓ Delete a Record from XML File
- ✓ Update a Record in XML File
- ✓ Storing XML file in Database
- ✓ Retrieving Database table in XML File

8.1 What is XML

XML stands for Extensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). It is a very widely used format for exchanging data, mainly because it's easy readable for both humans and machines. XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. It is important to understand that XML is not a substitute for HTML. In the future development of the Web, XML will be the main language to describe the structure and the Web data, and the HTML language will be responsible for displaying the data.

Moreover, as we know that a programming language consists of grammar rules and its own vocabulary which is used to create computer programs. These programs instructs computer to perform specific tasks. XML does not qualify to be a programming language as it does not perform any computation or algorithms. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML.

1.1 Characteristics of XML

There are three important characteristics of XML that make it useful in a variety of systems and solutions:

XML is extensible: XML allows you to create your own self-descriptive tags, or language, that suits your application.

XML carries the data, does not present it: XML allows you to store the data irrespective of how it will be presented.

XML is a public standard: XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

8.2 Where XML is Used

- ✓ XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- ✓ XML can be used to exchange the information between organizations and systems.
- ✓ XML can be used for offloading and reloading of databases.

- ✓ XML can be used to store and arrange the data, which can customize your data handling needs.
- ✓ XML can easily be merged with style sheets to create almost any desired output.
- ✓ Virtually, any type of data can be expressed as an XML document.

8.3 XML Document Structure

XML files are made up of tags that contain 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. i.e.

< Header > Header Information < /Header >

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

*** XML is case sensitive**

Example:

< Header > is not same as < HeadeR> .

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

Example :

< first-tag >< second-tag > Data here < /second-tag > < /first-tag >

8.4 ASP.Net and XML

XML is a platform independent language, so the information formatted in XML can be use in any other platforms (Operating Systems) . The .Net technology is widely supported XML file format. The .Net Framework provides the Classes for read, write, and other operations in XML formatted files. We can read an XML file in several ways depends on our requirement. These methods are included:

1. Reading XML File using XmlReader
2. Reading XML File using XmlDocument
3. Reading XML File using LINQ
4. Reading XML File using XmlTextReader

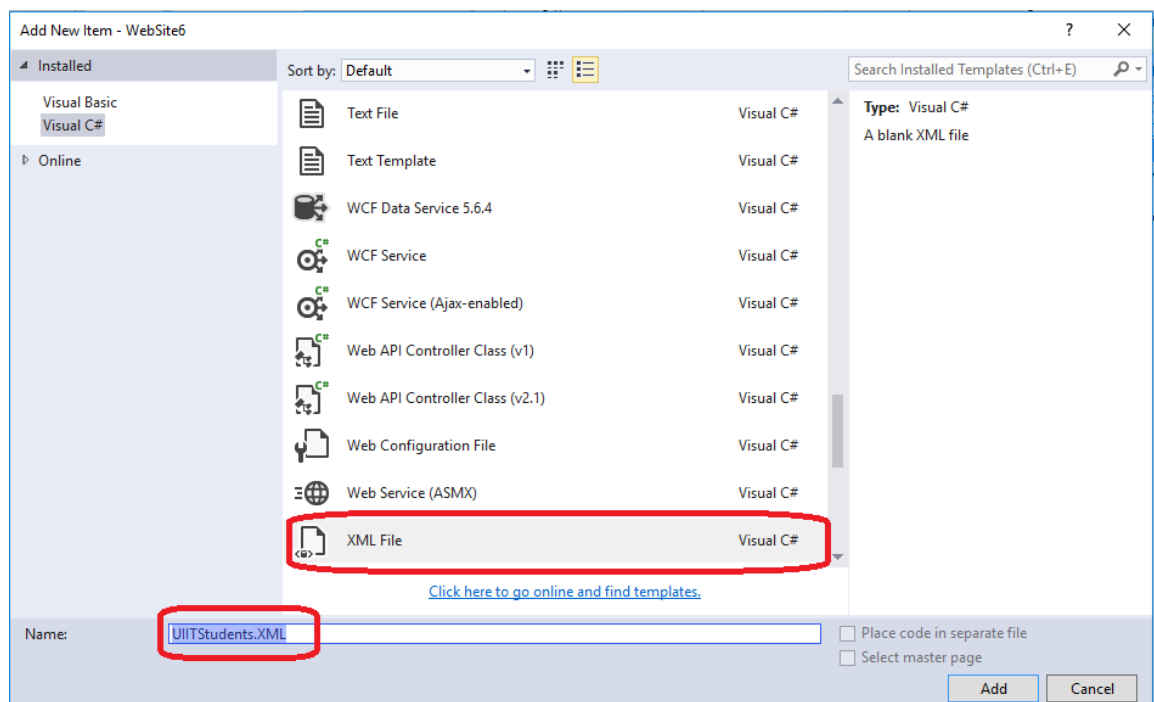
5. Reading XML File using Using XPath with XmlDocument

For example, in the following code we are going to read the XML file using XmlReader method

In this example, we are going to fill the GridView control with an XML file. For this example, we consider the XML file **UIITStudents.XML**.

To fill the GridView from this XML File follow the steps

1. Create a new Web Site rename the Web Site as **XMLInASPNetDeom**
2. Right Click on the Web Site Name Add ➔ New Item ➔ XML File as shown



3. Add few **Records** to XML File as shown

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

  <Student>
    <SID>101</SID>
    <SName>Muhammad Bilal</SName>
    <Sclass>BSCS</Sclass>
    <Semester>7th</Semester>
    <OMarks>400</OMarks>
  </Student>

  <Student>
    <SID>102</SID>
```

```

    <SName>Abu Bakar</SName>
    <Sclass>BSIT</Sclass>
    <Semester>6th</Semester>
    <OMarks>340</OMarks>
</Student>

<Student>
    <SID>103</SID>
    <SName>Naqash</SName>
    <Sclass>BSCS</Sclass>
    <Semester>6th</Semester>
    <OMarks>490</OMarks>
</Student>
</UIIT>

```

4. Design an ASP.NET form. Drag and drop a GridView on the web form and rename the GridView with **gvStudents**.
5. Open the Code view and locate the **Page_Load()** Event. Add the following code in the **Page_Load()** event.

```

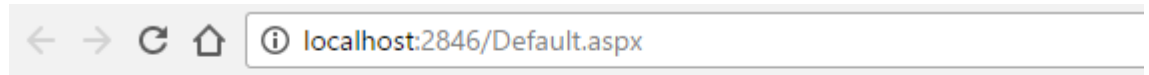
1  using System;
2  using System.Web;
3  using System.Data;
4  public partial class _Default : System.Web.UI.Page
5  {
6
7      DataSet ds;
8      protected void Page_Load(object sender, EventArgs e)
9      {
10         ds = new DataSet();
11         ds.ReadXml(Server.MapPath("UIITStudents.xml"));
12         GridView1.DataSource = ds.Tables[0];
13         GridView1.DataBind();
14     }
15 }
16

```

Server.MapPath("XMLFileName.XML") : This method is used to return the Physical Path of the file correspond to the specified virtual path.

In the code above, we read an XML file using a DataSet. Here Dataset is using an XmlReader for read the content of the file. Locate the XML file using XmlReader and pass the XmlReader as argument of Dataset.

6. Save and Browse the Web Site.



SID	SName	Sclass	Semester	OMarks
101	Muhammad Bilal	BSCS	7th	400
102	Abu Bakar	BSIT	6th	340
103	Naqash	BSCS	6th	490

We can read the XML file using other methods mentioned above. You can try the other methods by yourself.

8.5 Searching the XML File

In this example, we are going to search the records from the XML file. Design an ASP.NET web form. Rename the file as SearchXML

Design the Web Form as shown here

XML File Searching

Marks:

Add a Label after the Show Result in Table Button. Double Click on the button to create the

button click event as shown here.

```
0 references
protected void btnSearchXML_Click(object sender, EventArgs e)
{
}
}
```

Add the following code here:

```
36 protected void btnSearchXML_Click(object sender, EventArgs e)
37 {
38     ds = new DataSet();
39     ds.ReadXml(Server.MapPath("UIITStudents.xml"));
40     int marksSearchcriteria = Convert.ToInt16(txtMarks.Text);
41     string result= @"<Table width=100% border=3>
42     <tr>
43     <Td><h2>Name</td>
44     <Td><h2>Obt: Marks</h2></td>
45     <Td><h2>%age</h2></td>
46     <Td><h2>Class</h2></td>
47     </tr>";
48
49     for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
50     {
```

```

62         result += "</table>";
63         lblResult.Text = result;
64     }

```

The code is

```

protected void btnSearchXML_Click(object sender, EventArgs e)
{
    ds = new DataSet();
    ds.ReadXml(Server.MapPath("UIITStudents.xml"));
    int marksSearchcriteria = Convert.ToInt16(txtMarks.Text);
    string result= @"<Table width=100% border=3>
    <tr>
    <Td><h2>Name</td>
    <Td><h2>Obt: Marks</h2></td>
    <Td><h2>%age</h2></td>
    <Td><h2>Class</h2></td>
    </tr>";

    for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
    {
        if (Convert.ToInt16(ds.Tables[0].Rows[i]["OMarks"].ToString()) >
            marksSearchcriteria)
        {
            double per =
            (Convert.ToInt16(ds.Tables[0].Rows[i]["OMarks"].ToString())*100.0)/600.0;
            result+= @"<tr><Td>" + ds.Tables[0].Rows[i]["SName"].ToString() +
            "</td><Td>" + ds.Tables[0].Rows[i]["OMarks"].ToString() +
            "</td><Td>" + per.ToString() +
            "</td><Td>" + ds.Tables[0].Rows[i]["Sclass"].ToString() + "</td></tr>";
        }
    }
    result += "</table>";
    lblResult.Text = result;
}

```

Save and run the Web Site.

XML File Searching

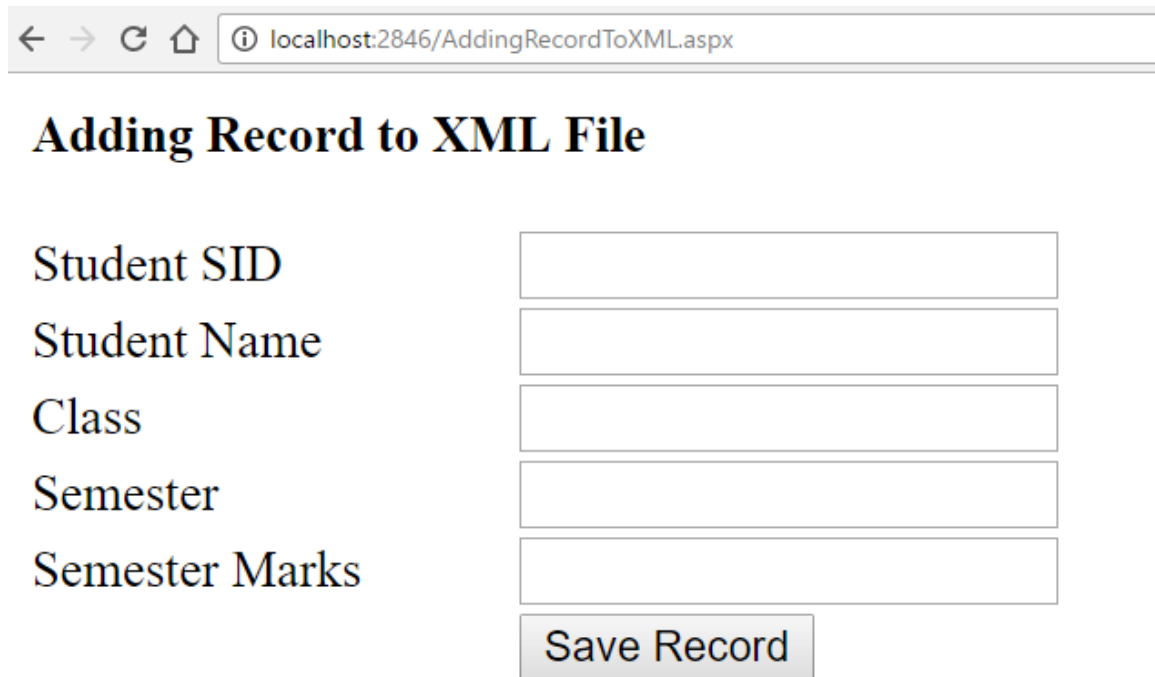
Marks:

Name	Obt: Marks	%age	Class
Muhammad Bilal	400	66.6666666666667	BSCS
Abu Bakar	340	56.6666666666667	BSIT
Naqash	490	81.6666666666667	BSCS

Therefore, the results shown here is the records of those students whose obtained marks are greater than the user supplied value. Similarly, we can search the XML File using other attributes such as Class, Student Name etc.

8.6 Adding New Record in XML File

We will use the UIITStudent.XML file created earlier in this chapter. Add a New ASP.Net form to your Web Site created in this chapter. Rename the form as AddingRecordToXML file. Therefore, design the form as shown below



← → ↻ 🏠 ⓘ localhost:2846/AddingRecordToXML.aspx

Adding Record to XML File

Student SID

Student Name

Class

Semester

Semester Marks

Now, double click on the Save Record button to create its click event and add the following code in the event as shown here

```
protected void Button1_Click(object sender, EventArgs e)
{
    XmlDocument xmldoc = new XmlDocument();
    xmldoc.Load(Server.MapPath("UIITStudents.xml"));

    XmlElement Student = xmldoc.CreateElement("Student");

    XmlElement SID = xmldoc.CreateElement("SID");
    SID.InnerText = txtSID.Text;

    XmlElement stdName = xmldoc.CreateElement("SName");
    stdName.InnerText = txtName.Text;

    XmlElement stdClass = xmldoc.CreateElement("Sclass");
    stdClass.InnerText = txtClass.Text;

    XmlElement Semester = xmldoc.CreateElement("Semester");
    Semester.InnerText = txtSemester.Text;

    XmlElement obtMarks = xmldoc.CreateElement("OMarks");
    obtMarks.InnerText = txtOmMarks.Text;
}
```

```
Student.AppendChild(SID);
Student.AppendChild(stdName);
Student.AppendChild(stdClass);
Student.AppendChild(Semester);
Student.AppendChild(obtMarks);

xmldoc.DocumentElement.AppendChild(Student);
xmldoc.Save(Server.MapPath("UIITStudents.xml"));
Response.Write("New Student is saved to database
Sucessfully...");
}
```

Explanations:

```
XmlDocument xmldoc = new XmlDocument();
xmldoc.Load(Server.MapPath("UIITStudents.xml"));
```

This will load the file into XmlDocument object named xmldoc.

Next, step is to create the an object of **XmlElement** named **Student** (This is the first node in the XML File of **UIITStudent.XML** file) using the code

```
XmlElement Student = xmldoc.CreateElement("Student");
```

In the Next two lines, we have created an XmlElement named **SID** which is the first element in the **UIITStudent.XML** file

```
XmlElement SID = xmldoc.CreateElement("SID");
```

Now, we add the information for the SID element. This information is retrieved from the Textbox named txtSID

```
SID.InnerText = txtSID.Text;
```

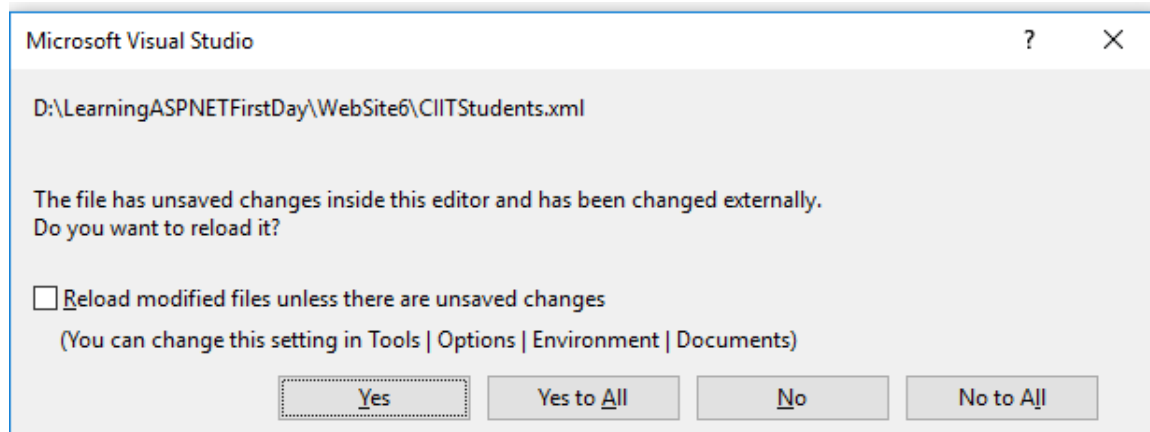
Repeating the same steps, we have added two lines for each of the element in the **UIITStudent.XML** file

Finally, Save the File. Browse the Web Form and add the sample data and press the Save Record button.

New Student is saved to database Sucessfully... Adding Record to XML File

Student SID	<input type="text" value="1001"/>
Student Name	<input type="text" value="Muhammad Imran"/>
Class	<input type="text" value="MCS"/>
Semester	<input type="text" value="3"/>
Semester Marks	<input type="text" value="450"/>
<input type="button" value="Save Record"/>	

Open the **UIITStudent.XML** file, you will notice



Press the **Yes To All** button. You will see that a new Student Record is saved to **UIITStudent.XML**, as shown

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <UIIT>
3      <Student>...</Student>
10     <Student>...</Student>
17     <Student>...</Student>
24     <Student>
25         <SID>1001</SID>
26         <SName>Muhammad Imran</SName>
27         <Sclass>MCS</Sclass>
28         <Semester>2</Semester>
29         <OMarks>450</OMarks>
30     </Student>
31 </UIIT>

```

8.7 Delete a Record from XML File

8.8 Update a Record in XML File

8.9 Difference between XML and HTML

HTML was designed for data display to focus the data look while XML was designed for storing and transporting data.

HTML is used for displaying web pages while XML does not used for displaying web pages.

HTML was invented in 1990 while XML was invented in 1996.

HTML is static type while MXL is dynamic type.

in XML author can create custom tags but in HTML custom tags are predefined.

XML can preserve white spaces while XML can't preserve white spaces.

HTML is presentation type language while XML is neither a presentation type language nor programming type.

HTML doesn't have strict rules of processing while XML has to follow strict rule otherwise processing file will be terminated.

These ASP.Net Tutorials are prepared only for the purpose of learning ASP.Net with C#.

Anyone can copy, print or reuse it in any format

Saif Ur Rehman Saifi

Assistant Professor, UIIT

PMAS Arid Agriculture University, Rawalpindi

If you found this tutorial helpful or you want to give us any valuable suggestions contact at

www.ASPUIIT.blogspot.com