STUDENT ACTIVITY 4.3: UNDERSTAND XML CLASSES IN THE .NET FRAMEWORK

MTA Course: 10754 Microsoft .NET Fundamentals
Topic: Understand XML classes in the .NET Framework
File name: 10754_Msft.NET_SA_4.3

Lesson Objective

4.3: Understand XML classes in the .NET Framework. *This objective may include but is not limited to:* understanding *XMLReader, XMLWriter,* and XML Schemas.

Resources, software, and additional files needed for this lesson:

- Microsoft® Visual Studio® 2010; students may also use Microsoft Visual C#® 2010 Express or Microsoft Visual Basic® 2010 Express, available free at *http://www.microsoft.com/express/Windows/.*

Events and Event Handlers

Directions to the student:

Follow the steps below to create a simple application that reads and writes Extensible Markup Language (XML).

Read and write XML

1. This assignment uses the project named XML Practice that you created during the Microsoft PowerPoint® presentation.
2. In the Solution Explorer, select the XML file you created, called Computers.xml. In the Properties panel, change the Copy to Output Directory property to Copy always.
3. Add the following controls to your form:
   a. A `ListBox` named `lstOutput`; this will be used to display the XML data.
   b. A `Button` named `btnRead` with the `Text` property set to "Read"; this will read and process the XML file.

c. A `Button` named `btnWrite` with the `Text` property set to "Write"; this will write new data to the XML file.

4. Access the source code of the Windows Form (View menu and select Code). Add the following to the `using` directives at the top of the file:

```
using System.Xml;
```

5. Create an event handler for the `btnRead Click` event. Create an instance of `XmlTextReader` by adding the following code to the event handler:

```
XmlTextReaderrdr = new XmlTextReader("computers.xml");
```

The constructor loads the specified file, in this case Computers.xml.

6. To iterate through the XML and display the contents in the `ListBox`, add the following code:

```
while (rdr.Read())
{
lstOutput.Items.Add(rdr.Name);
}
rdr.Close();
```

Execute the application and click the `Read` button. Notice that it displays the names of the XML elements but not the data itself. Note: if the Computers.xml file does not exist in the bin/Debug folder, you may get an error.

7. To make the output a little more clear, modify the loop by adding the following code:

```
while (rdr.Read())
{
    if (rdr.NodeType == XmlNodeType.Element)
    {
lstOutput.Items.Add("--------------------------------");
lstOutput.Items.Add("Element name = " + rdr.Name);
    }
    if (rdr.NodeType == XmlNodeType.Text)
    {
lstOutput.Items.Add("Value = " + rdr.Value);
    }
}
rdr.Close();
```

Test your code again and note how the output is organized.

8. Create an event handler for the `btnWrite Click` event. This time, you will need an instance of the `XmlTextWriter` class, as follows:

```
XmlTextWriterwrtr = new XmlTextWriter("computers.xml", null);
wrtr.Formatting = Formatting.Indented;
```

The constructor parameters specify the file name and encoding type; `null` sets the encoding type to UTF-8. The second line sets the output formatting.

9. Finally, the following code uses the method pairs discussed in the lesson to add the elements. It will use the same schema as the original XML file that you created:

```
wrtr.WriteStartDocument();
wrtr.WriteStartElement("ComputerSales");
wrtr.WriteStartElement("Computer");
wrtr.WriteAttributeString("name", "netbook");

wrtr.WriteElementString("ProcessorSpeed", "3.0 GHz");
wrtr.WriteElementString("RAM", "8Gb");
wrtr.WriteElementString("HDD", "1 Terabyte");

wrtr.WriteEndElement();
wrtr.WriteEndElement();
wrtr.WriteEndDocument();

wrtr.Close();
```

10. Navigate to the bin/Debug folder for this project and open Computers.xml in Notepad (or another text editor) to see the XML output that this code created.