

Lab 15 – XML

Objective

You will write code to create a local XML file using the *XmlDocument* object. You will then write additional code to read the XML file from your local *c:* drive and then search the XML file for a specific XML element.

Step-by-Step Instructions

1. Start Visual Studio 2005
2. Create a new web site. **File** → **New** → **Web Site** (Or, **File** → **New Web Site...**)
3. Select the following options:

Setting	Value
Visual Studio installed templates	ASP.NET Web Site
Location	File System – C:\Code\Lab15
Language	Visual Basic

4. Open the *Default.aspx* page in **Design** mode.
5. Drag a **Button** onto the page.
6. Change the *ID* property of the button to “*cmdCreateXML*”
7. Change the *Text* property of the button to “*Create XML*”
8. Create the click event of the *Create XML* button.

9. Add the following code to the click event of the *Create XML* button.

This code will define an *XmlDocument* object and then add the *XmlDeclaration* followed by the root element of the *ProductList.xml* file. Finally, the code will save the file to the root of your *c:* drive.

```
Imports System.Xml

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub btnCreateXML_Click(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles btnCreateXML.Click

        Dim doc As New XmlDocument()
        Dim RootElement As XmlElement

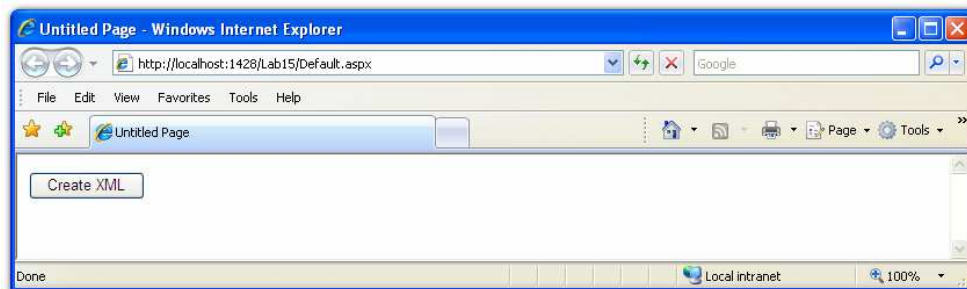
        Dim Declaration As XmlDeclaration
        Declaration = doc.CreateXmlDeclaration("1.0", Nothing, "yes")

        doc.InsertBefore(Declaration, doc.DocumentElement)

        RootElement = doc.CreateElement("ProductList")
        doc.InsertAfter(RootElement, Declaration)

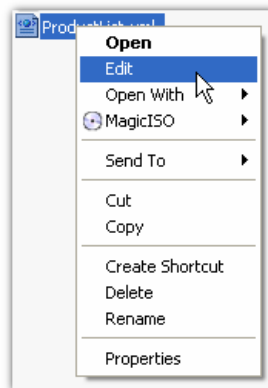
        doc.Save("c:\ProductList.xml")
    End Sub
End Class
```

10. Run the application.



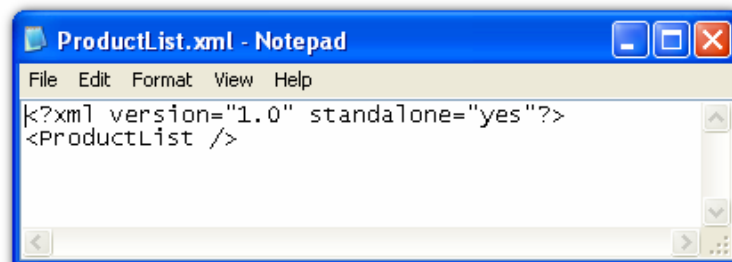
11. Click on the **Create XML** button to execute the code that creates the XML file.
12. Close the browser to stop running the application.
13. Open Windows Explorer.
14. Navigate to the root of your *c:* drive and locate the *ProductList.xml* file.

15. Right-click on the file and choose **Edit**



16. The *ProductList.xml* file should now be shown.

As you will see, the declaration of the file and the root node were created by the code in the click event of the button. Since nothing was placed in the root node, it is simply added as `<ProductList />`.



17. Add the following code to the *CreateXML* button.

This new code will add the *Product* element that will have two attributes: *ID* and *Name*. Then, the code will add the *Price* attribute.

```
Protected Sub btnCreateXML_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles btnCreateXML.Click

    Dim doc As New XmlDocument()
    Dim RootElement As XmlElement
    Dim ProductElement As XmlElement
    Dim PriceElement As XmlElement
    Dim ProductAttribute As XmlAttribute

    Dim Declaration As XmlDeclaration
    Declaration = doc.CreateXmlDeclaration("1.0", Nothing, "yes")

    doc.InsertBefore(Declaration, doc.DocumentElement)

    RootElement = doc.CreateElement("ProductList")
    doc.InsertAfter(RootElement, Declaration)

    ProductElement = doc.CreateElement("Product")
    RootElement.AppendChild(ProductElement)

    ProductAttribute = doc.CreateAttribute("ID")
    ProductAttribute.Value = "1"
    ProductElement.SetAttributeNode(ProductAttribute)

    ProductAttribute = doc.CreateAttribute("Name")
    ProductAttribute.Value = "Chair"
    ProductElement.SetAttributeNode(ProductAttribute)

    PriceElement = doc.CreateElement("Price")
    PriceElement.InnerText = "49.99"
    ProductElement.AppendChild(PriceElement)

    doc.Save("c:\ProductList.xml")
End Sub
```

18. Run the application.

19. Click the **Create XML** button to create the updated XML file.

20. Close the browser to exit the application.

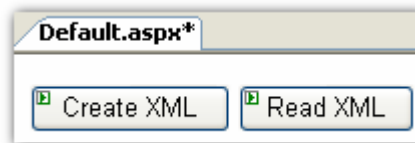
21. Locate the *c:\ProductList.xml* file in Windows Explorer.

22. Right-click on the file and choose **Edit**.

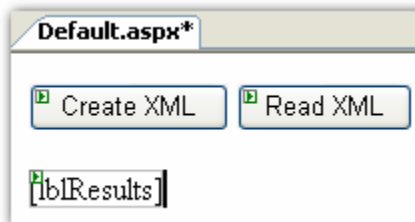
As you will see, the *Product* element has now been created with two attributes: *ID* and *Name*. Also, the *Price* element has been added under the *Product* element.

```
<?xml version="1.0" standalone="yes"?>
<ProductList>
  <Product ID="1" Name="Chair">
    <Price>49.99</Price>
  </Product>
</ProductList>
```

23. Return to the *Default.aspx*.
24. Drag a **Button** control onto the page next to the **Create XML** button.
25. Change the *ID* property of the button to “*cmdReadXML*”
26. Change the *Text* property of the button to “*Read XML*”



27. Drag a **Label** control onto the web page underneath the two buttons.
28. Change the *ID* property of the label to “*lblResults*”
29. Clear the *Text* property of the label control.



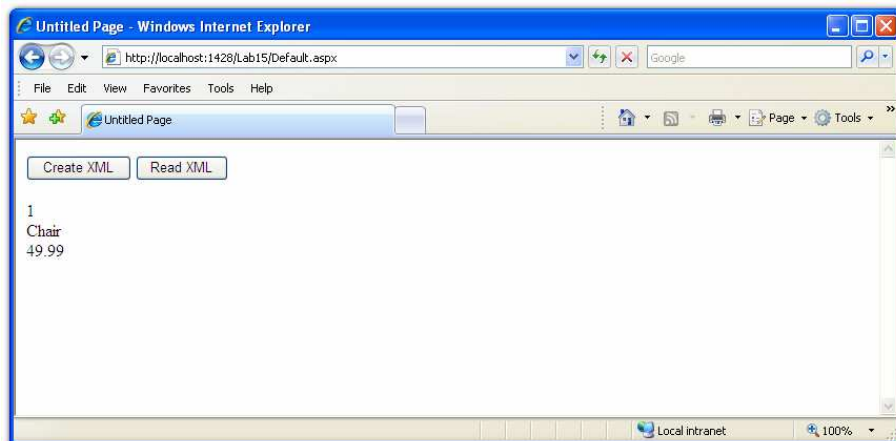
30. Create the click event of the *ReadXML* button
31. Add the following code to the click event of the *ReadXML* button.

This code will open the XML document and loop through the nodes to display them on the web page.

```
Protected Sub cmdReadXML_Click(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles cmdReadXML.Click  
  
    Dim doc As New XmlDocument()  
    doc.Load("c:\ProductList.xml")  
  
    Dim Element As XmlElement  
  
    For Each Element In doc.DocumentElement.ChildNodes  
        lblResults.Text &= Val(Element.GetAttribute("ID")) & "<br />"  
        lblResults.Text &= Element.GetAttribute("Name") & "<br />"  
        lblResults.Text &= Val(Element.ChildNodes(0).InnerText())  
    Next  
  
End Sub
```

32. Run the application.

33. Click the **Read XML** button. The page should now look similar to the one below which displays the two attributes (*ID* and *Name*) as well as the *Price*.



34. Close the browser to stop running the application.

35. Open the *ProductList.xml* file.

36. Edit the file to add multiple *Product* nodes as shown below.

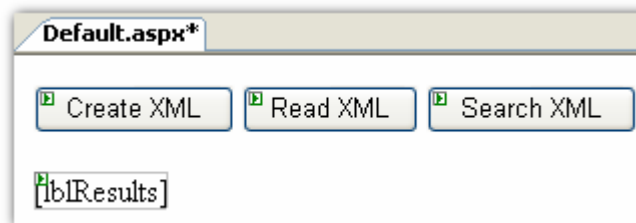
Note: this can be done by simple cut-and-pasting and then modifying the *ID*, *Name*, and *Prices* as shown below:

```
<?xml version="1.0" standalone="yes"?>
<ProductList>
  <Product ID="1" Name="Chair">
    <Price>49.99</Price>
  </Product>
  <Product ID="2" Name="Chair2">
    <Price>29.99</Price>
  </Product>
  <Product ID="3" Name="Chair3">
    <Price>19.99</Price>
  </Product>
  <Product ID="4" Name="Chair4">
    <Price>9.99</Price>
  </Product>
  <Product ID="5" Name="Chair5">
    <Price>4.99</Price>
  </Product>
</ProductList>
```

37. Drag a new **Button** control onto the page next to the **Read XML** button.

38. Change the *ID* property of the button to “*cmdSearchXML*”

39. Change the *Text* property of the button to “*Search XML*”



40. Create the click event for the **Search XML** button.

41. Add the following code to the event:

Note: This code will open the *ProductList.xml* file and then call the *GetElementsByTagName* method of the *XmlDocument* object to search for all the *Price* nodes. It will then loop through the price nodes and display the values.

```
Protected Sub cmdSearchXML_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles cmdSearchXML.Click

    Dim doc As New XmlDocument()
    doc.Load("c:\ProductList.xml")
    Dim Results As XmlNodeList

    Results = doc.GetElementsByTagName("Price")

    lblResults.Text = "Found: " & _
        Results.Count.ToString() & " matches.<br />"

    For Each Result As XmlNode In Results
        lblResults.Text &= Result.FirstChild.Value & "<br />"
    Next
End Sub
```

42. Run the application

43. Click on the **Search XML** button. As you will see, the XML file is searched for all the *Price* elements. Once found, the values are displayed on the web page.

