Tutorial – LINQ

In this tutorial we are going to demonstrate how to use Linq to XML in C# to create an XML document from in-memory objects. It also demonstrates how to write Linq to XML queries in C#.

Taking the Sprite Sheet editor as inspiration, this program creates an array of several simple class objects and uses embedded Linq queries to build the child XML elements.

Creating the Project:

Create a new C# Windows Forms application.

There is no need to add any controls to the form. For the purposes of this tutorial all code will be placed in the form's constructor so that the XML document is created upon application launch.

If you have yet to add XML output to your assignment, you could use your assignment application for this tutorial instead.

Creating the In-Memory Objects:

Open the .cs file for Form1 and add the following simple class definitions.

```
    public class ImageDesc
    {
        public string Path { get; set; }
    }

    public class Sprite
    {
        public ImageDesc Image { get; set; }
        public Point Position { get; set; }
        public Point Size { get; set; }
    }
```

We will make an array of Sprite objects that will hold basic attribute data that describes the sprites in our sprite sheet.

These classes have been overly simplified for the sake of demonstration, but could easily be adapted for use in a non-trivial program.

Next we define a function to create an array containing our sprite data. It is from this array that we will make our XML file.

```
private static Sprite[] CreateObjects()
{
    Sprite[] sprites = new Sprite[] {
        new Sprite {
            Image = new ImageDesc() {
                Path = "lolcat.png"
            },
            Position = new Point(0, 0),
            Size = new Point(32, 32)
        },
        new Sprite {
            Image = new ImageDesc() {
                Path = "success_kid.png"
            },
            Position = new Point(34, 0),
            Size = new Point(32, 32)
        },
        new Sprite {
            Image = new ImageDesc() {
                Path = "philosoraptor.png"
            },
            Position = new Point(66, 0),
            Size = new Point(32, 32)
        },
    };

    return sprites;
}
```

This gives us an array of Sprite objects that we can query with Linq.

Creating the XML Document:

We create the XML document based on the in-memory objects. Each element is under the default XML namespace http://www.w3.org/1999/xhtml.

```
private static void CreateXML(Sprite[] sprites, string path)
{
    XDocument doc = new XDocument(
        // XML declaration
        new XDeclaration("1.0", "uft-8", "yes"),
        // The root element
        new XElement(xmlNamespace + "Sprites",
            // Embeded Linq query to build the child XML elements
            from s in sprites
            select new XElement(xmlNamespace + "Sprite",
                // Create the element's attribute
                new XAttribute("path", s.Image.Path),
                new XElement(xmlNamespace + "Position",
                // Create the element's attribute
                    new XAttribute("x", s.Position.X),
                    new XAttribute("y", s.Position.Y)
                ),
                new XElement(xmlNamespace + "Size",
                // Create the element's attribute
                    new XAttribute("width", s.Size.X),
                    new XAttribute("height", s.Size.Y)
                )
            )         )
    );

    // Save the XML document to file system
    doc.Save(path);
}
```

This almost completes our class. The only thing left is to define the xmlNamespace variable, and then call the two functions we've just defined from within the form's constructor.

Here is the complete code for the Form1 class:

```
public partial class Form1 : Form
{
    private static XNamespace xmlNamespace = "http://www.w3.org/1999/xhtml";

    public Form1()
    {
        InitializeComponent();

        string path = "spritesheet.xml";

        // create the in-memory objects, build the XML document based on
        // these objects
        Sprite[] sprites = CreateObjects();

        // create the XML document based on the in-memory obejcts
        CreateXML(sprites, path);

    }

    private static void CreateXML(Sprite[] sprites, string path)
    {
        XDocument doc = new XDocument(
            // XML declaration
            new XDeclaration("1.0", "uft-8", "yes"),
            // The root element
            new XElement(xmlNamespace + "Sprites",
                // Embeded Linq query to build the child XML elements
                from s in sprites
               select new XElement(xmlNamespace + "Sprite",
                    // Create the element's attribute
                    new XAttribute("path", s.Image.Path),
                    new XElement(xmlNamespace + "Position",
                    // Create the element's attribute
                        new XAttribute("x", s.Position.X),
                        new XAttribute("y", s.Position.Y)
                    ),
                    new XElement(xmlNamespace + "Size",
                    // Create the element's attribute
                        new XAttribute("width", s.Size.X),
                        new XAttribute("height", s.Size.Y)
                    )
                )
            )
        );

        // Save the XML document to file system
        doc.Save(path);
    }

    private static Sprite[] CreateObjects()
    {
        Sprite[] sprites = new Sprite[] {
            new Sprite {
                Image = new ImageDesc() {
                    Path = "lolcat.png"
                },
                Position = new Point(0, 0),
                Size = new Point(32, 32)
            },
            new Sprite {
                Image = new ImageDesc() {
                    Path = "success_kid.png"
                },
                Position = new Point(34, 0),
                Size = new Point(32, 32)
            },
            new Sprite {
                Image = new ImageDesc() {
                    Path = "philosoraptor.png"
                },
                Position = new Point(66, 0),
                Size = new Point(32, 32)
            },
        };

        return sprites;
    }
}
```

Reading and Modifying the XML:

We can also read in the XML file, search for a specific element, and then modify its value.

This time we use Linq to filter the elements in our XML document.

```
private static void ModifyXML(string path)
{
    // Load the XML document
    XDocument doc = XDocument.Load(path);

    // Retrieve the root element
    XElement element = doc.Root;

    // Retrieve the certain Sprite element
    XElement spriteElement = element.Elements(xmlNamespace + "Sprite").
        Where(s => s.Attribute("path").Value == "philosoraptor.png").Single();

    if (spriteElement != null)
    {
        // Retrieve the certain Project element
        var positionElement = spriteElement.Elements(xmlNamespace +
"Position").Single();
        if (positionElement != null)
        {
            // Modify the position element value
            positionElement.Attribute("y").Value = "10";
        }
    }

    // Save the XML document
    doc.Save(path);
}
```

After loading the XML document, we then use Linq to search for the 'philosoraptor' sprite by filtering the xml elements with a predicate.

If we get a match, we retrieve the Position element and modify its 'y' attribute. Finally we save the updated XML document.

Exercise:

Update the Sprite Sheet application you wrote for your assignment to use Linq to generate and query the XML document.

If you need more examples on the Linq to XML classes, the Microsoft Developer Library (https://msdn.microsoft.com) has an invaluable collection of sample code.