Parse Local XML file in Windows Phone

This article explains how to parse a local XML file. It also helps you to know how to parse XML once and use parsed data throughout the application.

Introduction

This code snippet helps you to know:

1. how to parse local xml file and represent parsed data in terms of objects with the help of XmlSerializer
2. how to use Singleton design pattern (in the given code I used it to avoid multiple time parsing of xml file)

Create one WindowsPhone Application

1. Add **states.xml** file in project and keep its build action property to Resource.

States.xml

```
<root>
  <states>
    <state>
      <id>1</id>
      <name>state1</name>
      <cities>
        <city>
          <cityid>1</cityid>
          <cityname>city1</cityname>
          <pincode>0010</pincode>
        </city>
        <city>
          <cityid>2</cityid>
          <cityname>city2</cityname>
          <pincode>0011</pincode>
        </city>
       </cities>
    </state>
    <state>
      <id>2</id>
      <name>state2</name>
      <cities>
        <city>
          <cityid>1</cityid>
          <cityname>city1</cityname>
          <pincode>0010</pincode>
        </city>
        <city>
          <cityid>2</cityid>
          <cityname>city2</cityname>
          <pincode>0011</pincode>
        </city>
       </cities>
    </state>
  </states>
</root>
```


(C) Copyright Nokia 2014. All rights reserved.

Now create one class file and paste below code in it this class file serves as class representation of parsed xml data.

2.XMLClasses.cs

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Xml.Serialization;//add referance
using System.Collections.ObjectModel;////add referance
using System.ComponentModel;////add referance
namespace LocalXmlParsing
{
    [XmlRoot("root")]
    public class States
    {
        [XmlArray("states")]
        [XmlArrayItem("state")]
        public ObservableCollection<State> Collection { get; set; }
    }
    public class State
    {
        [XmlElement("id")]
        public string stateId { get; set; }

        [XmlElement("name")]
        public string stateName { get; set; }

        [XmlArray("cities")]
        [XmlArrayItem("city")]
        public ObservableCollection<City> cityCollection { get; set; }
    }
    public class City
    {
        [XmlElement("cityid")]
        public string cityId { get; set; }

        [XmlElement("cityname")]
        public string cityName { get; set; }

        [XmlElement("pincode")]
        public string pinCode { get; set; }
    }
}
```

Now create one class XMLParser which contains XML parsing logic. I've used Singleton pattern for this class. The advantage of this approach is that our XML file gets parsed only once and parsed data can be used from any point in the application without any need to parse XML file again and again and hence it reduces the file IO.

XMLParser.cs

(C) Copyright Nokia 2014. All rights reserved.

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

using System.Collections.ObjectModel;
using System.Xml.Serialization; //add referance
using System.Xml.Linq;//add referance

namespace LocalXmlParsing
{
    public class XMLParser
    {
        private string _dataToParse;
        private static XMLParser _instance;

        private ObservableCollection<State> _stateCollection;

        private XMLParser()
        {

        }

        public static XMLParser Instance
        {
            get
            {
                if (_instance == null)
                {
                    _instance = new XMLParser();
                }
                return _instance;
            }
        }


        public string DataToParse
        {
            get
            {
                return _dataToParse;
            }
            set
            {
                this._dataToParse = value;
            }
        }

        public ObservableCollection<State> StateCollection
        {
            get
```


(C) Copyright Nokia 2014. All rights reserved.

```
            {
                return _stateCollection;
            }
            set
            {
                _stateCollection = value;
            }
        }


        public ObservableCollection<State> ParseStateData()
        {
            try
            {
                XmlSerializer serlizer = new XmlSerializer(typeof(States));
                XDocument document = XDocument.Parse(this._dataToParse);

                States rootXML = (States)serlizer.Deserialize(document.CreateReader());

                this._stateCollection = rootXML.Collection;

                return rootXML.Collection;
            }
            catch (Exception ex)
            {
                MessageBox.Show("ParseStateData()::" + ex.Message);
            }

            return null;
        }
    }
}
```

Now you can execute above xmlparser from any point in your code as per your requirement, e.g. on button click event or any other event. I added code in button click event. Following is function for that:

```
private void btnparse_Click(object sender, RoutedEventArgs e)
{
        this._parser = XMLParser.Instance;
        StreamResourceInfo strm = Application.GetResourceStream(new
Uri("/LocalXmlParsing;component/States.xml",UriKind.Relative)); //needs to be done only
once
        StreamReader reader = new StreamReader(strm.Stream);
        string data = reader.ReadToEnd();
        _parser.DataToParse = data;
        _parser.ParseStateData();


        lstStates.ItemsSource = _parser.StateCollection;
}
```

The first six lines of function block is required to execute only once it actually parses xml data, and you can get parsed data from any point in application by just calling StateCollection property.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.