Reporting memory usage on Windows Phone

This article shows how to interactively report the amount of memory your app uses as it runs on the emulator or device.

Introduction

Sometimes apps crash because they use too much memory, throwing an `OutOfMemoryException`. Most of the app memory is consumed by data, so you should plan for this in advance. Code reviews and re-factoring are very useful for reclaiming memory.

This article shows you how to monitor memory usage of your app. It shows how you can report the data to display it in a view made in XAML. There are also a few lines of code showing how to reduce current memory usage.

You can find different tips and best practices in Best practice tips for delivering apps to Windows Phone with 256 MB (and other articles in Portal:Windows Phone Performance and Optimisation and on MSDN - e.g. Developing apps for lower-memory phones).

Call from Debug Window

Memory data is written to the **Debug** window every 3 seconds using the `Timer` class. The app uses calls to `Microsoft.Phone.Info.Devices` to report peak memory. Depending on the value returned, your app will determine the upper limit for peak memory.

C# Code

```
public static PhoneApplicationFrame RootFrame { get; private set; }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.

            Utilities.BeginRecording();
```

C#

```
public class Utilities
    {

        private static Timer timer = null;

        public static void BeginRecording()
        {

            // start a timer to report memory conditions every 3 seconds
            //
            timer = new Timer(state =>
            {
                string c = "unassigned";
                try
                {
                    //
                }
                catch (ArgumentOutOfRangeException ar)
                {
                    var c1 = ar.Message;

                }
                catch
                {
                    c = "unassigned";
                }


                string report = "";
                report += Environment.NewLine +
                   "Current: " + (DeviceStatus.ApplicationCurrentMemoryUsage /
1000000).ToString() + "MB\n" +
                   "Peak: " + (DeviceStatus.ApplicationPeakMemoryUsage /
1000000).ToString() + "MB\n" +
                   "Memory Limit: " + (DeviceStatus.ApplicationMemoryUsageLimit /
1000000).ToString() + "MB\n\n" +
                   "Device Total Memory: " + (DeviceStatus.DeviceTotalMemory /
1000000).ToString() + "MB\n" +
                   "Working Limit: " +
Convert.ToInt32((Convert.ToDouble(DeviceExtendedProperties.GetValue("ApplicationWorkingSetLimit"))
/ 1000000)).ToString() + "MB";

                Deployment.Current.Dispatcher.BeginInvoke(delegate
                {
                    Debug.WriteLine(report);
                });

            },
                null,
                TimeSpan.FromSeconds(3),
                TimeSpan.FromSeconds(3));
        }
    }
```

Monitoring and Releasing Memory

In the example an `ObservableCollection` of XML Documents is populated to increase the memory footprint until it crashes. To reduce the footprint the app provides a button to `null` the object and reclaim memory using the Garbage Collector. The use of `GC.Collect()` is not ideal but sometimes essential when memory is tight. To exceed memory usage just keep clicking **Fill Memory** and you will crash the app! We are glad Windows Phone 8 has increased the memory available but it helps if you can manage client expectations of downloading large payloads on demand, especially over weak connections.

MainPage.xaml.cs (C#)

```
public partial class MainPage : PhoneApplicationPage
    {

        static long soc;

        ObservableCollection<XDocument> oc;
        public MainPage()
        {
            InitializeComponent();
            oc = new ObservableCollection<XDocument>();
            soc = 0;

        }

        private void FillMemory_Click(object sender, RoutedEventArgs e)
        {

            for (int i = 1; i < 500; i++)
            {
                this.report.Text = "";
                var xd = XDocument.Load("ServiceAlterations.XML");
                oc.Add(xd);

                soc += xd.ToString().Length;

                this.report.Text = Environment.NewLine +
                   "Current: " + (DeviceStatus.ApplicationCurrentMemoryUsage /
1000000).ToString() + "MB\n" +
                   "Peak: " + (DeviceStatus.ApplicationPeakMemoryUsage /
1000000).ToString() + "MB\n" +
                   "Memory Limit: " + (DeviceStatus.ApplicationMemoryUsageLimit /
1000000).ToString() + "MB\n\n" +
                   "Device Total Memory: " + (DeviceStatus.DeviceTotalMemory /
1000000).ToString() + "MB " + "\n" +
                   "Working Limit: " +
Convert.ToInt32((Convert.ToDouble(DeviceExtendedProperties.GetValue("ApplicationWorkingSetLimit"))
/ 1000000)).ToString() + "MB";
            }
        }

        private void ClearMemory_Click(object sender, RoutedEventArgs e)
        {
            oc = null;
            oc = new ObservableCollection<XDocument>();
            GC.Collect();
            soc = 0;
        }
```

Example Working Video

The media player is loading...

Source Code

You can download sample project from File:Memory Using Project.zip.