I believe that I have found an important memory leak in your library. It came to my attention that the memory consumption of my asp.net application is exaggeratedly larger than what I would expect, so I have been debugging my application with several different Memory Profilers in order to find the problem.

I was able to realize that the problem is that ALL the aspx pages are kept in memory even after the garbage collector was executed and the pages were disposed. As I mentioned before I used different Memory Profilers (ANTS, CLR Profiler, MemProfiler) to analyze the Problem and came to the same findings across the different Profilers. The images below are screenshots of the results of the MemProfiler Tool.

Reproducing the Problem

The first step that I took was to reproduce the problem in an entirely new dummy project. The Project consist of two aspx sites with identic functionality, the only difference between the two sites is that one has a C1Window and the other does not. The page with the C1Window is called yesleak.aspx and the one without the C1Window is called noleak.aspx (I have included the screenshots of the code as images). What the pages do is pretty straight forward: it creates an HtmlTable with 50.000 Rows and 500.000 Cells and adds the Table to a Placeholder on the page.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="YesLeak.aspx.cs" Inherits="ReproduceC1MemoryLeak.YesLeak" %>
<%@ Register Assembly="C1.Web.UI.Controls.3" Namespace="C1.Web.UI.Controls.C1Window" TagPrefix="cc1" %>
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <ajaxToolkit:ToolkitScriptManager CombineScripts="false" ID="scripts" runat="server" ScriptMode="Release" LoadScriptsBeforeUI="true"></ajaxToolkit:ToolkitScriptManager>
        <cc1:C1Window ID="filterWnd" runat="server" Width="450px" height="300px">

    </cc1:C1Window>
    <div>
    <asp:PlaceHolder ID="phMenuList" runat="server"></asp:PlaceHolder>
    </div>
    </form>
</body>
</html>
```

Abbildung 1 yesleak.aspx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;

namespace ReproduceC1MemoryLeak
{
    public partial class YesLeak : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            HtmlTable table = new HtmlTable();
            for (int i = 0; i < 50000; i++)
            {
                HtmlTableRow tr = new HtmlTableRow();
                for (int x = 0; x < 10; x++)
                {
                    HtmlTableCell tc = new HtmlTableCell
                    {
                        InnerText = String.Format("Cell {0}/{1}", i, x)
                    };
                    tr.Cells.Add(tc);
                }
                table.Rows.Add(tr);
            }
            phMenuList.Controls.Add(table);
        }
    }
}
```

Abbildung 2 yesleak.aspx.cs

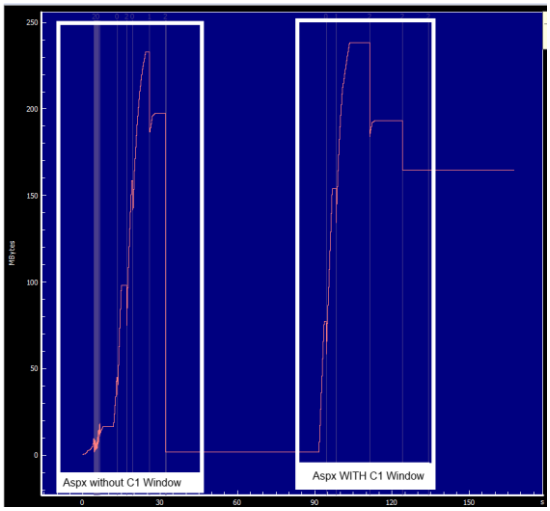
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="NoLeak.aspx.cs" Inherits="ReproduceC1MemoryLeak.NoLeak" %>
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <ajaxToolkit:ToolkitScriptManager CombineScripts="false" ID="scripts" runat="server" ScriptMode="Release" LoadScriptsBeforeUI="true"></ajaxToolkit:ToolkitScriptManager>
    <div>
    <asp:PlaceHolder ID="phMenuList" runat="server"></asp:PlaceHolder>
    </div>
    </form>
</body>
</html>
```

Abbildung 3 noleak.aspx

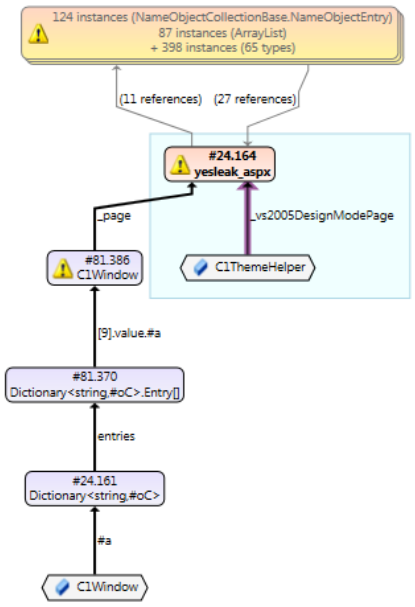
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;

namespace ReproduceC1MemoryLeak
{
    public partial class NoLeak : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            HtmlTable table = new HtmlTable();
            for (int i = 0; i < 50000; i++)
            {
                HtmlTableRow tr = new HtmlTableRow();
                for (int x = 0; x < 10; x++)
                {
                    HtmlTableCell tc = new HtmlTableCell
                    {
                        InnerText = String.Format("Cell {0}/{1}", i, x)
                    };
                    tr.Cells.Add(tc);
                }
                table.Rows.Add(tr);
            }
            phMenuList.Controls.Add(table);
        }
    }
}
```

Abbildung 4 noleak.aspx

So far, so good. Everything works as expected, but while looking at the behavior of the two pages in a memory profiler I see a very important difference between the two. On the Image below I recorded the memory usage after each call to the aspx pages, after the page was finished loading I took a snapshot of the memory, which forces the Garbage Collector to be executed. On the Image below the area in the first white square is the call the noleak.aspx page, the memory behavior is exactly as I would expect it. It allocates a bunch of memory for the controls and releases all of it once the page was unloaded and the GC executed. On the area inside the second white square is the memory usage of the yesleak.aspx. As you can see the behavior with this page is not what one would expect since over 150MB of data is kept on memory.



A closer look at the instances inside the memory confirmed my suspicion that the entire aspx page with all of the controls etc. etc. is being kept in memory. The screenshot below shows that an instance of the C1ThemeHelper and a Dictionary in C1Window keep the entire page from being Garbage collected:



This issue is very serious for my application since we use very large tables and this behavior is eating up all of the memory. I have (as far as I know) the latest version of the C1WebUI library http://prerelease.componentone.com/dotnet20/c1webui/2013-T1/. Is there anything I can do to influence this caching behavior?