Profiler.cs

|  |  |
| --- | --- |
| **Line** | **Code** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | using System;  using System.Diagnostics;  using System.Threading;  namespace Benchmarking {    /// <summary>    /// Make sure you compile in Release with optimizations enabled, and Run the    /// tests outside of Visual Studio (This part is important because the JIT  /// stints its optimizations with a debugger attached, even in Release mode).    /// </summary>    static public class Profiler {      static public double Profile(string desc, uint iterations, Action func) {        //Run at highest priority to minimize fluctuations caused by others        Process.GetCurrentProcess().PriorityClass = ProcessPriorityClass.High;        Thread.CurrentThread.Priority = ThreadPriority.Highest;        // Warm up        func();        // Clean up        GC.Collect();        GC.WaitForPendingFinalizers();        GC.Collect();//To make sure the "finalized" objects are also collected.        var watch = Stopwatch.StartNew();        for (uint i = 0; i < iterations; i++) {          func();        }        watch.Stop();        double elapsedTime = watch.Elapsed.TotalMilliseconds;        if (desc != null) Console.WriteLine("{0,-40}\t{1,15:n} ms", desc,  elapsedTime);        return elapsedTime;      }  static public double Profile(string desc, uint iterations, Action func,  out int gcCount) {        Process.GetCurrentProcess().PriorityClass = ProcessPriorityClass.High;        Thread.CurrentThread.Priority = ThreadPriority.Highest;        func();        GC.Collect();        GC.WaitForPendingFinalizers();        GC.Collect();        gcCount = GC.CollectionCount(0);        var watch = Stopwatch.StartNew();        for (uint i = 0; i < iterations; i++) {          func();        }        watch.Stop();        gcCount = GC.CollectionCount(0) - gcCount;        double elapsedTime = watch.Elapsed.TotalMilliseconds;        if (desc != null) Console.WriteLine("{0,-40}\t{1,15:n} ms",  desc, elapsedTime);        return elapsedTime;      }    }  } |