

Automating performance profiling with ANTS Performance Profiler 6

Routinely profiling your applications in ANTS Performance Profiler 6 helps to ensure that your programs are using resources efficiently and will perform well even when scaled.

To help you integrate performance profiling into your usual testing or build processes, ANTS Performance Profiler 6 now allows you to run profiling sessions at the command prompt. The profiler results can be output to CSV, XML or HTML, which means that you can easily check the results for abnormal values as part of your automated routines.

This example uses the following simple C# Console Application (called *SimpleApp.exe*) which prints '.' to the console 100 times:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace SimpleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("The application has started");
            // Count from 0-99
            int i = 0;
            while (i < 100)
            {
                Console.Write('.');
                i++;
            }
            Console.WriteLine("The application is exiting");
        }
    }
}
```

Another simple C# Console Application can read the results CSV file created by ANTS Performance Profiler, to check that Write() is called exactly 100 times, thereby verifying that SimpleApp is performing correctly:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace ReadOutput
{
    class TextFileReader
    {
        static void Main(string[] args)
        {
            StreamReader sr = new StreamReader(new
FileStream("C:\\testing\\results.csv", FileMode.Open, FileAccess.Read));
            string line;
            int ok = 1;
            // Read file line-by-line
            while ((line = sr.ReadLine()) != null)
            {
                char separator = ',';
                string[] linedata = new string[10];
                linedata = line.Split(separator);
                // Try to cast linedata[3] as int, not string
                int i;
                try
                {
                    i = (int)int.Parse(linedata[3]);
                    // Check the number of times that Write(char value) is run
                    if ((linedata[2] == "Write(char value)") && (i != 100))
                    {
                        Console.WriteLine("Test failed");
                        ok = 0;
                    }
                }
                catch
                {
                    // Do nothing
                }
            }
        }
    }
}

```

```
        }
    sr.Close();
    if (ok == 1)
    {
        Console.WriteLine("Test passed OK");
    }
}
}
```

Finally, an MS-DOS batch file can be written to profile SimpleApp in ANTS Performance Profiler and, when this is complete, to check that the test passed:

```
C:
CD /
CD "Program Files\Red Gate\ANTS Performance Profiler 6\"
Profile.exe /e:"C:\testing\SimpleApp.exe" /ll /csv:"C:\testing\results.csv"
C:
CD testing
ReadOutput.exe
```

The console shows:

```
C:\testing>ReadOutput.exe
Test passed OK
```

giving the expected confirmation that SimpleApp is performing correctly.