

## Re: Do I produce a handle-leak?

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2005-06/msg01882.html>

---

- *From:* "Manfred Braun" <aa@xxxxx>
  - *Date:* Fri, 10 Jun 2005 17:50:09 +0200
- 

Hello All,

after Nicholas came up with his question, I started Perfmon. Now, I see the Taskmanager is not a good compaign for a deeper analyzis :-)

In perfmon, I am seeing a saw tooth, which means, currently, the number of handles is changing from a footprint of about 361 up to a max of 461. I thing the GC collects handles over time. But my start was about 172.

How to interpret this results from perfmon? After I saw the saw tooth, I think, this is related to CLR and there is really no problem in my code – that's what I tried to understand, because I think, I really understand my code ;-)

I'll let Perform running for a time now.

Thanks,  
Manfred

"Manfred Braun" <aa@xxxxx> wrote in message  
[news:OowfeHdbFHA.3144@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:OowfeHdbFHA.3144@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

> Hi,

>

> this proggi is very simple, was just created to see some small/large cpu  
> load over time. Here it is in full:

>

> Thanks so far and

> best regards,

> Manfred

> ===

> /\*

>

> Name: TP.cs

> Author: mb

> Created: 15.0.2005

> Purpose: Sleep a time, use cpu, sleep

> Compile csc /nologo /debug:full /t:exe /out:TP.exe TP.cs

>

Re: Do I produce a handle-leak?

```
> */
>
> using System;
> using System.Diagnostics;
> using System.Threading;
> using System.Timers;
>
>
>
> namespace Test
> {
>
>
> public class Test
> {
>
> private System.Timers.Timer timer;
> int counter;
>
>
> public static int Main(string[] args)
> {
> int rtc = 0;
>
> if(args.Length != 0)
> {
> Test t = new Test();
> t.Run(args);
>
> Console.WriteLine("Press <enter> to exit.");
> Console.ReadLine();
> }
> else
> {
> Console.WriteLine("Args !!! [P1=Interval{s}]");
> rtc = 1;
> }
>
> return rtc;
> }
>
>
> private void Run(string[] args)
> {
> //Create the "Timer"
>
> this.counter = 0;
> this.timer = new System.Timers.Timer(Int32.Parse(args[0]) * 1000);
> this.timer.Elapsed += new ElapsedEventHandler(this.OnTimer);
> this.timer.Enabled = true;
> }
```

Re: Do I produce a handle-leak?

Re: Do I produce a handle-leak?

```
>
>
> private void OnTimer(object sender, ElapsedEventArgs e)
> {
> Console.WriteLine("TP.OnTimer;Counter:{0}", this.counter);
> int steps = 0;
>
> this.counter++;
> this.counter = this.counter % 10;
>
> if(this.counter == 4)
> {
> steps = 1000;
> }
> else
> {
> if(this.counter == 9)
> {
> steps = 10000000;
> }
> }
>
> if(steps != 0)
> {
> Job j = new Job(steps);
> ThreadStart ts = new ThreadStart(j.Run);
> Thread t = new Thread(ts);
> t.IsBackground = true;
> t.Start();
> }
> }
>
>
> private class Job
> {
> private int steps;
>
> public Job(int steps)
> {
> this.steps = steps;
> }
>
> public void Run()
> {
> Console.WriteLine("Job.Run;Steps:{0}", this.steps);
>
> //Consume some cpu
>
> for(int i = 0; i < this.steps;i++);
> }
>
```

Re: Do I produce a handle-leak?

Re: Do I produce a handle-leak?

```
> }//class
>
> }//class
>
>
>
> }//namespace
>
> ===
>
> "Willy Denoyette [MVP]" <willy.denoyette@xxxxxxxxxx> wrote in message
> news:ecLYdEdbFHA.2756@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>> Could you post the remaining of this program?
>>
>> Willy.
>>
>> "Manfred Braun" <aa@xxxxx> wrote in message
>> news:upW\$T1cbFHA.2128@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>> Hi All,
>>>
>>> I am writing a proggi, which should monitor some processes. While
doing
>>> this, I needed a test-program and wrote one, which does nothing else
> than
>>> to
>>> consume some cpu, sometimes more, sometimes less. As I monitored this
>>> process for a time, I found the handles increasing without end. I
assume
>>> this is a bug, but I do not understand, where this could be located in
> my
>>> simple app. The core is as follows:
>>>
>>> private void OnTimer(object sender, ElapsedEventArgs e) //fires every
10
>>> seconds
>>> {
>>> int steps = 0;
>>> this.counter++;
>>> this.counter = this.counter % 10;
>>>
>>> if(this.counter == 4)
>>> {
>>> steps = 1000;
>>> }
>>> else
>>> {
>>> if(this.counter == 9)
>>> {
>>> steps = 10000000;
>>> }
>>> }
```

Re: Do I produce a handle-leak?

Re: Do I produce a handle-leak?

```
>>>
>>> if(steps != 0)
>>> {
>>> Job j = new Job(steps);
>>> ThreadStart ts = new ThreadStart(j.Run);
>>> Thread t = new Thread(ts);
>>> t.IsBackground = true;
>>> t.Start();
>>> }
>>> }
>>>
>>> private class Job
>>> {
>>> private int steps;
>>>
>>> public Job(int steps)
>>> { this.steps = steps; }
>>>
>>> public void Run()
>>> {
>>> Console.WriteLine("Job.Run;Steps:{0}", this.steps);
>>>
>>> //Consume some cpu
>>>
>>> for(int i = 0; i < this.steps;i++);
>>> }
>>> }//class
>>>
>>> The number of threads is stable in this program [as I expect], the
> handles
>>> are increasing endless. Where is my fault? And what are these handles?
>>> Some help would really be very welcomed!!
>>>
>>> Best regards,
>>> Manfred Braun
>>>
>>> (Private)
>>> Mannheim
>>> Germany
>>>
>>> mailto: manfred.braun\_@xxxxxxxxxxxxx
>>> (Remove the anti-spam-underscore to mail me!)
>>>
>>>
>>
>>
>
>
```

Re: Do I produce a handle-leak?

- *Follow-Ups:*
  - ◆ **Re: Do I produce a handle-leak?**
    - ◇ *From:* Willy Denoyette [MVP]
- *References:*
  - ◆ **Do I produce a handle-leak?**
    - ◇ *From:* Manfred Braun
  - ◆ **Re: Do I produce a handle-leak?**
    - ◇ *From:* Willy Denoyette [MVP]
  - ◆ **Re: Do I produce a handle-leak?**
    - ◇ *From:* Manfred Braun
- Prev by Date: **Re: Do I produce a handle-leak?**
- Next by Date: **Re: .Net (C#) Service Fails To Start**
- Previous by thread: **Re: Do I produce a handle-leak?**
- Next by thread: **Re: Do I produce a handle-leak?**
- Index(es):
  - ◆ **Date**
  - ◆ **Thread**