

Re: memory leak

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.compactframework/2006-08/msg007>

- *From:* "<ctacke/>" <ctacke_AT_OpenNETCF_com>
 - *Date:* Mon, 14 Aug 2006 08:18:28 -0400
-

Sounds like an OOM on a bitmap allocation isn't triggering a GC like it should. If that's the case, it's certainly a bug.

-Chris

"Hilton" <nospam@xxxxxxxxxx> wrote in message
[news:gOTDg.688\\$q63.333@xx](news:gOTDg.688$q63.333@xx)

Chris,

I did some more testing. Using CF 2.0 makes no difference – still throws exception. Adding GC.Collect() causes the memory to drop, app keeps running both on desktop and PPC. So, the OOM is not causing a GC.Collect() because that would cause the OOM condition to go away, the "new" to then succeed etc. Looks like they fixed this on the desktop in a later version of the .NET Framework according to Doug's post. I'm still running 1.1.4322.2032.

Chris, do you agree? (given that my findings stated above are correct)

Hilton

"<ctacke/>" <ctacke_AT_OpenNETCF_com> wrote in message
[news:edV\\$yhzvGHA.4688@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](news:edV$yhzvGHA.4688@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

I don't have Studio on this machine. What happens if you put a GC.Collect in the exception handler, then allow it to continue?

-Chris

"Hilton" <nospam@xxxxxxxxxx> wrote in message
[news:f9PDg.9744\\$FN2.6734@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](news:f9PDg.9744$FN2.6734@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Re: memory leak

Chris,

I agree there is a misunderstanding of the fundamentals. Run the short program below and see for yourself. On the Pocket PC, the device runs out of memory and throws an `OutOfMemoryException`; on the desktop, it used all 2GB that was free when it started and threw a `"System.ArgumentException: Invalid parameter used"` on the `"new Bitmap"` line.

Now do you still agree that the GC will cleanup `Bitmap` objects? Sorry
Chris, it doesn't happen as you say cause the `"new Bitmap"` code below that `*does not keep a reference to each bitmap*` ran out of memory.

Hilton

Create a form, add a timer with the code:

```
this.timer = new Timer ();  
this.timer.Interval = 250;  
this.timer.Tick += new EventHandler(timer_Tick);  
this.timer.Enabled = true;
```

then add this method:

```
private void timer_Tick(object sender, EventArgs ea)  
{  
    bool error = false;  
  
    this.timer.Enabled = false;  
  
    try  
    {  
        this.bmp = new Bitmap (1024, 1024);  
  
        this.Text = DateTime.Now.ToLongTimeString();  
    }  
    catch (Exception e)  
    {  
        error = true;  
  
        using (System.IO.StreamWriter sw = new  
            System.IO.StreamWriter ("exp.txt"))  
        {  
            sw.WriteLine (e.ToString());  
        }  
    }  
}
```

Re: memory leak

```
this.Text = "Exception";  
}  
this.timer.Enabled = !error;  
}
```

"<ctacke/>" <ctacke[@]opennetcf[dot]com> wrote in
message
news:esCAJeyvGHA.1772@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

I think there's still a misunderstanding of the
fundamentals.

Dispose is not a required call – it is optional.
Your application
knows better than anything else when you
are done with resources, and
at that point, you can call Dispose to let the
GC know that you are
done with those resources. These classes are
written so that during
that call, managed resources are released.

If you don't call Dispose, that's fine. When
the GC goes to collect,
it will walk the roots. It will see that the
Bitmap has no references,
and it will then run Dispose on it for you and
move it to the Finalizer
queue. On the `_next_` GC cycle, the actual
Bitmap will be released.
This all happens automatically, without your
intervention. It just
happens late – when the GC is collecting –
likely because you're low on
resources.

Calling Dispose is simply a good idea
because it allows the release of
system resources when the app is done with
them rather than waiting for
the system to GC.

For more info, take a look at my presentation
from MEDC:

<http://blog.opennetcf.org/ctacke/PermaLink.guide806d34b-a8d8-45e8-9de8-bec588>

--
Chris Tacke

Re: memory leak

OpenNETCF Consulting
www.opennetcf.com

--

"Hilton" <nospam@xxxxxxxxxxx> wrote in message
[news:0IMDg.8207\\$9T3.540@xx](mailto:news:0IMDg.8207$9T3.540@xx)

Chris,

This has come up before, so no use beating a dead horse (I'm the guilty one here since I first mentioned it), but here are some points which hopefully helps to explain my comments:

1. If ALL objects were required to be disposed, would that be a design bug? Of course it would, it is .NET with a garbage collector.
2. Then why do we have to dispose Bitmap? Because MS took a short-cut. Note that MS have already said that they will *fix* this problem.
3. Another bad choice IMHO was to put Bitmap in System.Drawing – that is the wrong place since you don't only draw with Bitmap and a Bitmap image should not be directly associated with UI. It should be in System.Image or something similar. Here is an example in pseudo-C#:
Imagine you wanted to convert all BMP files to PNG files using a command line EXE. It

Re: memory leak

should simply be "using System.Image; foreach BMPFile in directory { new Bitmap (BMPFile).SaveAs (PNG); }" but instead we have to include System.Drawing (UI stuff), plus we need to do our own memory management (the code above would have a huge memory leak).

4. FWIW: I think Dispose should be an optional call, not a mandatory call required to prevent a memory leak.

5. There are other things like why they called it SortedList instead of SortedMap etc... (unrelated to Bitmap)

Bottom line, I love C#, work with it day in and day out, absolutely darn amazing on Pocket PCs etc, so don't think for a second I'm bashing the C# language, the .NET team, or Microsoft. Just sometimes, bad design decisions get made (*IHMO*).

Hilton

"<ctacke/>"

<ctacke[@]opennetcf[dot]com>

wrote in message

news:eWyovzwvGHA.3372@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

The same
has to be
done on the
full
framework,
and again,
it's not a

Re: memory leak

bug.

-Chris

"Hilton"

<nospam@xxxxxxxxxx>

wrote in

message

[news:gvJDg.12218\\$gY6.3863@xx](mailto:news:gvJDg.12218$gY6.3863@xx)

I
bet
you're
doing
something
like
"xyz.Image
=
new
Bitmap
(...)"
-
right?
If
so,
there
is
a
bug
in
the
CF
design/implementation
(IMHO)
that
forces
you
to
do
your
own
memory
management;
i.e.
for
every
"new
Bitmap
()"

you

Re: memory leak

do,
you'll
need
to
do
a
"Dispose()"
on
that
bitmap
when
you're
done
with
it
(aka
malloc
and
free).

Let
us
know
if
that
helps,

Hilton

"raju"
<ponnurajs@xxxxxxxx>
wrote
in
message
news:1155390117.643466.283160@xx

Hai

In
my
device
application
(windows
ce
application
using
vb.net),
i
am
having

Re: memory leak

nearly
25
forms.
Moving
from
one
form
to
another
form,
just
i
am
hide
the
first
form
and
show
the
second
form.

Each
form
I
am
having
some
picturebox,
putting
images
for
that
picturebox
using
imagelist.

My
application
works
very
well.
But,
if
the
application
running
contineously
for
1

Re: memory leak

Re: memory leak

or
2
hours,
i
am
getting
error
such
as
"low
memory".

when
i
checked
the
memory,
it
will
automatically
increasing
when
the
application
is
in
running.
First
initial
time,
it
occupies
22
mb
and
it
increasing
upto
32
to
35
mb.

What
is
the
reason
for
this?
and

Re: memory leak

Re: memory leak

how
to
correct
this
memory
problem.

Regards
Raju.