

# Re: Do-Nothing WinForm App Using 4 Threads?

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-01/msg02117.html>

---

- *From:* "Ignacio Machin \(.NET/ C# MVP \)" <machin TA laceupsolutions.com>
  - *Date:* Mon, 15 Jan 2007 13:50:36 -0500
- 

Hi,

The .NET threads are not physically related to the one from win. For example there is nothing like ThreadPool in native win.

At least one of those thread that you are seeing must be running the GC.

Take a look at this link ; it refers to the CF but I bet something similar should apply to the full framework

## Threads

There are up to four threads created by a .NET Compact Framework application:

- a.. A main application thread.
- b.. A thread used to control various period timers and time-outs that can be scheduled by the system or applications.
- c.. A thread used to track changes to the active TCP/IP interfaces (simulating the media sense behavior that is present on Windows XP but not Windows CE).
- d.. A thread that is used to run object finalizers. It is created when the first finalizable object is garbage collected.

For more information about threading support, see Threading in the .NET Compact Framework.

"gsimmons" <simmons.garry@xxxxxxxx> wrote in message  
<news:1168870674.106157.29190@xx>

| I've been researching multi-threaded WinForms apps and thread  
| synchronization stuff for a couple days since I'm working on  
| refactoring a multi-threaded GUI app at work and want to be sure it's  
| rock solid/thread-safe. I've seen all the posts about using BeginInvoke  
| to have worker threads interact with the UI. My question is this:

## Re: Do-Nothing WinForm App Using 4 Threads?

| I created a plain old Windows Form application (VS.NET 2005) with a  
| blank form, built it (release build), ran it from Windows Explorer and  
| Task Manager claims the application is using four threads. What are  
| they? Obviously one is the main app/UI. What are the others?

| I then added a Windows Timer control to the form and updated a label  
| every second. Still four threads. I expected a timer to be on its own  
| thread even though it posts "tick" events to the UI message  
| loop/thread.

| I removed the Windows Timer and replaced it with a System Timer which  
| runs "tick" events on a different thread, yet the thread count still  
| stays at four. Huh?

| Is there a built-in thread pool with a few workers hanging around to do  
| timers and stuff? What exactly is going on here?  
| Are those extra threads part of the CLR doing its magic to run the EXE?  
| Other?

| My next question will be how many threads remoting (inter-process on  
| the same PC) adds to the mix. The UI app I'm refactoring makes use of  
| several remote objects (running methods and receiving events with  
| data). I'm guessing there is a thread for each proxy/socket to handle  
| marshalling stuff. And the event handling from remote objects  
| supposedly grabs a thread from the thread pool. Inquiring minds what to  
| really KNOW what's going on under the hood.

| Pointers to good articles and/or books welcome. Spend the last weekend  
| Googling and reading...

| Thanks!