DotNet Startup Remote Debugging Hook

When performing remote debugging of an application or service, it may be necessary to see what is happening during early startup.

> Adapted from this article: I Wish I Knew About Debugger.Launch Earlier

Normally. By the time you have started the remote process and are in a position to attach a remote debugger, the target process is long past any startup logic that you might need to see, and has finished or crashed.

Here's a way to force a starting process to pause and wait for your remote debugging session to attach.

> There is no negative benefit to including a startup debug hook like below in any application or service that you develop.
> Having one, ensure that you can remotely troubleshoot early startup activity that is preventing operation.
> This is especially helpful when seeing startup failures, after deploying a new service, or to a new environment.
> Especially, when early startup activities may not show up in logs (or may occur before logging starts).

This technique involves adding a block of code to the application/service, and using a command line switch to tell it to pause for you.

Implementation

To implement this technique, add something like the below block, to the top of main, in your Program.cs:

```
// Look to see if we are to wait for the debugger...
// NOTE: To leverage this wait, add this command line argument:
//    -waitfordebugger=yes
{
```

```
    bool waitfordebugguer = false;
    // Get the command line arguments, so we can quickly parse them for a debugger wait signal...
    string[] arguments = Environment.GetCommandLineArgs();
    foreach(var f in arguments)
    {
        if(f.ToLower().Contains("waitfordebugger=yes"))
        {
            waitfordebugguer = true;
            break;
        }
    }

    // We are to wait for the debugger.
    if(waitfordebugguer)
    {
        //Spin our wheels waiting for a debugger to be attached....
        while (!System.Diagnostics.Debugger.IsAttached)
        {
            System.Threading.Thread.Sleep(100); //Or Task.Delay()
        }

        // We will stop here, once the debugger is attached, so no explicit breakpoint is required.
        System.Diagnostics.Debugger.Break();

        Console.WriteLine("Debugger is attached!");
    }
}
```

The above block makes use of a couple concepts. We'll explain each, below:

First, it checks for a command line argument to wait for a debugger to attach: 'waitfordebugger=yes'.
If not present, the application will startup as normal.

Next. If the waitfordebugger argument is set, the starting process will spin-wait in the WHILE loop until your remote debugger has attached.

> BEWARE: The above logic will wait FOREVER, in the WHILE loop, until a debugger is attached, if waitfordebugger=yes was set on the command line.
> So. Be sure to remove the command line argument, after troubleshooting session.
> Or, the process may appear hung on its next start.

Once a debugger attaches to the process, execution will drop past the WHILE loop, and pause at the Debugger.Break() statement.
This is a special statement the compiler allows in code, that forces a breakpoint stop if a debugger is attached.

> You don't have to explicitly set a breakpoint on the Debugger.Break() statement.
> The runtime will pause, there, as if a breakpoint was already set.

Usage

To leverage the above debug hook, add this command line argument to the startup command for your application/service:

```
waitfordebugger=yes
```

And, attach your remote debugger to the process.

Once attached, your remote debugging session will see the application paused at the Debugger.Break() statement.

You can then step execution past the Debugger.Break() in the same manner as if the application/service was started from your IDE.

This allows you to troubleshoot startup problems that may occur in test or prod, as long as you can attach a remote debugger.

See this page for how to accomplish that: Visual Studio Linux Remote Debugging

Cheers

Revision #5
Created 20 January 2025 20:47:40 by glwhite
Updated 20 January 2025 21:21:09 by glwhite