Visual Studio 2005: Cool Debugging Tricks

By Peter Bromberg

Printer Friendly Version

View My Articles

What percentage of your development time would you say that you spend debugging code? Here are some useful debugging techniques for developer productivity.

Most developers I've talked to say they spend anywhere from 30% to 50% or more of their time -- not writing the code -- but debugging and fixing it. Not only do we use the debugger to check code we've written before we can go on to write some more, we also use it in testing a completed project under various scenarios. And when there is a complaint about deployed code, we often need to resort to running it in debug mode to attempt to isolate and identify the reported problem.

With Visual Studio 2005, the debugging environment has become vastly superior to anything previously available. Developers migrating from other platforms to .NET often revel in the greatly superior debugger support we have in .NET, compared to what they have to work with in other environments.

Common logic would dictate that the better you are at a particular skill, the less time you will spend doing it and therefore the more efficient you become at the overall process. However, its a known fact that the debugging skills of a majority of developers consist primarily of knowing how to set a breakpoint with the F9 key or clicking the left margin on a line to set a breakpoint. An even bigger advance we master is how to write "System.Diagnostics.Debug.WriteLine(e.Message +e.StackTrace)" inside a catch block, set a breakpoint on it, and read the exception information (doh!).

How many developers do you know that routinely use the locals, autos, call stack, watch, quickwatch, or the "this" window? Who know there are at least two documents in the documentation tree for Visual Studio that describe the debuggers? How many know you can change the value of a variable in any of the debug-related windows, or that you can set multi-statement breakpoints on a single line of code? Who knows how to debug a single thread in a multithreaded application?

In this short piece, I'll share a few of the (what I think are) cool debugger techniques I've learned, as well as point you to some resources on debugging, and if you have a particular technique or trick you'd like to share, you can post it on our article discussion board.

Tip: Call Stack Window Debugging: Many developers are not aware that you can set breakpoints in the Call Stack Window (except when doing SQL debugging). Just highlight the call you want to stop on and either press F9 or right-click on the line and select Insert Breakpoint from the shortcut menu. You can also right-click on any breakpoint in the Call Stack Window to enable, disable or set the properties of the breakpoint. For those unfamiliar with the Call Stack Window, this is how you find out "how did I get here" - it enables you to go back up the call stack to find out "from where" the method that you are in was called. Not only can you see where it was called from, you can actually go "Back there" and trace the exact execution path in the debugger. Not to be an evangelist, but if you haven't discovered it yet, please do so now and "get religion".

Tip: Use Run to Cursor for one-shot breakpoints: All you need to do is right-click on the line you want and choose "Run to Cursor" from the menu, and program execution will immediately continue to the point where you are at. Many developers are not aware that this is available from both debugging and editing. In edit mode, this will actually start the debugger and run your program to the cursor location. As with breakpoints, right-clicking in the Call Stack window pops up the shortcut menu that also offers this feature.

Tip: Set sub-expression breakpoints: For example, if you have the expression:

```
for (int j=0; m=0;j<ActiveLines.Count ;j++;m--)
```

when debugging, and you click in the margin next to the line, the breakpoint doesn't extend on the subexpressions that do the incrementing. By placing the cursor in the j++ or m-- portion of the line and hitting F9, you can set multiple breakpoints on the subexpressions and view these as separate entries in the Breakpoints window.

Tip: Use the BreakPoints Window: If you know the name of the class and method you want to break on, you can choose New from the Breakpoints window and type it directly into the Function area of the window. Developers can spend 15 minutes wandering through their projects opening files just so they can position the cursor over a line of code and hit the F9 key to set a breakpoint. If you know the name of the class and method, doing it this way will find it and set the breakpoint. In addition, you can click it in the Breakpoints list of the window and go to the location automatically.

You can also set a breakpoint by just entering the method name, if it is unique in your application. And, if it isn't you will actually get a list of ALL the locations where the method can be found, which is extremely useful for setting multiple breakpoints on a commonly used function throughout your project. It also displays for overloaded methods in the same class.

Tip: Use the Find ComboBox: If you enter a method name in the Find Box (that's on the top menu bar, just to the right of the Configuration dropdown that reads Debug, Release, Configuration Manager) and hit F9, this will also find the method and set a breakpoint.

Tip: Use Hit Counts: In the Breakpoints window, when you right-click and choose Properties and then click the Hit Counts button, you can modify a breakpoint with four possible combinations of choices. Experiment with this to see how useful it can be. What makes this especially useful is that when you have stopped in the Debugger, it tells you how many times your breakpoint has executed. You can also create conditional expressions that will trigger the debugger only if your expression evaluates to true or has changed since the last time it was evaluated.

Tip: Use the Watch window (and its brethren): The power offered by the Watch window and its close cousins, the Quick Watch Dialog, Autos window, Locals window, and the This / Me window are what can make the difference between hopping around aimlessly all day looking for a bug and quickly solving a problem. Also, you can use any of these windows to change a variable's value by simply typing in a new value.

Tip: Break only when a specific thread calls a method: To set a per-thread breakpoint, you need to uniquely identify a particular thread that you

have given a name with its Name property. You can set a conditional breakpoint for a thread by creating a conditional expression such as "ThreadToStopOn" == Thread.CurrentThread.Name .

You can manually change the name of a thread in the Watch window by watching variable "myThread" and entering a Name value for it in the value window. If you don't have a current thread variable to work with, you can use Thread.CurrentThread.Name to set the current thread's name. There is also a private integer variable in the Thread class, DONT_USE_InternalThread, this is unique to each thread. You can use the Threads window to get to the thread you want to stop on, and in the Watch window, enter Thread.CurrentThread.DONT_USE_InternalThread to see the value of it so you can create the right conditional breakpoint expression.

Tip: Use Assertion Debugging: The idea behind asserts is very simple: When you are writing code, you understand that some condition is always expected to be true. If that condition is ever not true, then there is a bug. Asserts can be written so that if this condition is ever not true, the debugger will be launched at that exact point in your code:

System.Diagnostics.Debug.Assert (myValue >=0)

The nice thing about Assert windows is that they not only give you the opportunity to break or not, they also show you the stack trace at that point. A very powerful, yet highly under-utilized feature.

That's it for now. You can find out about the above tips and much more by reading some of the following books (there are many others besides these):

Debugging Applications for Microsoft .NET and Microsoft Windows by John Robbins (MS Press)

Debugging Strategies for .NET Developers by Darin Dillon (aPress)

Comprehensive VB.NET Debugging by Mark Pearce (aPress)

Finally, consider looking over the following documents:

in the Documentation or on MSDN Online in the section "Building Debugging and Testing":

http://msdn2.microsoft.com/en-us/library/d8k88a0k(vs.80).aspx

Biography - Peter Bromberg

Peter Bromberg is a C# MVP, MCP, and .NET expert who has worked in banking, financial and telephony for over 20 years. Pete focuses exclusively on the .NET Platform, and currently develops SOA and other .NET applications for a Fortune 500 clientele. Peter enjoys producing digital photo collage with Maya,playing jazz flute, the beach, and fine wines. You can view Peter's UnBlog and IttyUrl sites. Pete Tweets at peterbromberg ***No Emails! Post it to the forums, I answer there!***

Article Discussion: Visual Studio 2005: Cool Debugging Tricks

Peter Bromberg posted at Thursday, July 26, 2007 1:06 PM

Original Article

reply

re: Article Discussion: Visual Studio 2005: Cool Debugging Tricks

Phil replied to Peter Bromberg at Wednesday, March 31, 2010 10:53 PM

What do you mean by the "This / Me window" in debugging tips?

reply

re: Article Discussion: Visual Studio 2005: Cool Debugging Tricks

Shreejeet replied to Phil at Wednesday, September 08, 2010 1:28 AM

Hi...
This/Me windows its means that The Current active From and Class.
eg. this.close();
I am beginer in this field.

reply

Related Items

Visual Studio, Debugging

debugging c# code in Visual Studio 2005 slow in . NET

. . . debugging c# code in Visual Studio 2005 slow in . NET Java code in IntelliJ IDEA<br . . . . . Is it possible to change any compilation flags to ma" debugging c# code in Visual Studio 2005 slow Lars Schouw posted . . .

How Do I Access MSDN Forums in Visual Studio General

. . . How Do I Access MSDN Forums in Visual Studio General have been trying to access the MSDN Forums using . . . . . access a MSDN Forum. I have a posting on the Visual Studio Debugging Forum, I get notifications of replies, and I can . . .

Debugging without running explorer. exe in WindowsCE Embedded

. . . Debugging without running explorer. exe in WindowsCE Embedded i am . . . . . on an armv4i platform i can not connect the visual studio" Debugging without running explorer. exe thinfis posted on Friday, November 14 . . .

Remote debugging from Vista to XP64 in Visual Studio . NET Debugging

. . . Remote debugging from Vista to XP64 in Visual Studio . NET Debugging I'm try to enable remote debugging from Vista to . . .

Visual STUDIO 2005 is slow in Visual Studio . NET Debugging

. . . Visual STUDIO 2005 is slow in Visual Studio . NET Debugging now when i open the solutuion whit Visual . . .

Debugging Visual Studio C# into Visual C++ 6. 0 in . NET

. . . Debugging Visual Studio C# into Visual C++ 6. 0 in . NET registered to it that was . . .

Adding Excel Solver using C# under IIS7 Microsoft Excel

. . . 29 AM My development environment is Windows Server 2008 and Visual Studio 2008. Basically I create a Excel worksheet with C# and . . . . . perfectly without error when I run it through the local debugging Visual Studio Development Server of Visual Studio 2008. However, when I . . .

Visual Studio 2003 remote debugging in Visual Studio . NET Debugging

. . . Visual Studio 2003 remote debugging in Visual Studio . NET Debugging Studio 2003 and 2008. And have been . . .

Visual Studio 2008 Remote Debugging like Visual Studio 6. 0 in Visual Studio . NET Debugging

. . . Visual Studio 2008 Remote Debugging like Visual Studio 6. 0 in Visual Studio . NET Debugging Is . . .

PROBLEM: Remote debugging managed code in Visual Studio . NET Debugging

. . . PROBLEM: Remote debugging managed code in Visual Studio . NET Debugging Need some help here. I'm using Visual Studio . . .

customized debug engine in Visual Studio Extensibility

. . . customized debug engine in Visual Studio Extensibility Recently, I am trying to develop a debugger used for romote debugging. My Host machine(PC) is windows running VS. Net . . . . . I am trying to develop a debugger used for romote debugging. My Host machine(PC) is windows running VS. Net, and . . .

Drag'n drop from File explorer in VS debugging session doesn't work with Vista in Windows Developer Winfx Avalon

. . . Drag'n drop from File explorer in VS debugging session doesn't work with Vista in Windows Developer Winfx . . . . . file exlorer work when doing that (dropfile) in a Visual Studio debugging session. The DragOver an" Drag'n drop from File explorer . . .

Visual Studio Debugging Stopped working. . . Visual Studio . NET

. . . Visual Studio Debugging Stopped working. . . Visual Studio . NET Visual Studio Debugging Stopped working. . .

I can not debug Text Interpreter in Visual Studio debugging extensibility. . . . in Visual Studio Extensibility

. . . I can not debug Text Interpreter in Visual Studio debugging extensibility. . . . in Visual Studio Extensibility I try to debug the Text Interpreter . . .

How to Zero In on Thrown Exceptions in Visual Studio

. . . How to Zero In on Thrown Exceptions in Visual Studio How to Zero In on Thrown Exceptions in Visual Studio By Peter Bromberg Printer Friendly Version View My FAQ's . . .

Visual Studio 2005: Cool Debugging Tricks

. . . Visual Studio 2005: Cool Debugging Tricks Visual Studio 2005: Cool Debugging Tricks By Peter Bromberg Printer Friendly . . .