**VS2010调试技巧进阶**

**跳到当前光标处（Ctrl+F10）**

　　我经常看到人们为了到达目标代码位置，而在程序中早早设定了断点，然后反复地按F10/F11，一步步走到目标代码处。当程序员的确需要仔细观察每一步的状态变化时， F10/F11是合理的选择。然而多数情况下，人们只想快速到达他们真正关心的代码处，这时候F10/F11就不是最佳选择了。

　　这时，你应该利用“跳到当前光标处”这个功能。先把光标定位在要测的目标代码行上，再同时按Ctrl和F10，被测程序将直接跳到该行停下。你再也不用按许多次F10/F11了。即使目标代码位于独立的类或方法中，你仍然可以从当前正在检查的地方跳过去。

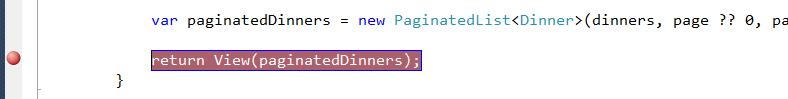
**条件中断**

　　另一种常见的情况是：开发人员设置断点，运行程序，利用不同的输入触发断点，然后在断点处手工检查是否满足某些特定的条件，从而决定是否继续调查。如果当前场景不是他们想要的，按F5继续运行程序，尝试别的输入，手动重复刚才的过程。

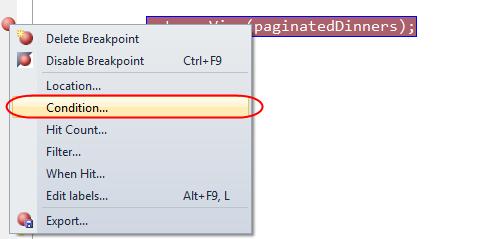
　　针对上述情况，Visual Studio提供了一个方便得多的功能——“条件中断”。只有当程序满足了开发人员预设的条件后，条件断点才会被触发，调试器中断。这将避免频繁地手工检查/恢复程序运行，大量减少调试过程中的手工和烦琐工作。

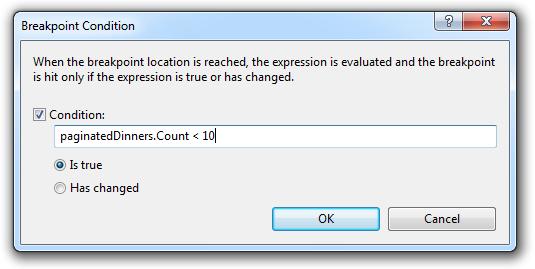
**如何设置条件断点**

　　设置条件断点非常容易。在特定的行上，按F9设置断点。



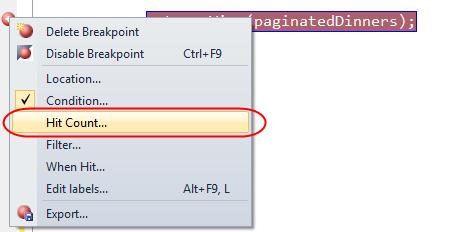
　　然后右击断点–编辑窗口左侧的红点，在上下文菜单上选择“Condition…”。

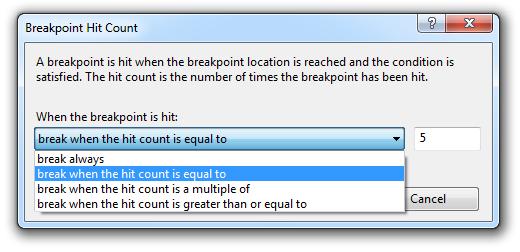
　　这时弹出一个对话框供你设置激活该断点所需的条件。比如：我们希望只有当局部变量paginatedDinners的尺寸小于10时，调试才中断。我们可以写出如下的表达式：

　　现在我再运行这个程序，实现搜索，只有返回值小于10时，程序运行才会被中断。对于大于10的值，该断点将被跳过。

**记录到达断点次数**

　　有时你希望，只有当第N次满足条件的运行到达断点时，才中断程序运行。例如：当第五次返回少于10份晚餐的查询结果时，中断程序运行。  
　　可以通过右击断点，然后在弹出菜单上选择“Hit count…”菜单命令实现。



　　这时系统弹出一个对话框，它允许你指定：（1）当满足条件，而且进入断点的累计次数等于N时，断点命中一次。（2）当满足条件，而且进入断点的累计次数是N的倍数时，断点命中一次。（3）当满足条件，而且进入断点的累计次数大于N时，每次命中断点。  


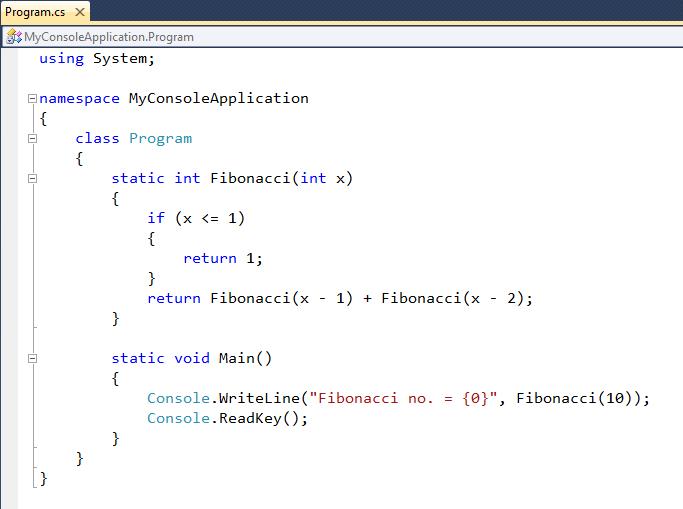
**机器/线程/进程过滤**

　　设置如下：右击断点；在弹出菜单上选择“Filter…”菜单命令;然后指定命中断点的特定条件:在指定的机器上、或指定的进程中、或指定的线程中。

**跟踪点—进入断点时的自定义操作**

　　许多人不知道“跟踪点(TrackPoints)”这个调试功能。“跟踪点“是种特殊的断点，当它被命中时，它会触发一系列自定义操作。如果你想观察程序的行为，而又不想中断调试的时候，这个功能尤其有用。

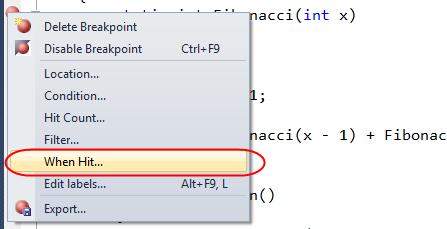
　　我将用一个简单的控制台程序来演示如何使用“跟踪点”。如下是斐波那契数列的一个递归实现：



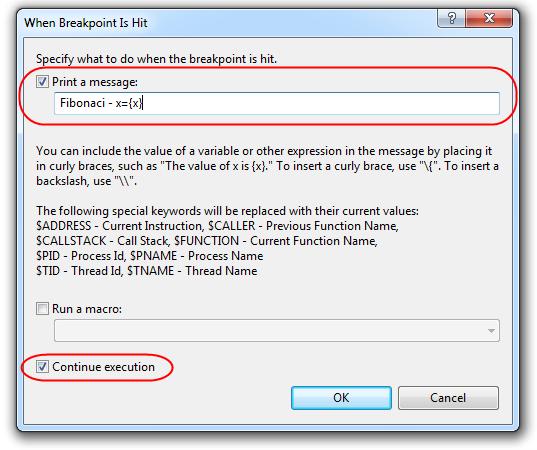
　　以上程序中，我们使用Console.WriteLine() 输出针对特定输入值生成的最终斐波那契数列。如果希望在调试器里观察操作中每一次递归运算后的数列而又不实际中断程序运行，该怎么办呢？“跟踪点”可以轻松实现。

**设置跟踪点**

　　你可以在特定的行上，按F9加跟踪点。然后  
　　右击断点，在上下文菜单中选择“When Hit…”：



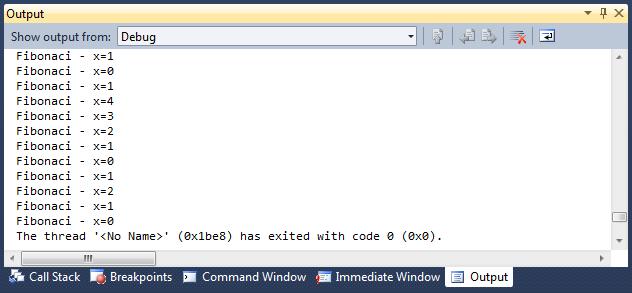
　　在弹出对话框上，你可以设置命中该断点时，所触发的事件。



　　在上面例子中，我们设定一旦命中断点时就打印追踪信息。注意，我们已经把局部变量“x”的值，作为追踪信息的一部分输出。局部变量可以通过{变量名}语法输出。你还可以利用系统内置的命令（$CALLER, $CALLSTACK, $FUNCTION等等），在追踪信息中输出常用的调试值。

　　在上例中，我们同时选中了底端的“continue execution“选项，这说明我们不希望程序中断调试状态，而是继续运行。唯一的不同是：每次断点条件满足时，我们的自定义追踪信息都将被输出。

　　现在当我们运行程序时，会发现自定义追踪信息自动显示在Visual Studio的“输出“窗口里。这让我们很容易看到程序的递归调用过程：



　　你也可以选择往应用程序中添加一个自定义追踪信息的监听器。这时追踪点的输出信息将通过它输出，而不是Visual Studio的“输出“窗口。

**跟踪点—运行自定义的宏**

　　上周，我在伦敦演讲时，有听众问道：当命中跟踪点时，能否自动输出所有的局部变量？

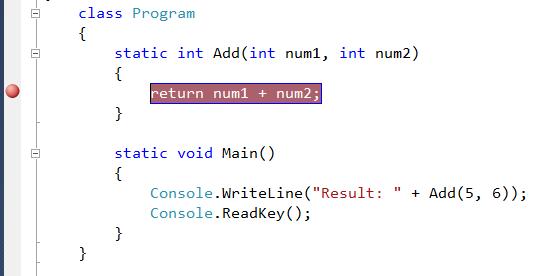
　　Visual Studio中并没有这样的内置功能，但我们可以写一个自定义宏来实现，然后在命中跟踪点时调用该宏。这个的实现需要先打开Visual Studio的宏编辑器（工具->宏->宏IDE菜单命令），然后在项目资源管理器的MyMacros节点下选择一个模块或创建新模块（如：加个名为“UsefulThings”的模块），再把下面的VB宏代码贴到模块中并保存。

Sub DumpLocals()  
Dim outputWindow As EnvDTE.OutputWindow  
outputWindow = DTE.Windows.Item(EnvDTE.Constants.vsWindowKindOutput).Object  
  
Dim currentStackFrame As EnvDTE.StackFrame  
currentStackFrame = DTE.Debugger.CurrentStackFrame  
  
outputWindow.ActivePane.OutputString(“\*Dumping Local Variables\*” + vbCrLf)  
For Each exp As EnvDTE.Expression In currentStackFrame.Locals  
outputWindow.ActivePane.OutputString(exp.Name + ” = ” + exp.Value.ToString() + vbCrLf)  
Next  
End Sub

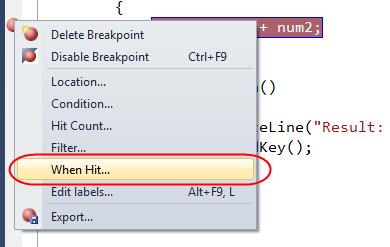
　　上述宏代码将循环当前的堆栈，把所有的局部变量输出到“输出”窗口。

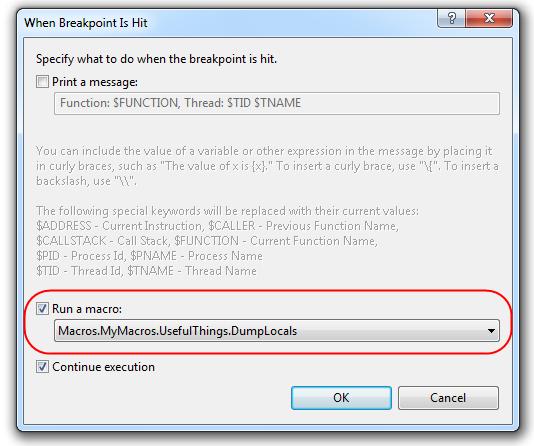
**使用自定义的“DumpLocals”宏**

　　然后，我们可以在如下的一个简单程序中使用刚定制的“DumpLocals”宏了：



　　上述代码中，我们用F9在“Add”方法的返回值处加了个断点，然后右击断点，在弹出菜单上选择“When hit”。



　　将显示如下对话框。和之前不一样， 我们不选“Print a message”选项，也不手工设定需要输出的变量；而是选择“Run a marco”复选框，并指定到我们上面创建的UsefulThings.DumpLocals宏上：  


　　为了使程序能在命中跟踪点后仍继续运行，我们将继续选中“continue execution”复选框。

**运行程序**

　　现在按F5运行程序，当“Add”方法被调用时，我们会在Visual Studio的“输出”窗口中看到如下结果。注意命中跟踪点时，宏会自动列出每个局部变量的名称和值:  
