Debugging

Debugging UI - Windows in Visual Studio

Some of the windows that can help when debugging

# Locals

this shows the variables that are local to the method in focus. This save adding to one of the watches window.

The locals window also shows $exception which shows the current exception, $Exception is a pseudo variable, full details of this and other pseudo variabl;es available at

<http://msdn.microsoft.com/en-us/library/ms164891.aspx>

# Auto

shows the variables used in the current statement and the previous one.

# Output

this display status messages for various features.

# Breakpoint

this window allows the managing of breakpoints.

# 

# Call stack

this shows functions , methods on the call stack, Various details can be switched on and off by right clicking the header row. You can also, when there is an exception unwind the frame - with managed code this is normally done automatically. You can also show calls from other threads.

# Immediate

debug and evaluate expressions, execute statements, print variable values, and so forth. It allows you to enter expressions to be evaluated or executed by the development language during debugging.

# Modules

shows shows the dlls, exe used by the program and related information - such as if the symbols are loaded. Symbols can be loaded from here.

# Processes

shows processes launched or attached to from visual studio.

# Memory

allows you to watch an area of memory

# Threading

shows details of which threads are running, allows you to name them, freeze them, thaw them

# Parallel stacks

in multi-threaded applications, this has 2 views. 1 shows the call stack for each thread, 2 the task view for managed code applications allows you to see the call stack for Task objects and promise .then code (javascript)

# Watch window

this window allows you to evaluate variables and methods, Values may be edited.

Sometimes when you are watching an object it goes out of scope and you can no longer see it for this we can use Object Id, right clicking an object in the watch window gives the option to Maske Object Id, select this gives the object a number like 1#, add this 1# to the watch window gives you the same view as the object but it doesn’t go out of scope, it stays visible until garbage collection actually runs.

# Quick watch

is similar but it is modal, avoids cluttering the watch window.

Visual Studio Settings

There are a number of setting in visual studio that have an impact on debugging.

# Overflow/underflow exceptions

By default visual studio compiled programs don’t report overflow and underflow errors e.g.if you have a byte value 255, and add 1 to it you get 0.

Overflow/underflow errors often come to light a while after the actual error - e.g. the value may be written to database , analysed a week later. Finding out the cause when the problem occurred the week before is not easy. If you look at the properties of the project under build, advance there is a checkbox which makes this an exception.

# Debugging options

Tools-options debug general

There are a number of options that change the way the debugger behaves.

Just my code (default) means that you only debug your code see exceptions in your code, Sometimes you may wish to be able to debug frameworks etc.

To distinguish user code from non-user code, Just My Code looks at three things: DBG Files, PDB files, and optimization

This setting also affects the exceptions window. From vs2013 this setting also applies to javascript jquery

Unwind the call stack on unhandled exceptions -with this set, if an unhandled exception is thrown, the call stack is unwound to before the exception was thrown. This allows the use of immediate window, watch window etc to resolve the exception.

# Exceptions

Under debug - exceptions. Here we can denote which exceptions the code will break at, if thrown is ticked when throw, or user-unhandled. User unhandled only available when just my code ticked

This can be done for individual errors, if just my code is unticked, only the thrown column is available

Debugging attributes

There are attributes that can be added to the defintion of a class, that help when viewing an instance of that class via watch windows etc.

# DebuggerDisplay

allows you to specify what is display when looking at an object instance.

# DebuggerBrowsable

allows you to specify how a property or field of a class is display .

Other available features

# Visualisers

you can write your own visualiser a built in example iof a visualiser is the xml visualiser - details <http://msdn.microsoft.com/en-us/library/zayyhzts.aspx>

# DebuggerTypeProxy Class

this allows you to project you class through a proxy class. The proxy class is then seen in the watch window details http://msdn.microsoft.com/en-us/library/d8eyd8zc.aspx

Breakpoints

are seen as an important focus of debugging there are various types, that apply to .net code and javascript and jquery.

# Data breakpoints

In managed code you can not set a breakpoint for when a variable changes, but you can achieve the same thing by setting a breakpoint on the set part of a property. Note the focus has to be the set part of the command. For breakpoints in properties to work you need to clear the Step over properties and operators debug setting. Private fields can be converted into private properties. In release mode this gets optimised out so a call to get the value of the property is direct to the backing field so doesn’t harm performance.

# Break on return, method return values.

This can be set via the call stack. If you step into a mthod and look at the call stack you will see the return step, breakpoints can be set in the call stack.all stack The values of returns from methods and functions are displayed in the autos window and there is anothe pseudo variable called $ReturnValue!. We shoudl think of debugging when writing code (easy to read) so it may be wise to put the return value into a variable and return that variable. These get optimised out in release mode.

The following breakpoints are set by right clicking an existing breakpoint and selecting the option.

# Condition breakpoints

Here a condition can be set on a breakpoint such as x==5 and the breakpoint will stop the code when x==5. There has been some change in the way this performs between vs2012 and vs2013, before 2013 you could do things such as x=5 on the breakpoint and so change the value. Full explanation of the change available on John Robbins blog. - <http://www.wintellect.com/blogs/jrobbins/a-behavior-change-in-visual-studio-2013-.net-breakpoints>

# HitCount

this allows you to set a breakpoint that will only break on the nth time, mutiple off or if greater than.

N can be found out by setting the hit count very high, running the program, when it errors look at the breakpoint window, and the current hit number is show.

What options you have chosen for debugging make a difference to what you do next, if rewind on exception then subtract 1 - else use the number

Filter -these are predominantly for multi threaded apps - <http://blogs.msdn.com/b/visualstudioalm/archive/2013/10/09/filter-breakpoints.aspx>

# When hit

allows you to define something to happen when a breakpoint is hit, The breakpoint does not have to pause execution. They can be used like tracepoints. Also you can have code that changes the value of something . Note in the debugger scope does not apply so private methods from another class may be called.,

Intellisense may need to be turned off if you wish to set a value.

# Break at Function

allows you to set breakpoints by the function name - sets breakpoints at all functions with that name.

# Label

allows the labelling of breakpoints, mainly used in export and import, or to refer to them in the breakpoint window

# Data tips

allow you to pin the watch of a variable on the screen, and change its variable - comments can also be added

# Import and export

debugging a problem various breakpoints set up, close of play , export breakpoints along with data points send to colleague. Also backup against accidental delete.

Dumps

# Examining dumps

A dump can be taken through task manager or another tool on a machine without visual studio. That dump can then be opened on a machine with visual studio and the state of play seen at the time the dump was taken.

This is all dependent on symbol servers.

The symbol server keeps record of the pdb files giving details of symbols required for debugging. Having a symbol server allows the system to access code from for any version that has a pdb file on the server. Microsoft have 30- 40 thousand versions of windows running somewhere at Redmond and they can access the code for any one of these versions. Each build produces new pdb files, if you wish to debug a build you need the right pdb files, especially important with things that relate to production debugging like dumps.

Other tools

# Page inspector

this works with IE9 and upwards it allows similar features to browser debuggers to inspect parts of a web page. This is an addin to visual studfio - as part of the azure sdk. Where it has a benefit over browser debuggers is that when an item is selected you are shown the file that has that element in, e.g the partial view, the layout page.

# Code map

ultimate and premium - allows to see the call stack visually and its dependencies. Makes it easier to understand code relationships, particularly when getting involved in a new or unfamiliar codebase. https://www.youtube.com/watch?v=KpNkDIrB-5w

# Intellitrace

in ultimate but brings debugging into the 21st century allows the user to reply the running of a program. Move the running forward and backwards. Trace files from machines can be taken, resolving the unable to reproduce bug problem.

# Windbg

is a debugging tool that allows you to get into the very depths of a program - looking at memory location, debug minsweeper -<http://blogs.msdn.com/b/debuggingtoolbox/archive/2014/05/14/hacking-minesweeper-for-windows-8.aspx>

# SOS

an extension to Windbg - for .net code.

Other useful links

Advance debugging presentation devdays

<http://channel9.msdn.com/Events/DevDays/DevDays-2011-Netherlands/Devdays084>

Visual Studio book

<http://proquestcombo.safaribooksonline.com.libezproxy.open.ac.uk/book/-/9781118416488>

A link on how to go about debugging though it is .net 2.0 it is useful, also perfomance debugging and reoslving hangs

<http://blogs.msdn.com/b/tess/archive/2008/02/04/net-debugging-demos-information-and-setup-instructions.aspx>

Best practices for DebuggerDisplay

<http://blogs.msdn.com/b/jaredpar/archive/2011/03/18/debuggerdisplay-attribute-best-practices.aspx>

Intellitrace with System Center

<https://www.youtube.com/watch?v=lkVeykQGDQg>

What every developer should know about pdb files

<http://www.wintellect.com/blogs/jrobbins/pdb-files-what-every-developer-must-know>