Delphi® XE2
The fastest way to build native Windows, Mac and iOS applications
Windows
Mac
iOS
Cloud
64-bit
FireMonkey
Database

Delphi XE2 Debugging

Jeroen Pluimers
jpluimers@better-office.com
better office benelux

DelphiLive 2011
DE, Düsseldorf, 20111026

Lot's of new debugger stuff

in the most recent Delphi versions

Demos @

- better office
  - http://bo.codeplex.com
  - http://bo.codeplex.com/SourceControl/BrowseLatest
  - https://bo.svn.codeplex.com/svn (SVN)
- RAD Studio Demos (XE/XE2)
  - http://sourceforge.net/projects/radstudioverins/
  - https://radstudioverins.svn.sourceforge.net/svnroot/radstudioverins (SVN)

New in 2010

- http://docwiki.embarcadero.com/RADStudio/en/What%27s_New_in_Delphi_and_C%2B%2BBuilder_2010#Debugger_Changes
- Two types of custom data visualizers can now be created and plugged in by using the Tools API. The product includes several built-in debugger visualizers:
  - TDateTime (Delphi and C++)
  - std::string and std::wstring (C++ only)
  - TStringList (Delphi and C++)
- You can select specific visualizers on the Tools > Options > Debugger Options > Visualizers dialog box and on the various debug windows that support visualizers. See Debugger Visualizers and Enabling/Disabling Debugger Visualizers. The video Debug Visualizer in RAD Studio 2010 shows more about the visualizers.
- The Event Log has been renovated in the following ways:
  - The Event Log is now implemented as a TVirtualStringTree rather than as a TStringGrid. This change makes logging faster.
  - You can stop the scrolling of events by clicking inside the Event Log window and selecting an event (the Scroll new events into view option must be selected on the Tools > Options > Debugger Options > Event Log page) (see Event Log Options).
  - Multiline events in the Event Log now appear on discrete lines, as follows:
    - Event type appears on first line
    - Event text appears on following lines
    - Process information appears on the final lines
  - Lengthy event messages can be viewed in a hint window by hovering the mouse over the event in the Event Log.
- In multithreaded applications:
  - You can now set breakpoints on specific threads. See Setting and Modifying Breakpoints.
  - You can now freeze/thaw specific threads. See Thread Status.
- The Allow side effects in new watches option has been moved from the Embarcadero Debuggers page to the Tools > Options > Debugger Options page and is now renamed Allow side effects and function calls in new watches. This option on the Debugger Options dialog box is the same as the option Allow side effects and function calls on the Watch Properties dialog box.
- The Registers pane in the CPU Windows has three new Follow context menu commands. Each of these commands positions one of the other panes in the CPU view to the address contained in the currently selected register:
  - Follow > Near Code positions the Disassembly pane to the address contained in the currently selected register.
  - Follow > Offset to Data positions the Memory pane to the address contained in the currently selected register.
  - Follow > Offset to Stack positions the Stack pane to the address contained in the currently selected register.
- The following Debug windows have new context menu commands:
  - On the Debug Inspector:
    - Watch
    - Evaluate/Modify
    - Visualizers
  - On the Watch List Window:
    - Evaluate/Modify
    - New Watch
    - Visualizers

New in XE

- http://docwiki.embarcadero.com/RADStudio/en/Debugger_Changes_for_XE
- Auto close views after debugging:
  - You can now specify that one or two specific debugger windows (the CPU view and the Modules view) are to be closed when you exit the debugger. This is in addition to optionally closing all files that were implicitly opened by the debugger. Set the Auto close views after debugging option on the Tools > Options > Debugger Options dialog box.
- For multithreaded applications:
  - You can temporarily name a thread while you are debugging so that you can more easily track threads on the Thread Status window. Use the Name Thread context menu command; see Naming a Thread While Debugging.
  - The name of the current thread (when available) now appears in the caption in the following views:
    - Main IDE window
    - Watch List Window
    - Local Variables Window
    - Call Stack Window
    - Evaluate\Modify dialog box
    - Debug Inspector
    - FPU view
    - All Standalone CPU views:
      - Disassembly pane
      - CPU Stack pane
      - Registers pane
      - Memory pane
- Non-user breakpoints:
  - The Event Log Window now displays "Non-user breakpoint" when a non-user breakpoint is encountered – that is, a breakpoint that was not set using the RAD Studio debugger. (To specify that the debugger is to ignore non-user breakpoints, check Ignore non-user breakpoints on the Tools > Options > Debugger Options > Embarcadero Debuggers dialog box.)
- Disabling the "Source has been modified. Rebuild?" prompt:
  - You can now specify that this prompt does not appear if changes are made to the source code while you are debugging. If you uncheck the Prompt to rebuild projects modified while debugging option on the Tools > Options > Debugger Options dialog box, the debugger will no longer prompt to rebuild the project, and performs the actions you request without rebuilding.
- Run Without Debugging is now on the Debug toolbar:
  - The Run Without Debugging button (F9) has been added to the debug toolbar.
- Breakpoint List and Watch List now have editable fields:
  - The field labels on the Breakpoint List Window and the Watch List Window are now clickable dropdown lists. Click a field label (such as Line/Length) to select from values that were entered in this field previously or on other debug windows.
- Visualizer icon is now clickable on the Debug Inspector:
  - If you inspect an item that has an external-viewer visualizer associated with that data type, you can now click the icon or use the Visualizers context menu command to invoke the visualizer. See Debugger Visualizers.

New in XE2 (1/3)

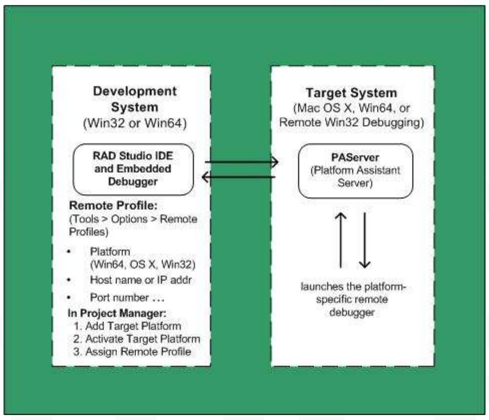
- http://docwiki.embarcadero.com/RADStudio/en/Debugger_Changes_for_XE2
- Cross-Platform Debuggers
  - In addition to the integrated native Embarcadero Win32 Debugger, RAD Studio has added two cross-platform debuggers:
    - Embarcadero Mac OS X Debugger
    - Embarcadero Win64 Debugger
  - The appropriate platform-specific cross-platform debugger runs on the target platform with your cross-platform application. You use the cross-platform debugger in the IDE in the same way you use the integrated debugger for native Win32 debugging.
  - For more information about the new debugging solution, see Debugging Cross-Platform Applications.
- New Remote and Cross-Platform Debugging Solution
  - The new remote and cross-platform debugging solution requires:
    - The Platform Assistant remote application server on the remote system (Win32, Win64, or OS X)
    - A remote profile on the development system that describes the remote target
  - The new debugging solution also enables you to use the new Deployment Manager to manage the deployment of your project.
  - The old-style remote debugging procedures are still supported, but we recommend that you start using the new remote debugging solution immediately.
  - For more information about the new debugging solution, see Debugging Cross-Platform Applications.
- New Options for 'Load Process' and 'Attach to Process' Dialog Boxes
  - On all four of the Load Process dialog boxes and Attach to Process as well, you now have the ability to select the debugger that will be used. You can choose:
    - Embarcadero Win32 Debugger is the default embedded debugger for 32-bit Windows applications.
    - Embarcadero Win64 Debugger is specifically designed for debugging applications that target 64-bit Windows.
    - Embarcadero Mac OS X Debugger is specifically designed for debugging cross-platform applications that target a Mac running OS X.
    - Choose the debugger that matches the target platform of the process you want to load.
  - On Load Process Remote:
    - You now specify the Remote Host and Remote Path differently if you are using the new remote and cross-platform debugging solution as opposed to the old remote debugging solution. For example, in order to enter the host name using the new remote and cross-platform debugging solution, you click the ellipsis ([...]) to specify a remote profile that contains the host name.
    - The Remote Path can be specified as a ./ (dot-slash) relative path, which is relative to the remote profile directory on the target platform (the default location of output files associated with the current remote profile).
    - When you choose the Embarcadero Mac OS X Debugger, a field becomes visible and enabled: Use launcher application. The command-line field is preloaded with the command to run Xterm on the Mac. You can, however, change the command line, so that you can use an alternate terminal emulator on the Mac.

New in XE2 (2/3)

- http://docwiki.embarcadero.com/RADStudio/en/Debugger_Changes_for_XE2
- New Options for 'Load Process' and 'Attach to Process' Dialog Boxes
  - ...
  - On Load Process Local and Load Process Remote, the checkbox for "Execute startup code on load" has been replaced by the following After load: fields:
    - Do not run - loads the executable but does not execute any code, equivalent to F7 (Step Into)
    - Run to program entry point - run to the first entry point, such as main(), equivalent to F8 (Step Over)
    - Run to first source - run to the first line of source code, equivalent to Shift-F7 (Trace to Next Source Line)
    - Run - Run to completion (or, if errors occur, enter Debug mode), equivalent to F9 (Run)
    - To use these options, click Run > Load Process.
  - On Attach to Process:
    - You can select one of the three Embarcadero debuggers (Win32, Win64, Mac OS X).
    - You can specify the Remote Machine by associating a remote profile if you are using the new remote and cross-platform debugging solution.
- Figures Show 64-bit Registers
  - The help contains figures that represent the registers, as follows:
    - The following 64-bit registers are shown in Registers pane:
      - x64 CPU General Purpose Registers
      - x86 FPU Data Registers
      - x64 SSE Data Registers
    - The following registers are shown in FPU:
      - x86 FPU Data Registers
      - x64 SSE Data Registers
- See Also
  - Debugging Cross-Platform Applications

New in XE2 (3/3)

- Platform Assistant Server


Development System
(Win32 or Win64)
RAD Studio IDE and Embedded Debugger
Remote Profile:
(Tools > Options > Remote Profiles)
Platform (Win64, OS X, Win32)
Host name or IP addr
Port number ...
In Project Manager:
1. Add Target Platform
2. Activate Target Platform
3. Assign Remote Profile
Target System
(Mac OS X, Win64, or Remote Win32 Debugging)
PAServer
(Platform Assistant Server)
launches the platform-specific remote debugger

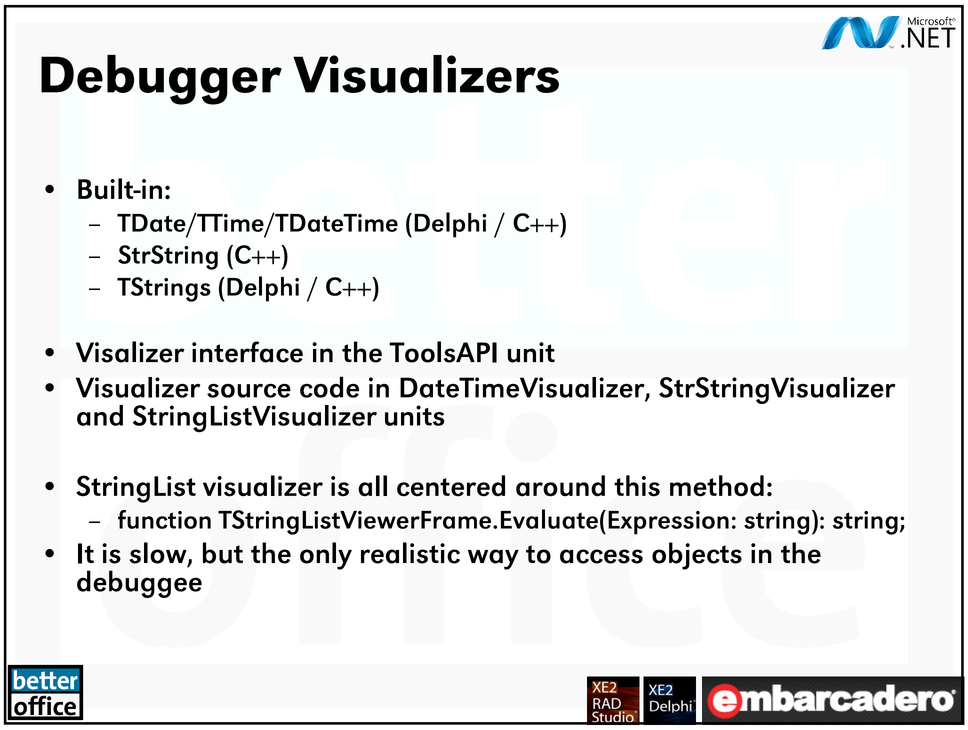
- http://docwiki.embarcadero.com/RADStudio/en/Debugging_Cross-Platform_Applications

PAServer

- implicit in x64
  - dbkw64_16_0.exe
- Explicit for Mac Debugging
  - Show demo

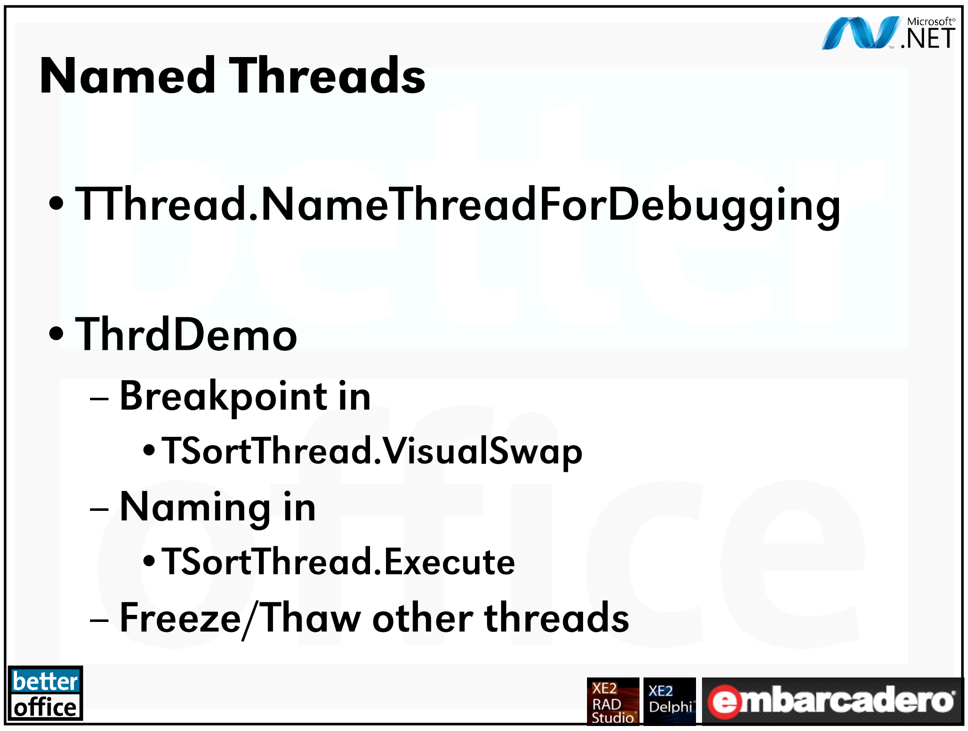
“Conventional” Debugging Tips

- Important Project Options
  - Stack Frames – ON
  - Optimizations – OFF
  - Debug DCUs – ON
  - Debug Info (TD32) – ON



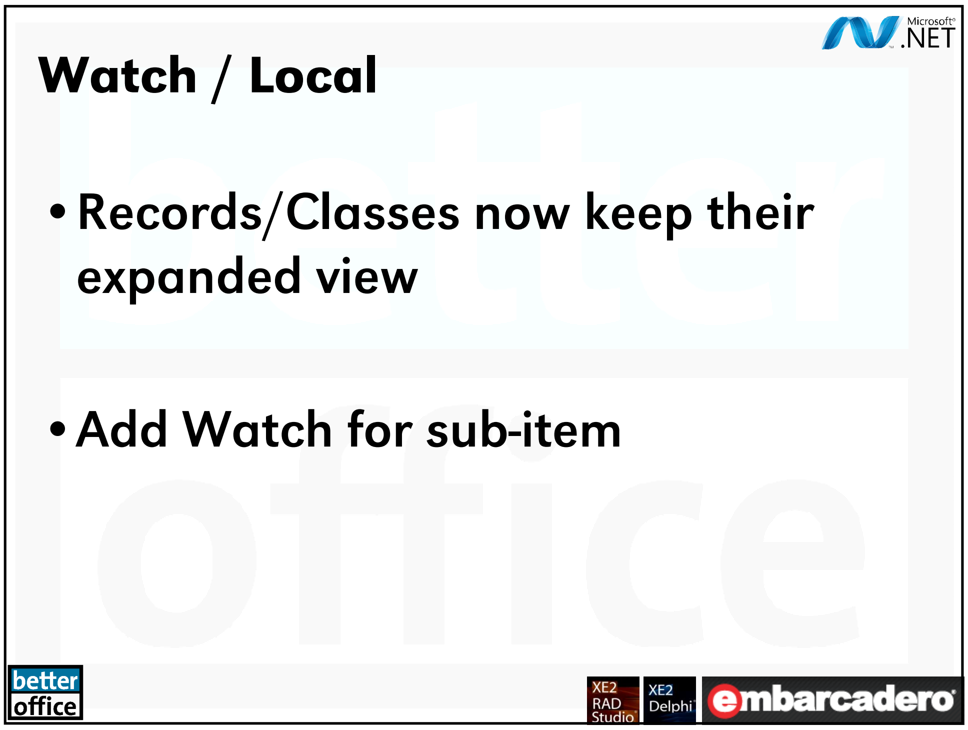
Debugger Visualizers

- Built-in:
  - TDate/TTime/TDateTime (Delphi / C++)
  - StrString (C++)
  - TStrings (Delphi / C++)
- Visalizer interface in the ToolsAPI unit
- Visualizer source code in DateTimeVisualizer, StrStringVisualizer and StringListVisualizer units
- StringList visualizer is all centered around this method:
  - function TStringListViewerFrame.Evaluate(Expression: string): string;
- It is slow, but the only realistic way to access objects in the debuggee



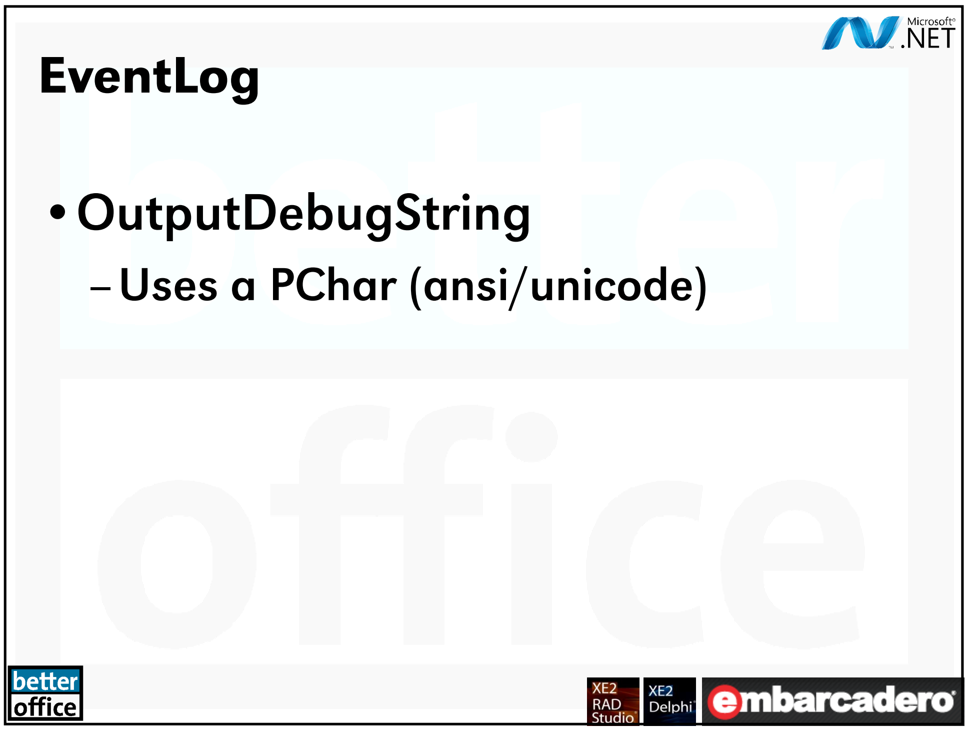
Named Threads

- TThread.NameThreadForDebugging
- ThrdDemo
  - Breakpoint in
    - TSortThread.VisualSwap
  - Naming in
    - TSortThread.Execute
  - Freeze/Thaw other threads



Watch / Local

- Records/Classes now keep their expanded view
- Add Watch for sub-item



EventLog

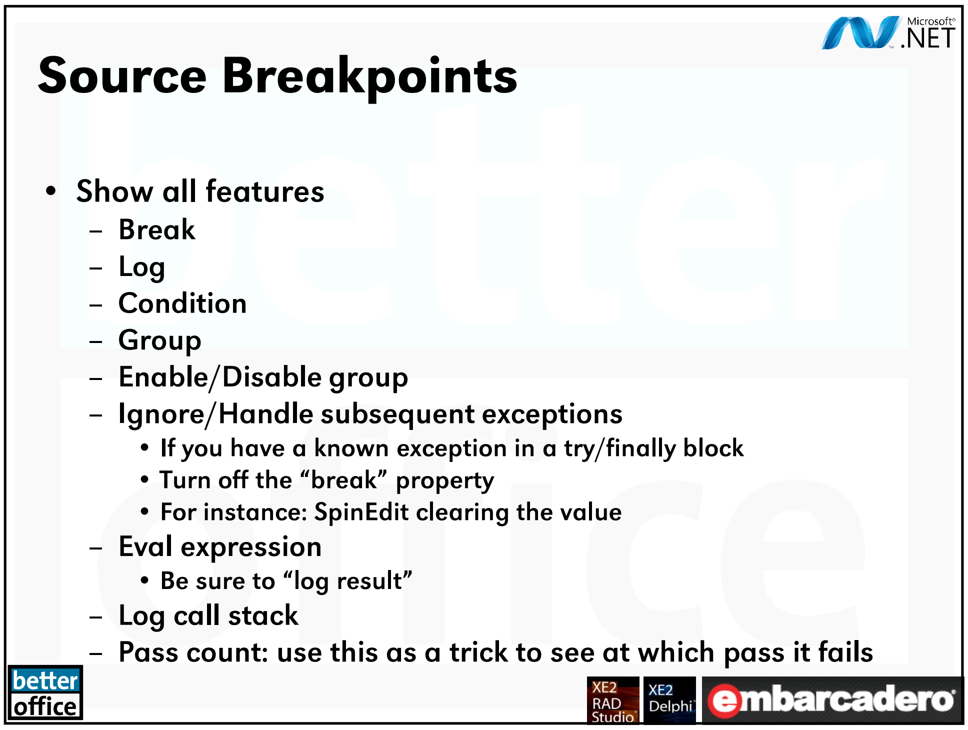
- OutputDebugString
  - Uses a PChar (ansi/unicode)

Dragging in the gutter

- Move breakpoint by dragging
- Copy breakpoint by Ctrl + dragging
- Move execution pointer
- (Works for bookmarks too!)

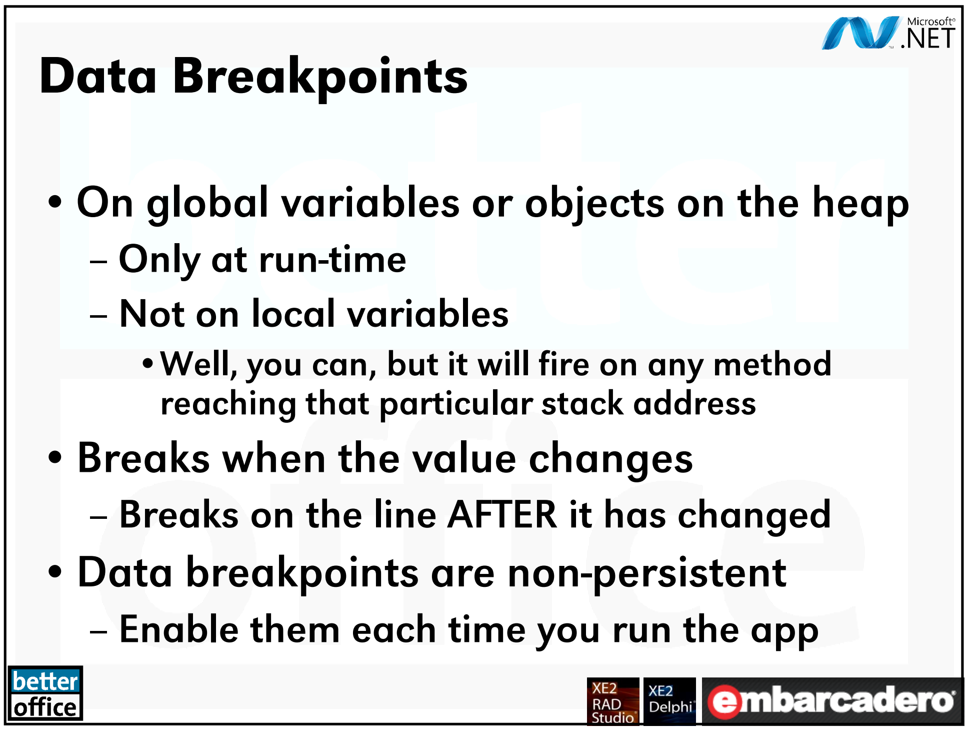
x64 debugging: Output Directories

- Before 2010 it was empty
- In 2010/XE it was
  - .\ $(Config)\$(Platform)
- Since XE2 it is
  - .\$(Platform)\$(Config)
  - Except for DUnit:
    - http://qc.embarcadero.com/wc/qcmain.aspx?d=90084
- If you get an error like this:
  - [DCC Fatal Error] xxxx.dpr(7): F2048 Bad unit format: 'yyyy.dcu' - Expected version: 23.0, Windows Unicode(x64) Found version: 23.0, Windows Unicode(x86)
- make sure you set these directories on all pre-XE2 projects before debugging in x64:
  - Output Directory
  - Unit Output Directory
- this doesn't always work: sometimes you have to recreate the .dpr/.dproj
  - This is in QC and should be fixed soon http://qc.embarcadero.com/wc/qcmain.aspx?d=100309



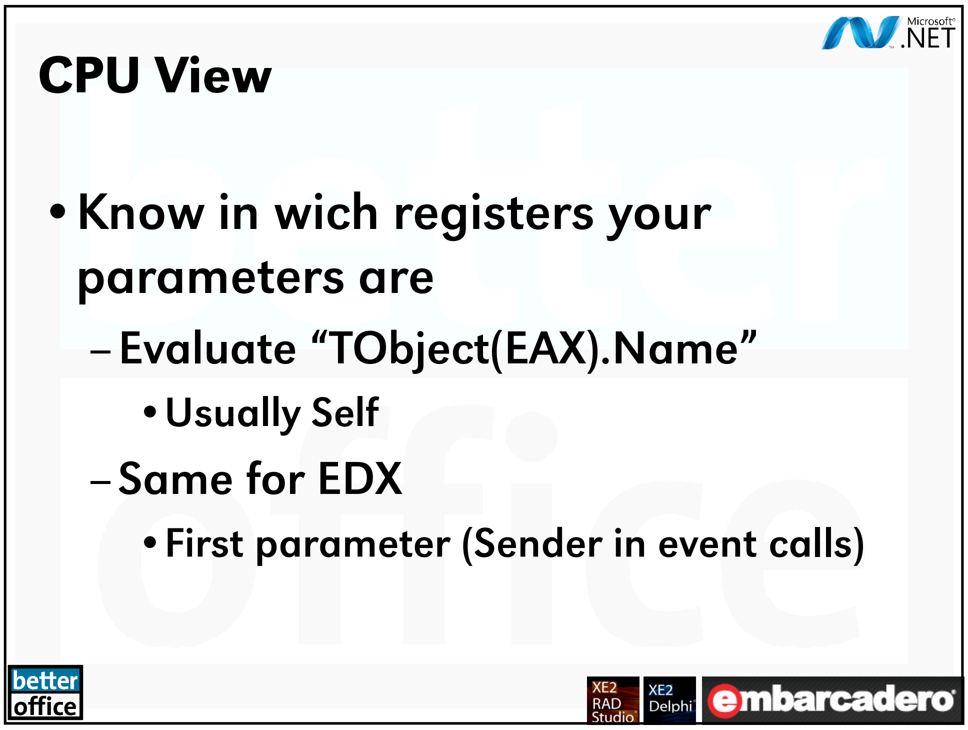
Source Breakpoints

- Show all features
  - Break
  - Log
  - Condition
  - Group
  - Enable/Disable group
  - Ignore/Handle subsequent exceptions
    - If you have a known exception in a try/finally block
    - Turn off the “break” property
    - For instance: SpinEdit clearing the value
  - Eval expression
    - Be sure to “log result”
  - Log call stack
  - Pass count: use this as a trick to see at which pass it fails



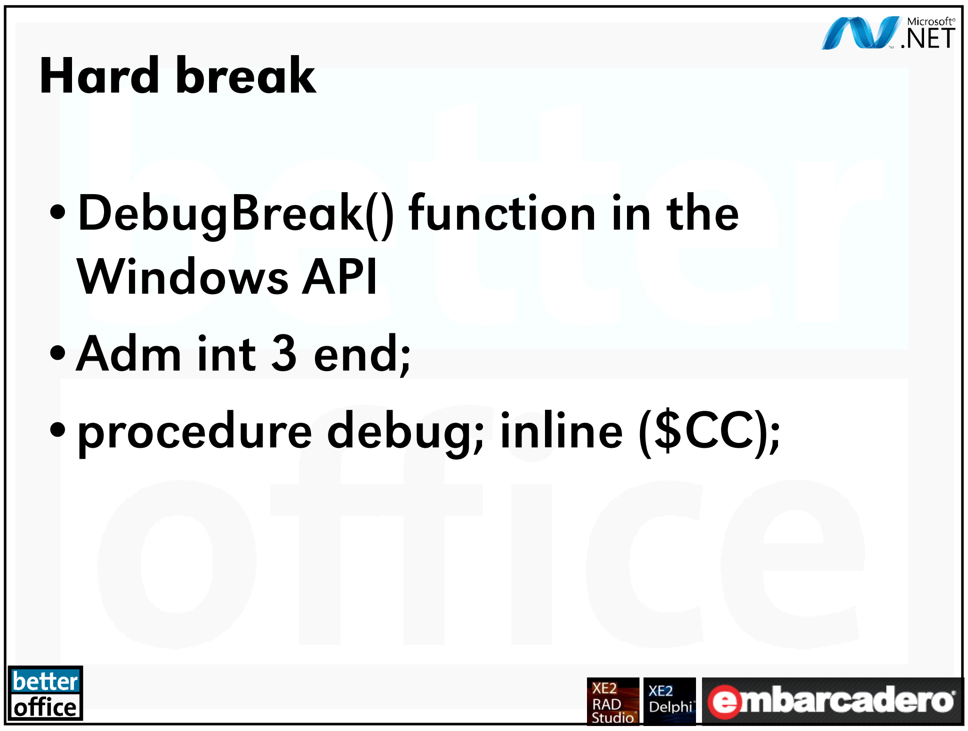
Data Breakpoints

- On global variables or objects on the heap
  - Only at run-time
  - Not on local variables
    - Well, you can, but it will fire on any method reaching that particular stack address
- Breaks when the value changes
  - Breaks on the line AFTER it has changed
- Data breakpoints are non-persistent
  - Enable them each time you run the app



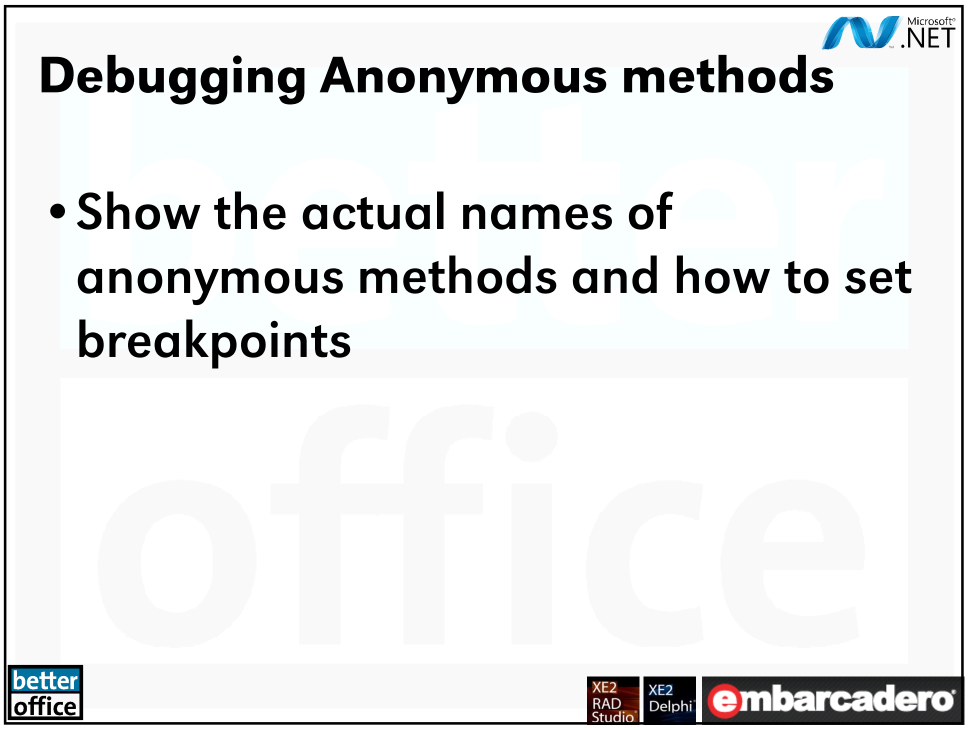
CPU View

- Know in wich registers your parameters are
  - Evaluate “TObject(EAX).Name”
    - Usually Self
  - Same for EDX
    - First parameter (Sender in event calls)



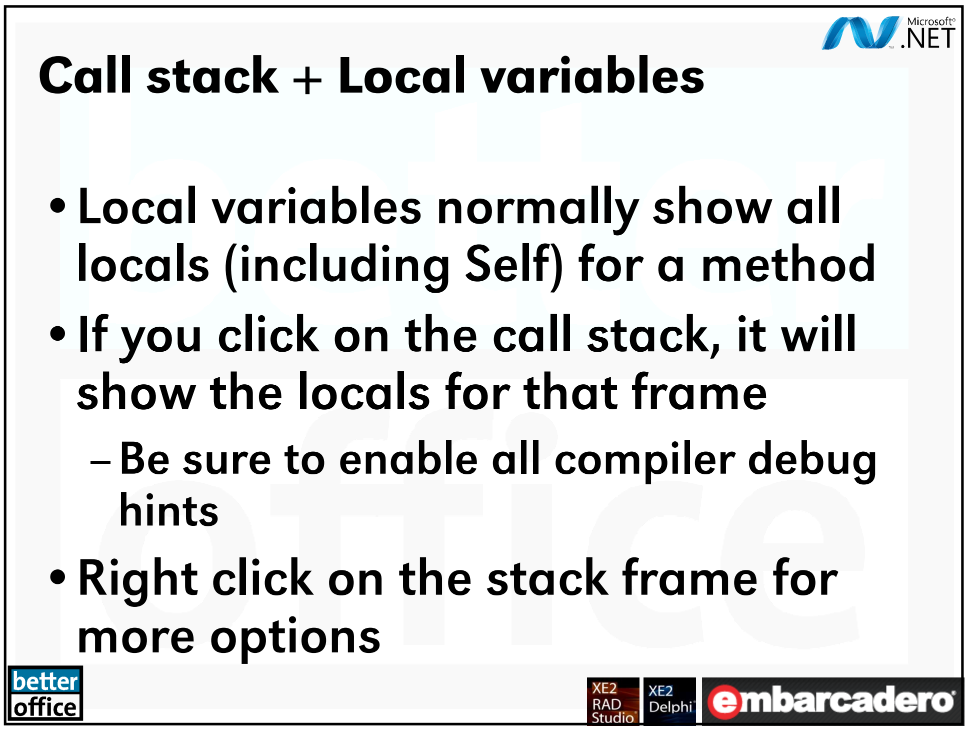
Hard break

- DebugBreak() function in the Windows API
- Adm int 3 end;
- procedure debug; inline ($CC);



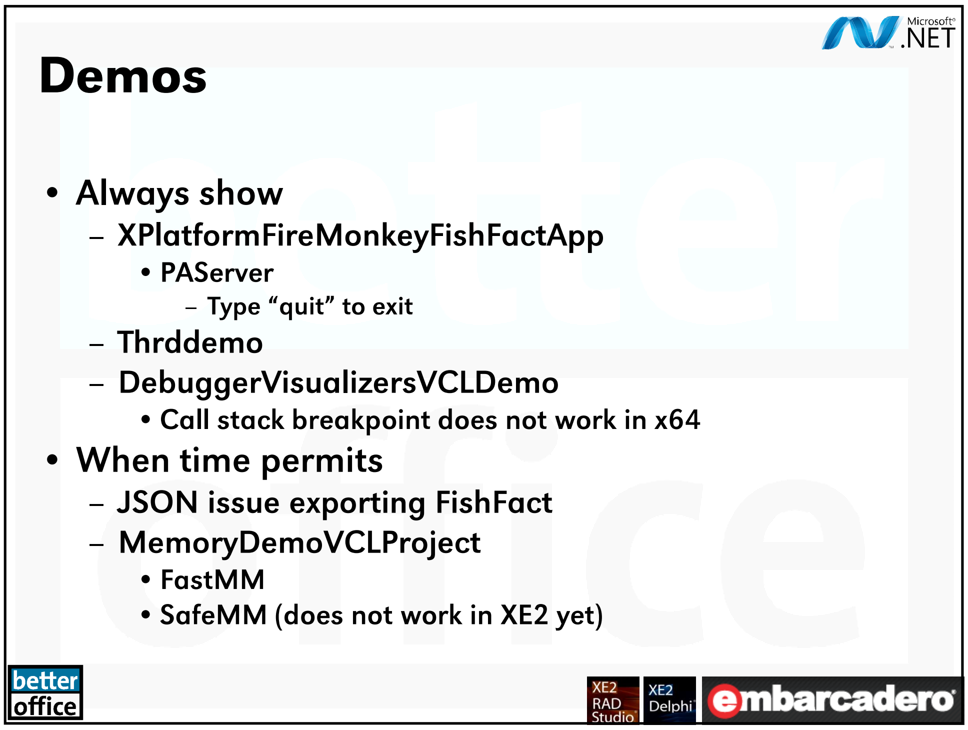
Debugging Anonymous methods

- Show the actual names of anonymous methods and how to set breakpoints



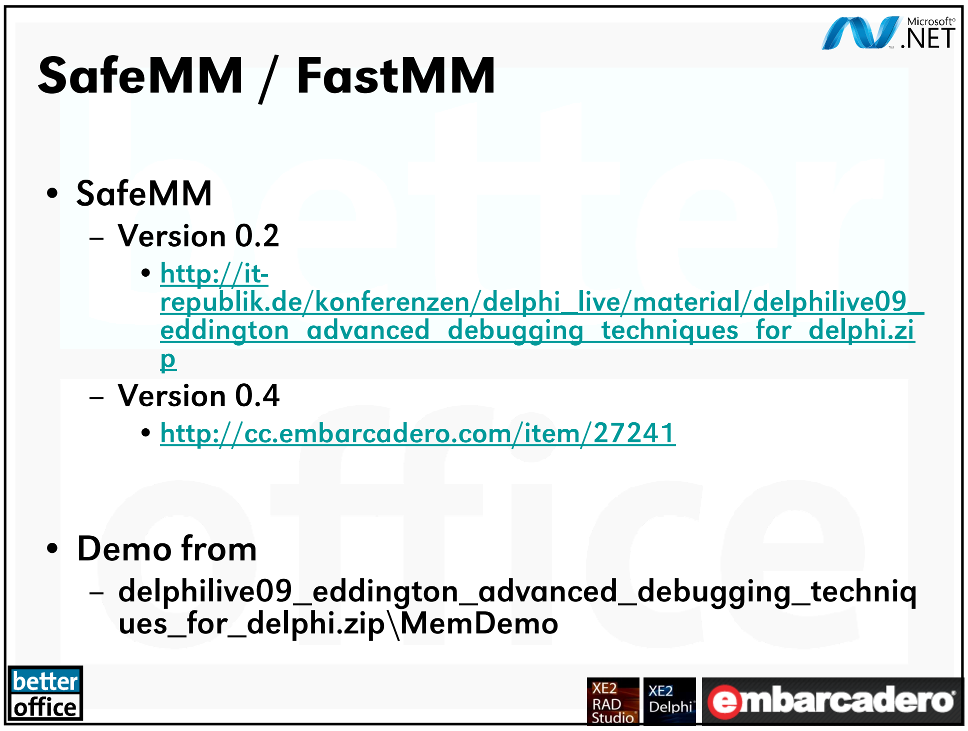
Call stack + Local variables

- Local variables normally show all locals (including Self) for a method
- If you click on the call stack, it will show the locals for that frame
  - Be sure to enable all compiler debug hints
- Right click on the stack frame for more options



Demos

- Always show
  - XPlatformFireMonkeyFishFactApp
    - PAServer
      - Type “quit” to exit
  - Thrddemo
  - DebuggerVisualizersVCLDemo
    - Call stack breakpoint does not work in x64
- When time permits
  - JSON issue exporting FishFact
  - MemoryDemoVCLProject
    - FastMM
    - SafeMM (does not work in XE2 yet)



SafeMM / FastMM

- SafeMM
  - Version 0.2
    - http://it-republik.de/konferenzen/delphi_live/material/delphilive09_eddington_advanced_debugging_techniques_for_delphi.zip
  - Version 0.4
    - http://cc.embarcadero.com/item/27241
- Demo from
  - delphilive09_eddington_advanced_debugging_techniques_for_delphi.zip\MemDemo

Thanks for some ideas

- Delphi R&D team
  - Chris Hesik
    - http://cc.embarcadero.com/item/27263
      CodeRage 5 – What's New in the Delphi and C++Builder Debugger
  - Mark Edington
    - http://cc.embarcadero.com/item/27285
      CodeRage 4 - Debugging Techniques for Delphi
- CodeRage speakers
  - Francois Gaillard
    - http://cc.embarcadero.com/item/26446
      CodeRage 3 - Delphi Debugging for Dummies
  - Robert Love
    - http://cc.embarcadero.com/item/28594
      CodeRage 6 - Exploring the Delphi Debugger

Q & A

Jeroen Pluimers
better office benelux
jpluimers@better-office.com

If you have questions after the workshop, please mail me

my blog: http://wiert.wordpress.com
code repository: http://bo.codeplex.com