Debuggowanie w VS2010

Rafał Zarębski

- F5 Start Debugging/Continue
- F9 Place breakpoint
- F10 Step over
- F11 Step into

Set Next Statement

- Ustawia następna instrukcje która będzie wykonana
- Można cofać program, pomijać fragmenty kodu
- Ctr-Shift-F10 lub drag the arrow

Etykiety dla breakpointów

- Pozwalają porządkować breakpointy w większych projektach
- Można filtrować breakpointy i wykonywać na nich operacje (okno breakpoints)

Conditional Breakpoint

- Program zatrzyma się jeśli spełniony będzie jakiś zadany warunek
- Intellisense przy zadawaniu warunku
- Is true, Is Changed

- Hit Count: sprawdza ile razy dany breakpoint został napotkany
- When hit: Pozwala wypisać wiadomość w output lub uruchomić makro
- Export/Import, format xml

Data Tip

- Podczas debugowania daje możliwość podglądania wartości zmiennych.
- Pozwala oglądanie nie tylko zmiennych typów prostych ale również złożonych obiektów
- Można "przypinać" sobie Data Tipy tak aby pracować na nich
- Można dodawać komentarze

Data Tip

- Można zmieniać wartości zmiennych podczas działania programu
- Import/Export, format xml

Watches

- Kolejny sposób na analizowanie wartości zmiennych
- Do 4 okien z watchami
- Okno Locals zawiera watche dla zmiennych lokalnych
- Okno Autos zawiera watche dla zmiennych ważnych z punktu widzenia obecnie wykonywanego polecenia

Watches

- Zakładać watch można albo klikając na kod, albo ręcznie wpisując wyrażenie
- Działa Intellisense
- Quick-Watch

Immediate Window

 Okno w którym w każdej chwili można wykonać kawałek kodu. W przypadku kiedy aplikacja zatrzymana jest w trakcie wykonywania można edytować zmienne w programie. Działa Intellisense.

Inne okienka

Call Stack:

Okienko pokazujące stos wywołań funkcji. Zawiera nazwę funkcji język, w którym została napisana oraz dodatkowe informacje między innymi nr linii.

- Registers: pokazuje zawartość rejestrów.
- Memory: pokazuje pamięć programu.

W porównaniu z tradycyjnymi debugerami debugowanie przy pomocy IntelliTrace, dostępne w programie Microsoft Visual Studio 2010 Ultimate, zapewnia ulepszony obraz aplikacji. Tradycyjne debugery pokazują stan aplikacji w czasie bieżącym oraz ograniczone informacje na temat zdarzeń, jakie miały miejsce w przeszłości. Należy zatem wywnioskować zdarzenia, które miały miejsce w przeszłości, na podstawie stanu aplikacji w czasie bieżącym, lub ponownie uruchomić aplikację, w celu odtworzenia minionych zdarzeń. Dzięki IntelliTrace można zobaczyć zdarzenia, które miały miejsce w przeszłości oraz kontekst ich wystąpienia. Redukuje to liczbę ponownych uruchomień, których wymaga się, w celu debugowania aplikacji oraz możliwość, że błąd nie zdoła się odtworzyć, gdy ponownie uruchomimy aplikację.

Przykłady zastosowań:

- Breakpointy
- Wyjątki
- Błędy nie powtarzalne

Zbieranie danych domyślnie włączone

- Zdarzenia debuggera
- Zdarzenia wyjątków
- Zdarzenia strukturowe

. 💭 💑 🖽 🖳 🖱 ▾ (ё ▾ (ё ་ (ё) ་ (ё)) 🕨 Debug 🛛 🖓 . Iransform	- I 🖓 🖀 🕼 22 XX 🛃 🖳 V T 🚽 🗷 🐁 A2 (E) 🚝 🚝 I
🗙 Disassembly ProcessMonitor.cs 🛍 SiteMonitor.cs 🛍 🛛 🗢	IntelliTrace 🔹 🕂 🗙
NsCommonThings.NsLogMana 👻 🔍 AddEntry(SzpiegMasterEventArgs logEntry) 🔷 👻	
<pre>string logFileNameSecondary = "logFileNameSecondary.xml"</pre>	(i) Switch to Calls View
)	All Categories 💽 All Threads 💽
<pre>public bool AddEntry(SzpiegMasterEventArgs logEntry)</pre>	Search 🔎
<pre>if (!File.Exists(logFileNameSecondary) logEntry==null) ret</pre>	File: Access D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\
	File: Close D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\t
StreamWriter sw;	File: Access D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\
try	File: Close D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\t
{	File: Access D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\
<pre>sw= File.AppendText(logFileNameSecondary); }</pre>	File: Access D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\
catch	File: Close D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\t
{	File: Close D:\Users\Zet\Documents\Visual Studio 2010\Projects_svn exchange\t
<pre>XmlTextWriter xmlWriter = new XmlTextWriter(sw); xmlWriter.Formatting = Formatting.None;</pre>	Gesture: Clicked "C" (Button)
	Gesture: Clicked "C" (Button)
	<i> </i>
	 File: Access logFileNameSecondary.xml Initialized a FileStream to the path logFileNameSecondary.xml Thread: Main Thread [4196]
	Related Views: Calls View, Locals, Call Stack

|: 💟 | 🏠 🏠 | 🖄 🔎 | 😢 💴 🔛 | 🎝 🔛 📰 🔜 🖏 | E LE 🕫 🖕

Di	sassen	nbly Modules Program.cs 🗎 🗙	•	IntelliTrace
	BukL	ao.Program 🗸 🗟 🖗 Main(string[] args)	•	i 🗄 🚰 🖆 🚯 🚳
			÷	(i) Switch to IntelliTrace Events View
		static void Metoda3()		Calls for thread Main Thread (5612)
		<pre>{ Console.WriteLine("Metoda3"); }</pre>		[System.Threading.ThreadHelper.ThreadStart()]
				BukLao.Program.Main(string[] args = {string[0]})
	11			Function Entry: BukLao.Program.Main
	[<pre>static void Main(string[] args) {</pre>		Beginning of Application: Main, Program.cs line 68
	₩ <u> </u>			[System.Console.WriteLine()]
	Ē.			🥪 📮 BukLao.Program.Metoda1()
	₹.	Metoda2();		📮 BukLao.Program.Metoda2()
	4	<pre>Metoda3(); Console.WriteLine("Po Metodzie"); Console.WriteLine(); Console.ReadLine(); string[] stringi = { "string1", "string2", "stri</pre>		📮 BukLao.Program.Metoda3()
				[System.Console.WriteLine()]
				[System.Console.WriteLine()]
				[System.Console.ReadLine()]
			n	
100) %	<pre></pre>	-	



Obsługiwane aplikacje i scenariusze debugowania

IntelliTrace obsługuje debugowanie aplikacji Visual Basic oraz C#, które korzystają z .NET v. 2.0, 3.0, 3.5, lub 4. Możliwe jest debugowanie większości aplikacji, w tym aplikacji utworzonych przy użyciu ASP.NET, Windows Forms, WPF, Windows Workflow oraz WCF. IntelliTrace nie obsługuje debugowania C++, skryptów, ani innych języków. Debugowanie aplikacji F# obsługiwane jest eksperymentalnie.



