Visual Studio 2010

DEVELOPMENTOR

DEVELOPING PEOPLE WHO DEVELOP SOFTWARE

Objectives

- Code Contracts
- Intellitrace
- Task Debugging
- Editor Enhancements

Early Bugs are Cheap Bugs

- **The earlier a bug is found the cheaper it is to fix**
  - Compiler syntax errors far cheaper to fix than bug discovered in user acceptance testing
  - Helping developer find bugs during development means better, cheaper development
- **.NET 4.0 ships with Code Contracts**
  - Compile and run time checking of pre and post conditions for methods and objects

Code Contracts

- **Code contracts have a number of benefits**
  - Improve unit testing
  - Allow static analysis of method and object requirements
  - Improve documentation
- **Code Contracts part of core framework**
  - Shipped in mscorlib
  - System.Diagnostics.Contracts namespace
  - Originally a Microsoft Research project called Spec#
  - Pivots around static class **Contract**

Pre-conditions

- **Preconditions allow developer to specify things that should be true before a method executes**
  - Normally relate to method parameters
  - Play same role as checking parameter and throwing exception
  - **Can specify exception to throw** on failure or default to ContractException

```
public void Hire(Person p)
{
  Contract.Requires<ArgumentNullException>(p != null);
  Contract.Requires(!IsEmployee(p));

  employees.Add(p);
}
```

Post Conditions

- **Specify things that should be true when method has finished**
  - **Can specify data which should not have changed**
  - Can specify properties of method return
  - Can specify conditions that should be met **if exception is thrown**

```
public void Hire(Person p)
{
  Contract.Ensures(Contract.OldValue(p) == p);
  Contract.EnsuresOnThrow<NullReferenceException>
      (Contract.OldValue(employees.Count) == employees.Count);

  employees.Add(p);
}
```

Invariants

- **Objects will often have rules that must always be true for their entire lifetime**
  - Contained references being non-null
  - Collections not being empty
  - Member variables being non-negative
- **Can create a method that details invariant rules and mark with `ContractInvariantMethod` attribute**
  - Name of method irrelevant

```
[ContractInvariantMethod]
private void Validate()
{
  Contract.Invariant(Owner != null);
  Contract.Invariant(employees != null);
  Contract.Invariant(employees.Count > 0);
}
```

Static Analysis

- **Performed in background**
  - Will be part of team system versions of VS2010
  - Checks call sites for code that would break contract
- **Indentified failures show up as warnings**
  - Always in pair – first is call site generating failure, second is contract assertion that fails

1	CodeContracts: requires is false	Program.cs	20	13
2	+ location related to previous warning	Program.cs	78	13

How do Code Contracts Work at Runtime?

- **Pre-Conditions contain code similar to standard parameter checking**
- **Post-Condition checks generated using IL-rewriting**
  - Currently not built into Visual Studio
  - Download from Devlabs
- **Invariant method injected into every method**
  - IL-rewriter inserts the call

Purity

- **Methods invoked in contract statements must be Pure**
  - Must not contain side-effects
  - Currently an honor system although MSR working on purity checker
  - Many framework methods marked as pure
  - Must **mark own methods as pure using attribute**

```
public void Hire(Person p)
{
  Contract.Requires(!IsEmployee(p));
  employees.Add(p);
}
```

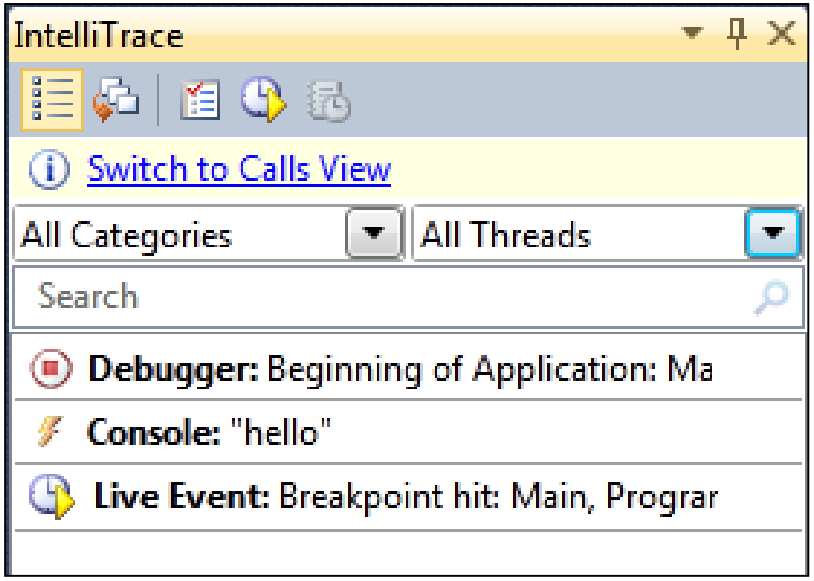
```
[Pure]
public bool IsEmployee(Person p)
{
  return employees.Contains(p);
}
```

Intellitrace

- **Introducing historical debugging of application**
  - Allows viewing of “interesting” things that have happened in the application
- **Two modes**
  - **Intellitrace Events Only** – records specific events that can be enabled and disabled. Default setting
  - **Intellitrace Events and call information** – collects richer information that allows stepping through the code historically but is more invasive

Interactive Mode

- **The intellitrace events and information can be viewed during interactive debugging session**
  - Debug -> Windows -> Intellitrace Events



Interactive Mode (Contd)

- **If call tracing is enabled can replay debugging session**
  - F10 /F11 walk forward Ctrl-Shift-F11 walks backwards
  - Edit and Continue is disabled when using call tracing

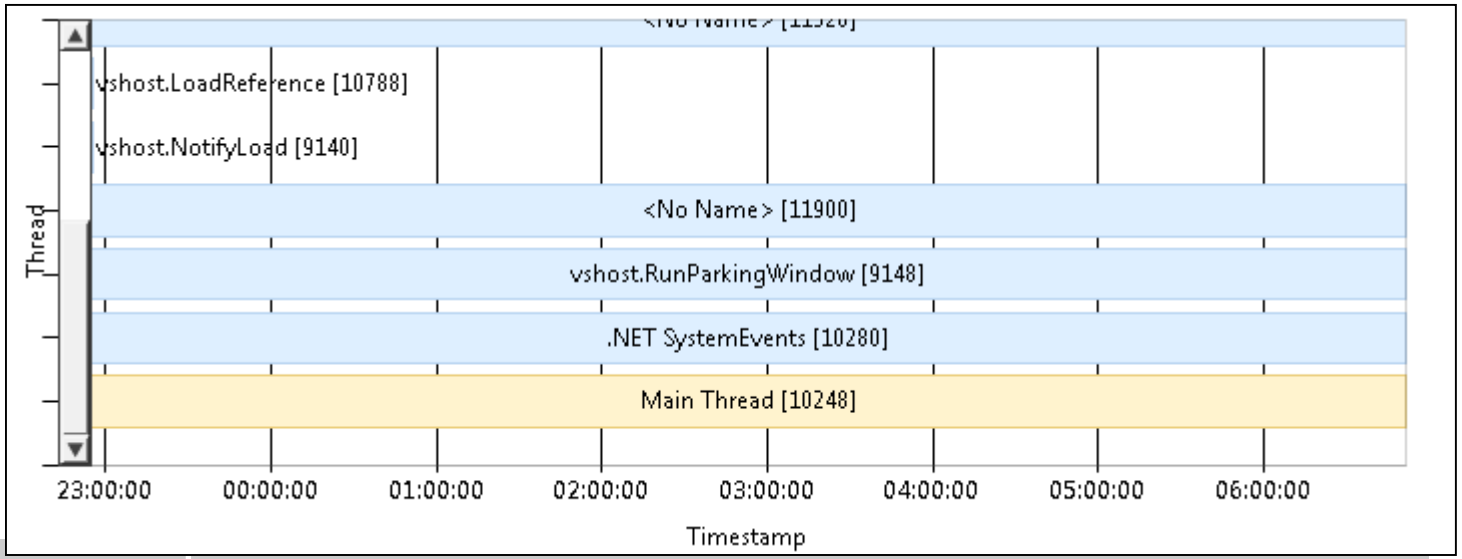
```
Console.WriteLine("hello");

Person rich = new Person { Name = "Rich", Age = 44 };
Company c = new Company(rich);

Person andy = new Person { Name = "Andy", Age = 37 };
c.Hire(andy);
c.Fire(andy);
```

Historical Mode

- **Intellitrace data is saved to file**
  - Location configurable on advanced tab of Intellitrace settings
  - Files have .itrace extension
- **After debugging session has ended can load the file into VS2010**
  - Double click on thread of interest to replay program execution (how much depends on level of tracing)



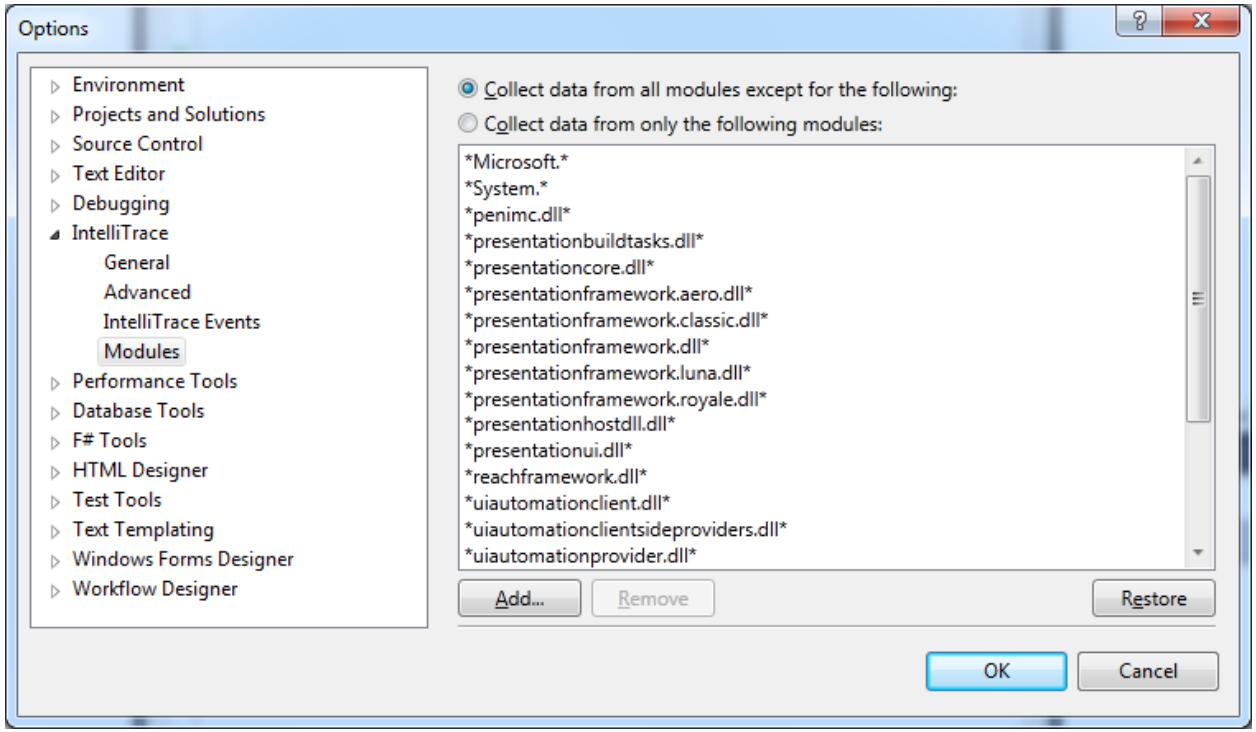
Collecting Data Outside of VS2010

- **Command line tool for collecting trace data**
  - Intellitrace.exe
  - Provide XML file to define trace profile (Collection Plan)
- **Can launch executable independently and then load log file into VS2010**

```
C:\"\Program Files\Microsoft Visual Studio 10.0\Team
Tools\TraceDebugger Tools\IntelliTrace.exe" launch
/f:TestRun.itrace /cp:CollectionPlan.xml Tasks.exe
```

Tuning the Data Collection

- **Maybe want to exclude some modules from data collection**
  - Third party code
  - Fully tested modules
  - May want to focus exclusively on a set of modules



Task Debugging

- **Two new debugging windows for working with tasks**
  - Parallel Tasks
  - Parallel Stacks
- **Parallel Tasks**
  - Task orientated view of work
  - Shows status of task: Scheduled, Running, Waiting, Waiting-Deadlocked
- **Parallel Stacks**
  - Shows call stacks and relationship between running threads
  - Can show from call tack or method perspective

Parallel Tasks

- **Task version of Thread Window**
  - Different view to thread window as it shows tasks that are not yet running

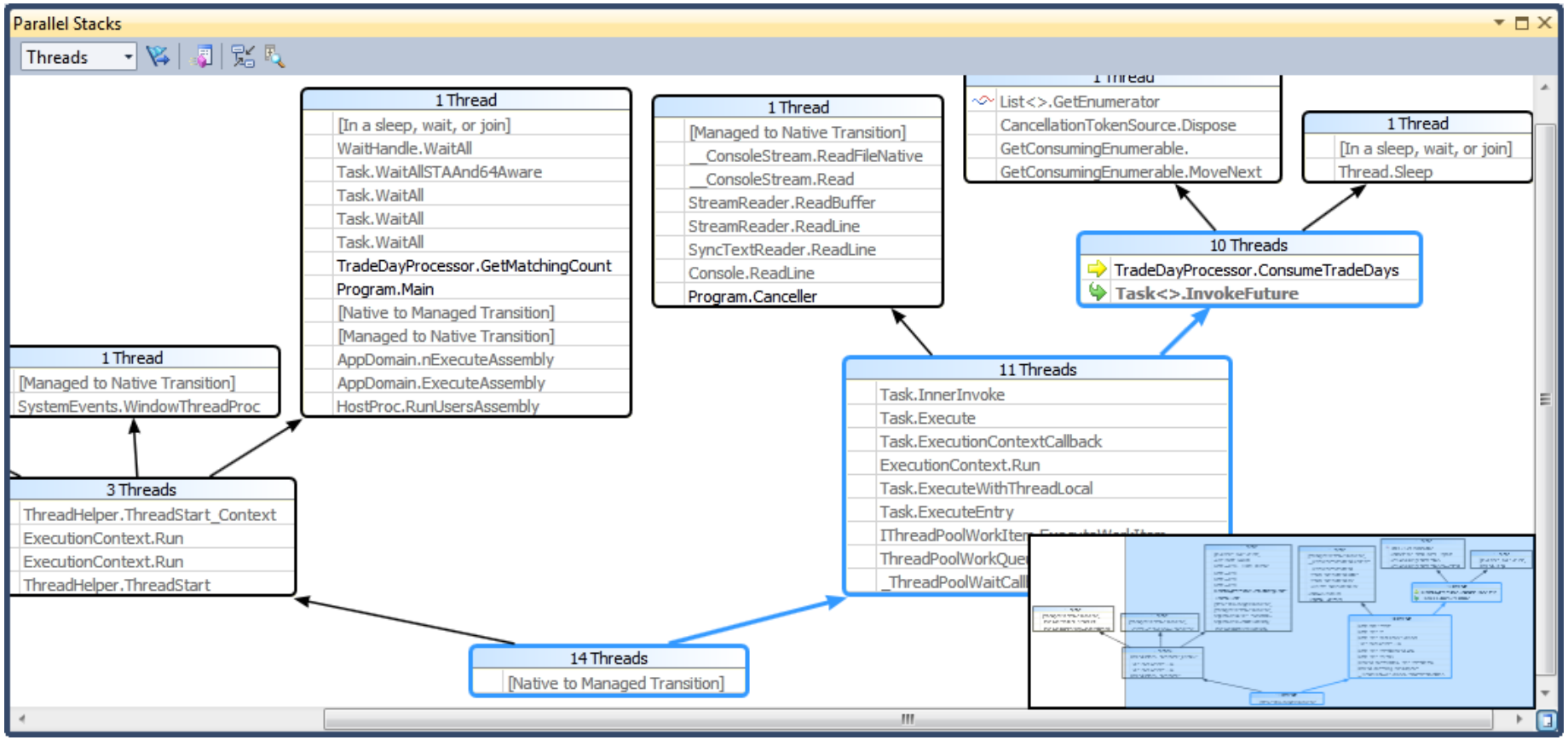
Parallel Tasks

ID	Status	Location	Task	Thread Assignment	AppDomain
1	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	10052 (Worker Threa	1 (Tasks.vshost.exe)
3	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	11340 (Worker Threa	1 (Tasks.vshost.exe)
4	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	11080 (Worker Threa	1 (Tasks.vshost.exe)
5	Waiting	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	9348 (Worker Thread	1 (Tasks.vshost.exe)
6	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	4112 (Worker Thread	1 (Tasks.vshost.exe)
7	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	8092 (Worker Thread	1 (Tasks.vshost.exe)
8	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	11600 (Worker Threa	1 (Tasks.vshost.exe)
9	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	12136 (Worker Threa	1 (Tasks.vshost.exe)
10	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	6204 (Worker Thread	1 (Tasks.vshost.exe)
11	Running	Tasks.TradeDayProcessor.ConsumeTradeDays	ConsumeTradeI	11752 (Worker Threa	1 (Tasks.vshost.exe)
12	Running	Tasks.Program.Canceller	Canceller({Syst	3204 (Worker Thread	1 (Tasks.vshost.exe)

Locals | Autos | Watch 1 | Threads | Parallel Tasks

Parallel Stacks

- **Shows relationships between running tasks**

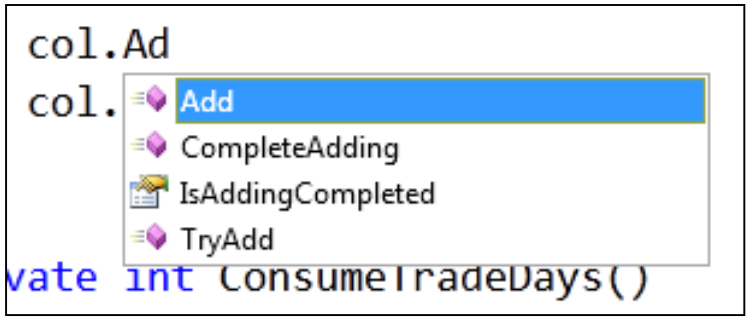


Editor Improvements

- **Editor has been rewritten**
  - Now written in WPF
  - Improved Intellisense
  - Better Multi-monitor support
- **WPF Editor**
  - Zoom everywhere
  - Inline data and code visualization

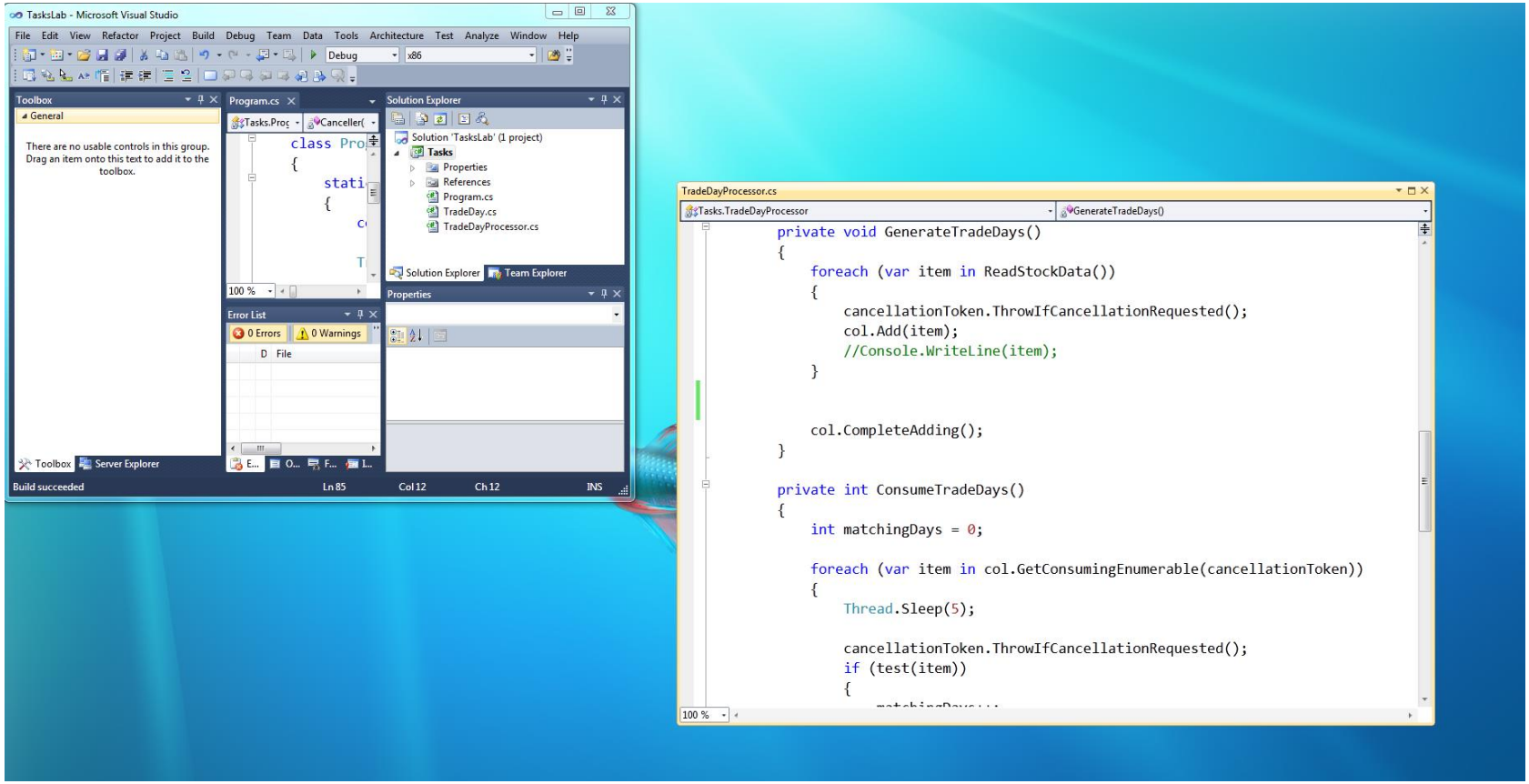
Improved Intellisense

- **Intellisense now filters**
  - Only shows options matching code being typed
  - Uses intelligent matching
  - Can even use Pascal casing initials



Multi-monitor Support

- **Code Windows can now be undocked/detached from Visual Studio**



Summary

- **Code Contracts can improve code quality**
- **Improved Debugger support**
  - IntelliTrace
  - Task Windows
- **Editor Enhancements**