

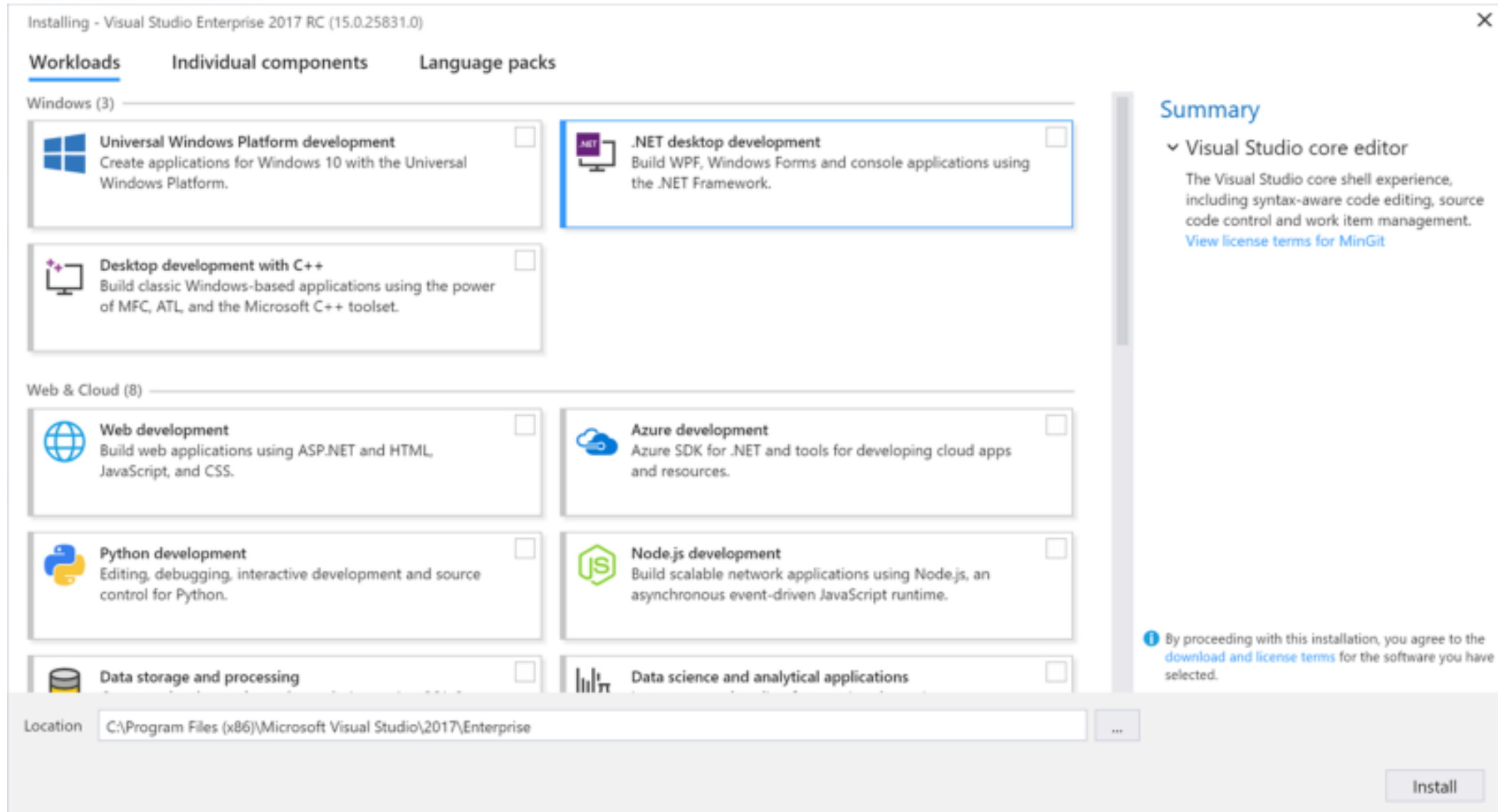
# VS2017 and C#7.0

Crystal Tenn  
Crystal.Tenn@microsoft.com

# What's New in Visual Studio 2017



# New setup experience



# Visual Studio 2017

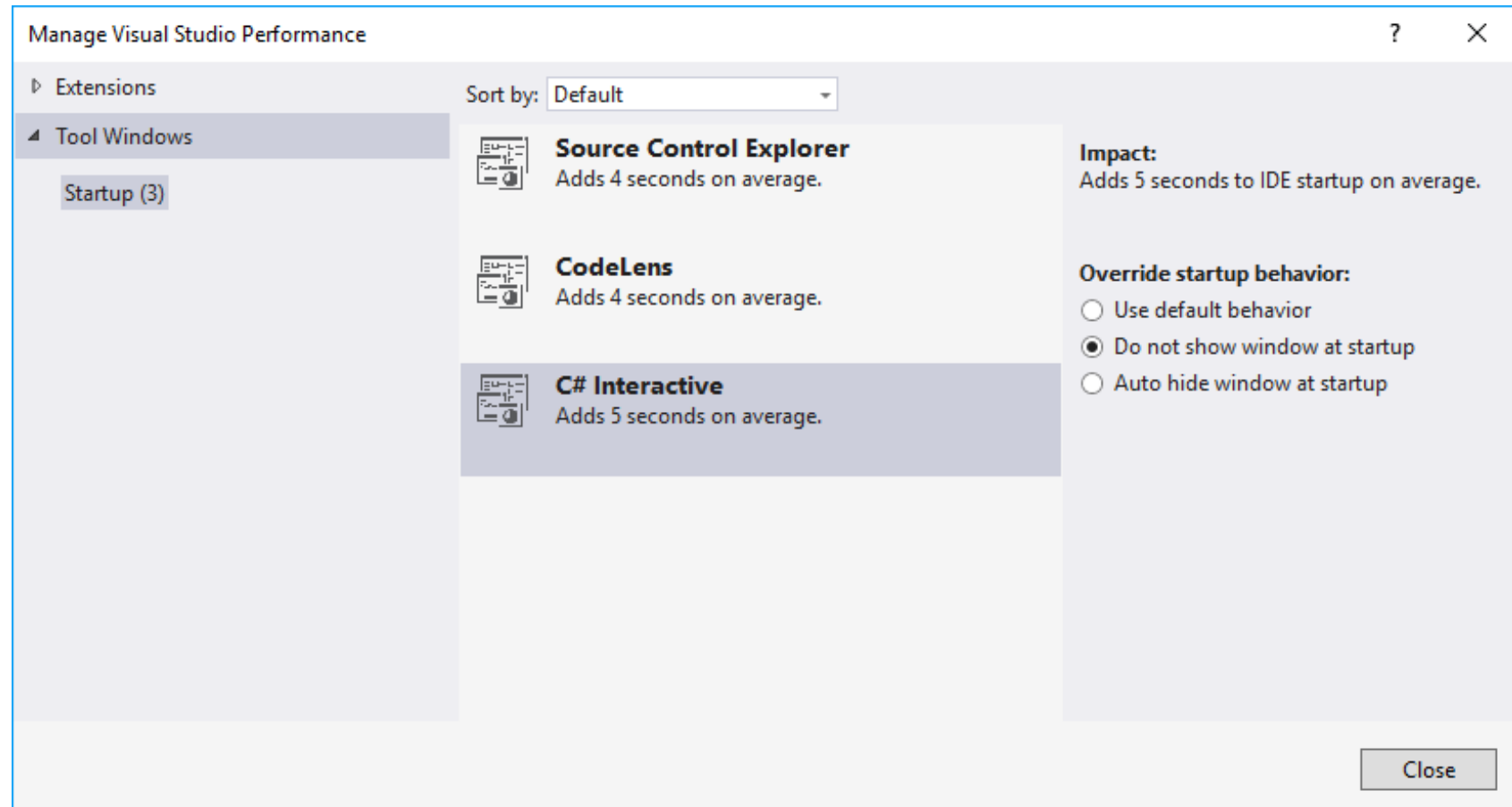
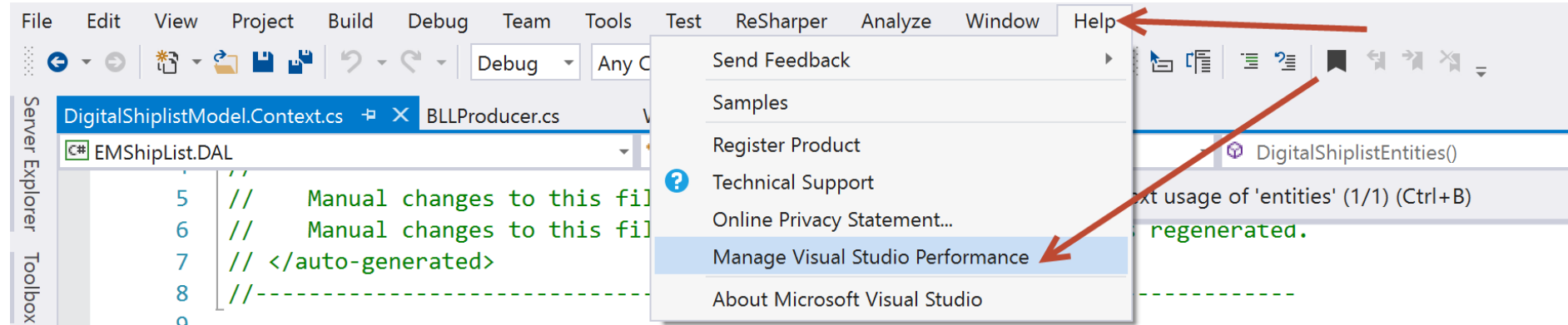
- Faster Startup
- Faster Solution load
- Faster builds
- Less memory



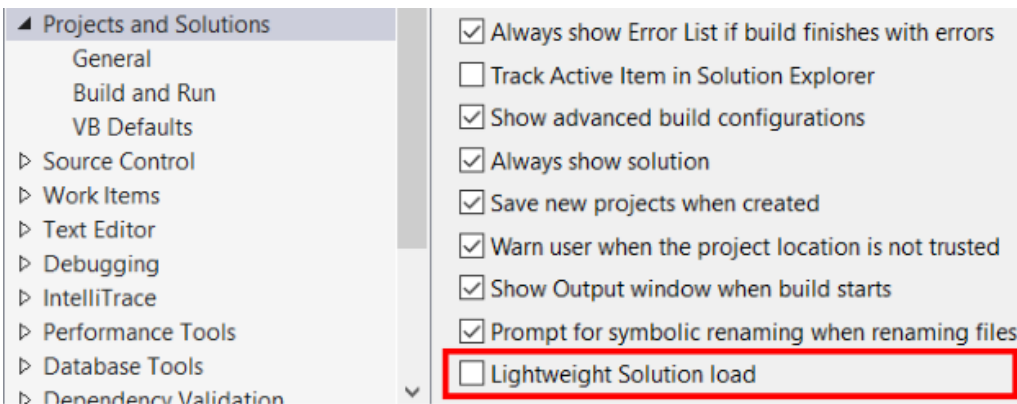
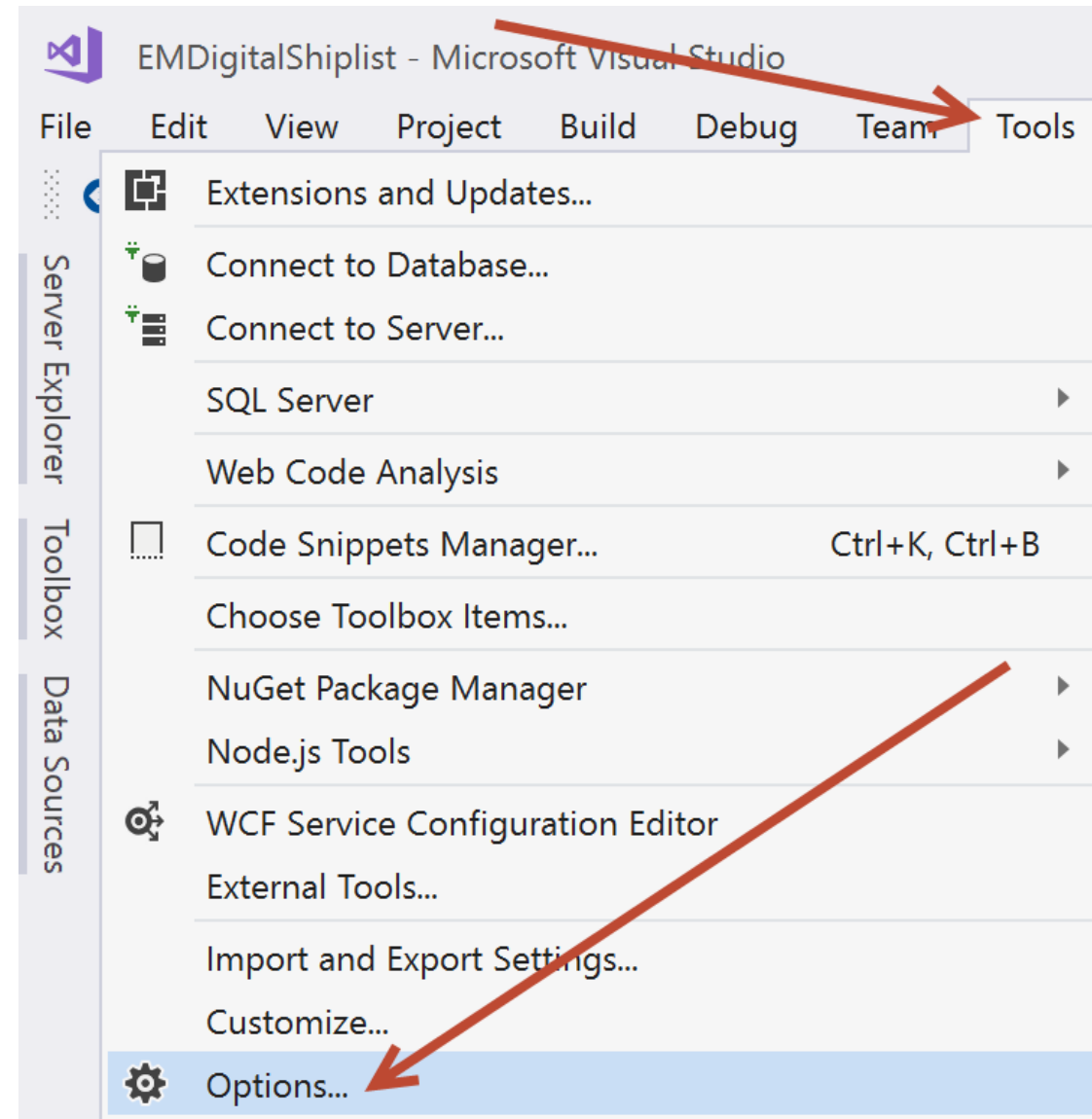
# Visual Studio Performance Center

- Can help you optimize your IDE start-up time.
- The Performance Center lists all the extensions and tool windows that might slow down the IDE startup.
  - You can use it to improve startup performance by determining when extensions start, or whether tool windows are open at startup.

# Perf improvements



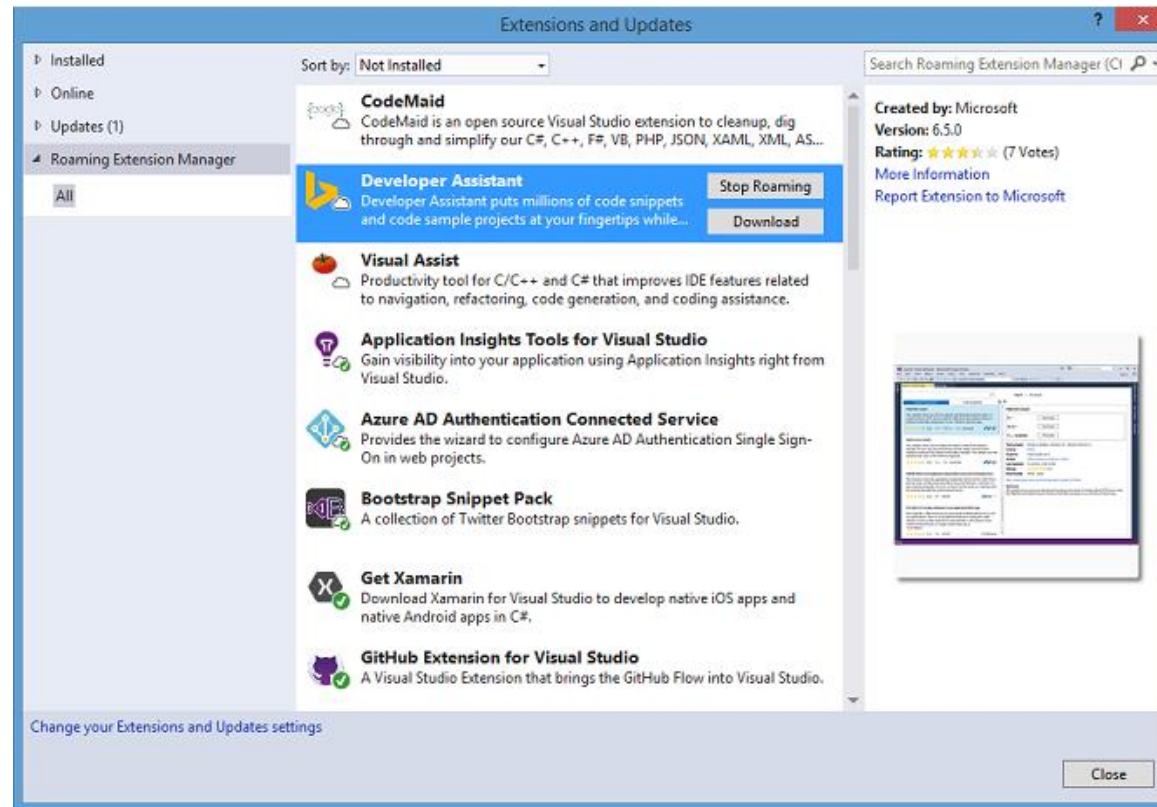
# Perf improvements





# Productivity

- Sign in across multiple accounts
- Manage your extensions with Roaming Extensions Manager



# Structure Visualizer

- Vertical guidelines
  - Hover to see source of container
- Nested code visualization
- C#, VB and XAML (and others)

```
namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello world");
        }
    }
}
```

Sources

16

0 references | Crystal Tenn, 14 days ago | 1 author, 1 cf

```
public class AutoCompleteLocation : System.Web.Services.WebService
{
    public AutoCompleteLocation ()
    {
    }
}
```

19

//Uncomment the following line

20

//InitializeComponent();

21

22

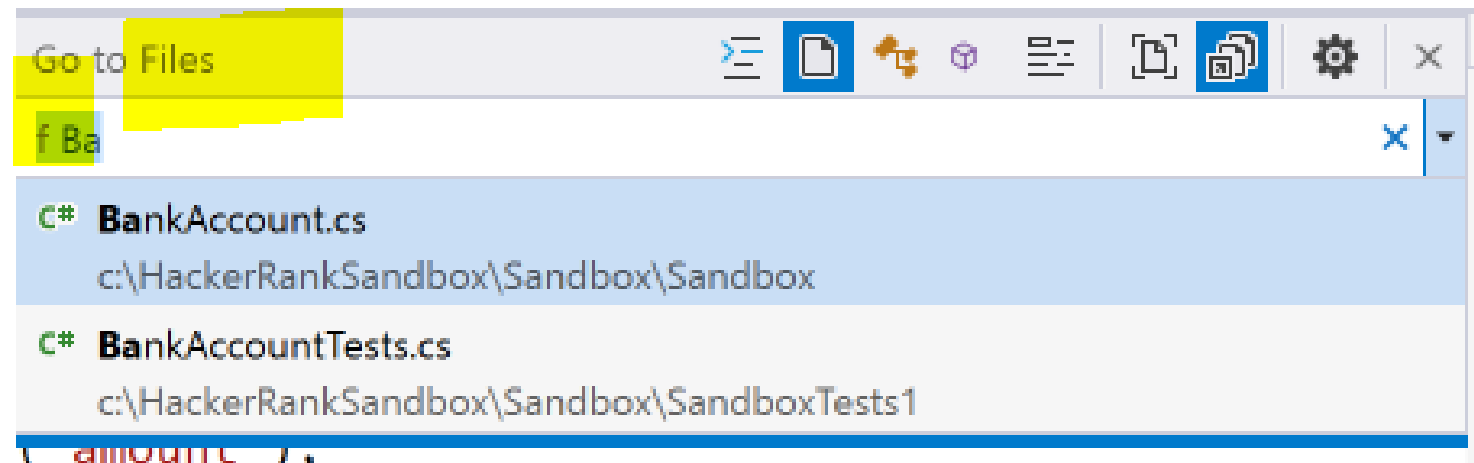
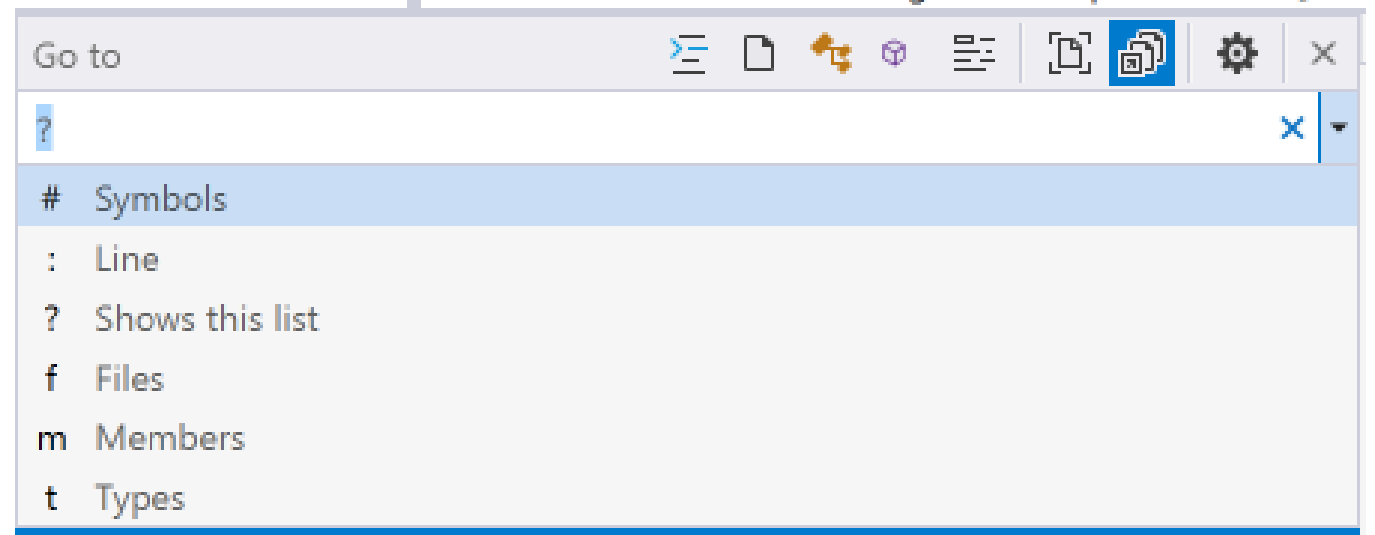
}

# Visual Studio 2017 Features

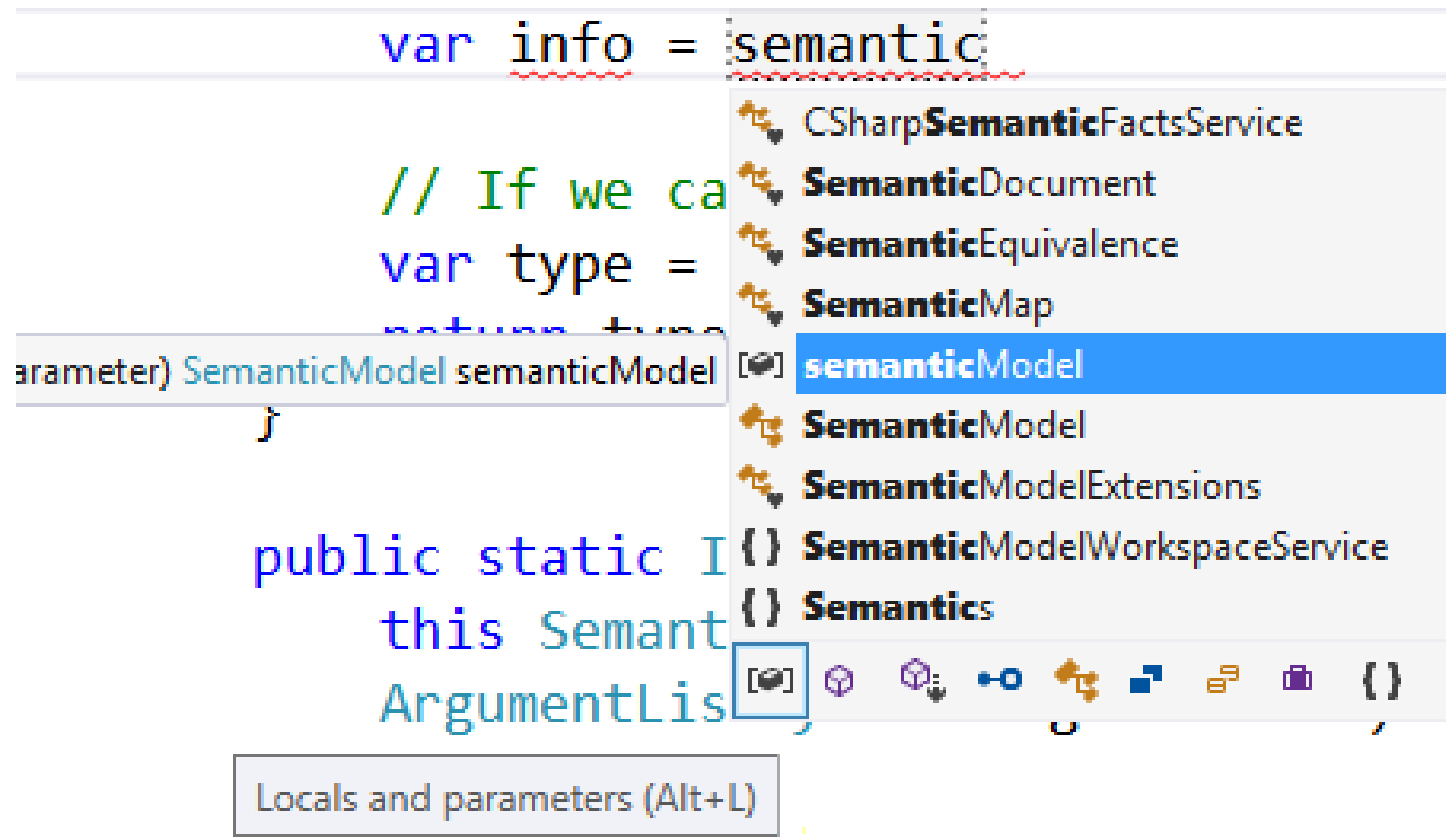
- Navigation
- IntelliSense
- Run to Click

# Go To (combined with old “Navigate to”)

- **Ctrl+,** or **Ctrl+T**
- **Filters**
  - # = Symbol
  - F = file name
  - M = members
  - T = type
  - := line number (Ctrl-g)

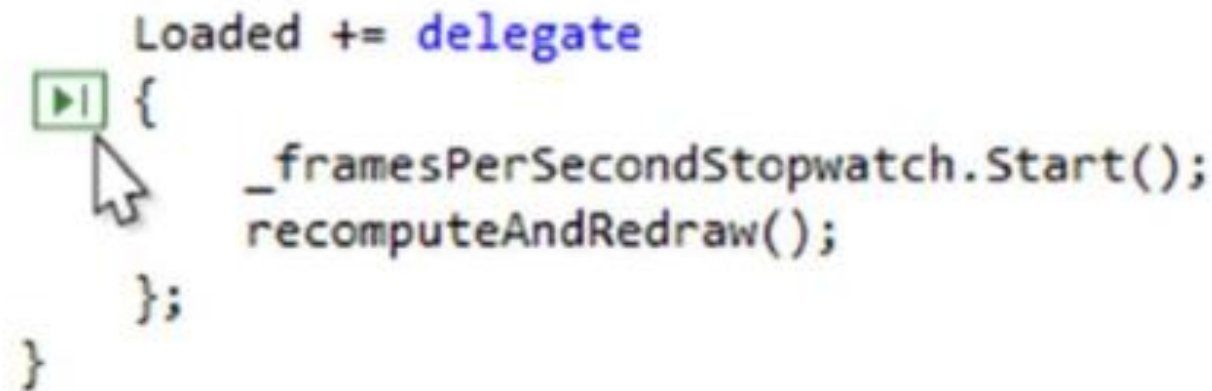


# Intellisense filters



# Run to click

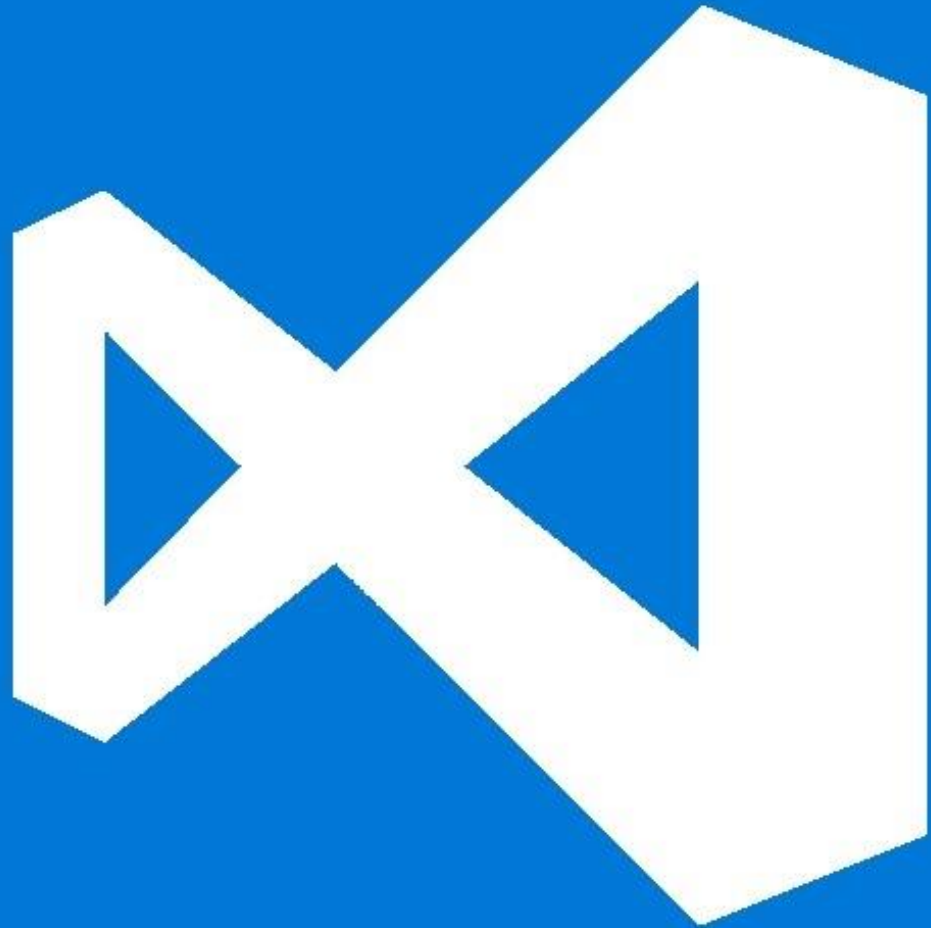
- Easily skip ahead during debugging without setting a breakpoint to stop on the line you want.
- When you are stopped in the debugger, simply click the icon that appears next to the line of code that your mouse is over.
- Your code will run and stop on that line the next time it is hit in your code path.



The image shows a snippet of code in a debugger. The code is as follows:

```
Loaded += delegate  
{  
    _framesPerSecondStopwatch.Start();  
    recomputeAndRedraw();  
};  
}
```

A green square icon with a white right-pointing triangle and a vertical bar (the 'Run to click' icon) is positioned to the left of the opening curly brace of the first block. A mouse cursor is hovering over this icon.



# Visual Studio 2017

Demo on navigation, IntelliSense, and Run to Click

# Visual Studio 2017 Demo Summary

- Navigation
  - All the navigation is together under the Edit menu now
  - Improved Navigate To which is now called Go To All
    - CTRL + T or CTRL + ,
    - VS opens files in temporary viewer as you scroll through search results
- IntelliSense
  - Filters have been added
  - Smarter about finding Capital letters 'SV' would find 'Set Value' method for example
  - Most likely rather than simply a top property
  - Find All References has grouping, colors, and a peek preview
- Run to Click
  - Click green glyph on any lines of code below a breakpoint to go there
  - Perf tip glyph tells how long execution took and can open performance diagnostics tools window

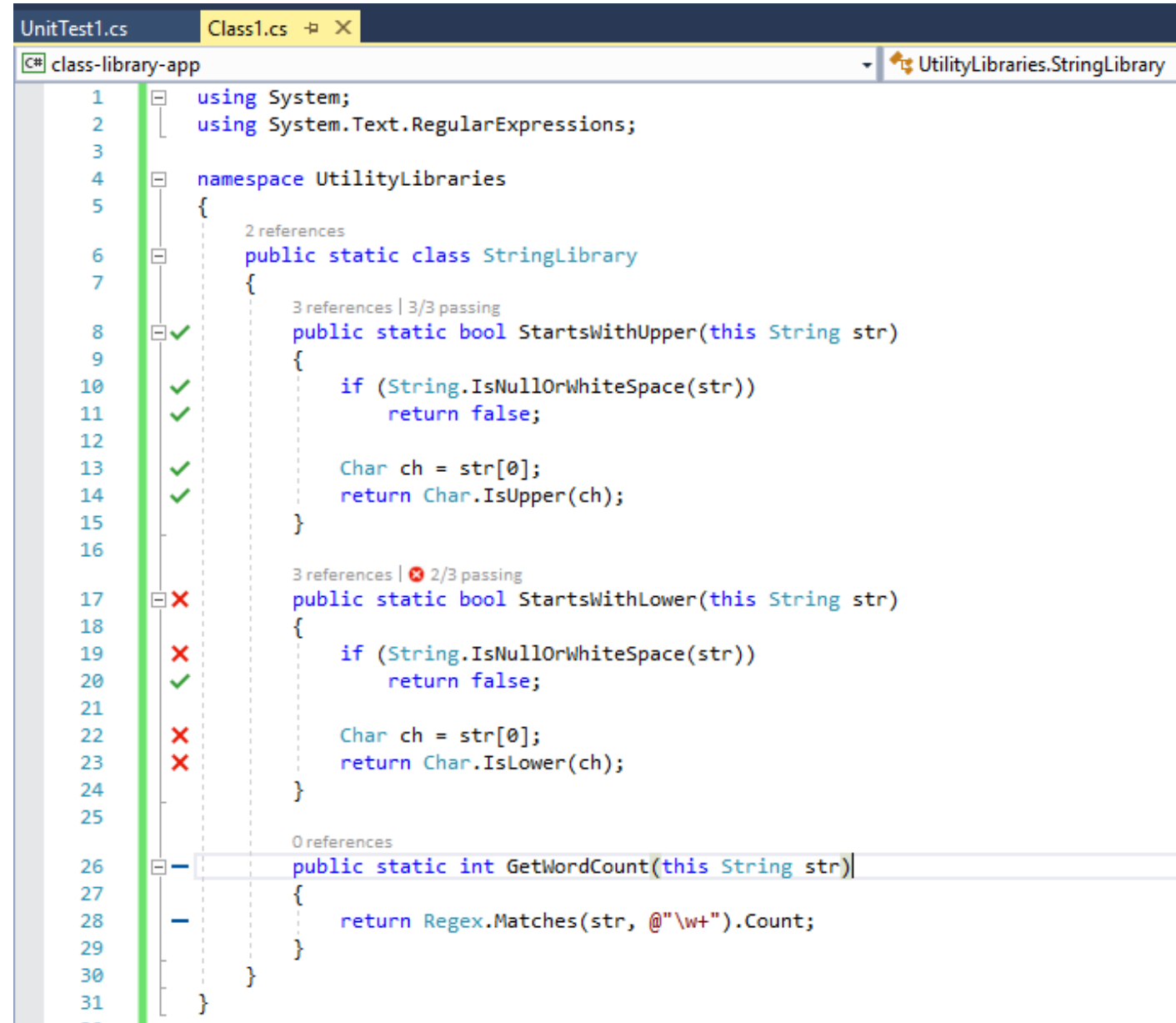


# Visual Studio 2017 Features

- Live Code Analysis
- Real-Time architecture validation
- Live Unit Testing

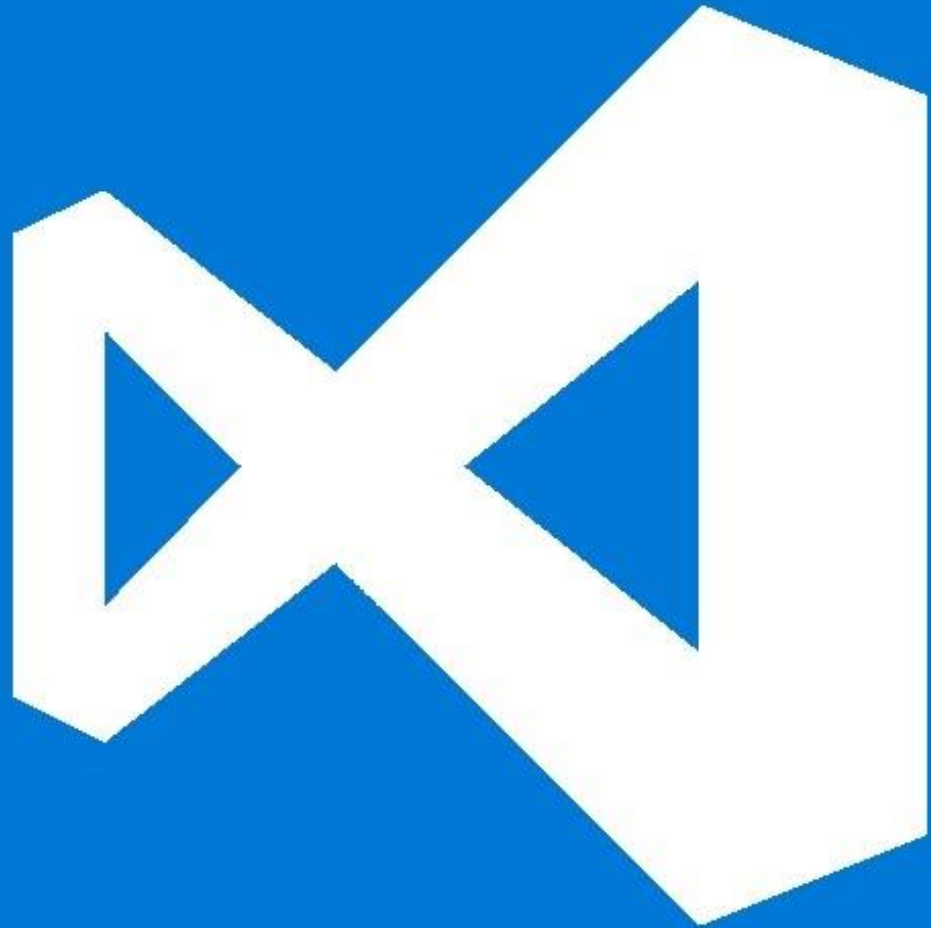
# Live unit testing

- Only in Enterprise edition
- Visualizes test results on the editor during coding
- C#/VB.NET w/ MSTest, xUnit, NUnit



The screenshot shows a Visual Studio IDE with two tabs: 'UnitTest1.cs' and 'Class1.cs'. The 'Class1.cs' tab is active, displaying the source code for a class library named 'UtilityLibraries.StringLibrary'. The code includes two static methods: 'StartsWithUpper' and 'StartsWithLower'. To the left of the code, a vertical green bar indicates the test results for each line of code. The 'StartsWithUpper' method (lines 8-15) is marked with green checkmarks, indicating it is passing. The 'StartsWithLower' method (lines 17-24) is marked with red X's, indicating it is failing. The 'GetWordCount' method (lines 26-29) is marked with a blue minus sign, indicating it is not yet tested. The status bar at the bottom shows '0 references' for the 'GetWordCount' method.

```
1  using System;
2  using System.Text.RegularExpressions;
3
4  namespace UtilityLibraries
5  {
6      2 references
7      public static class StringLibrary
8      {
9          3 references | 3/3 passing
10         public static bool StartsWithUpper(this String str)
11         {
12             if (String.IsNullOrEmpty(str))
13                 return false;
14             Char ch = str[0];
15             return Char.IsUpper(ch);
16         }
17         3 references | 2/3 passing
18         public static bool StartsWithLower(this String str)
19         {
20             if (String.IsNullOrEmpty(str))
21                 return false;
22             Char ch = str[0];
23             return Char.IsLower(ch);
24         }
25     }
26     0 references
27     public static int GetWordCount(this String str)
28     {
29         return Regex.Matches(str, @"\w+").Count;
30     }
31 }
```



# Visual Studio 2017

Demo on Live Unit Testing

# Visual Studio 2017 Demo Summary

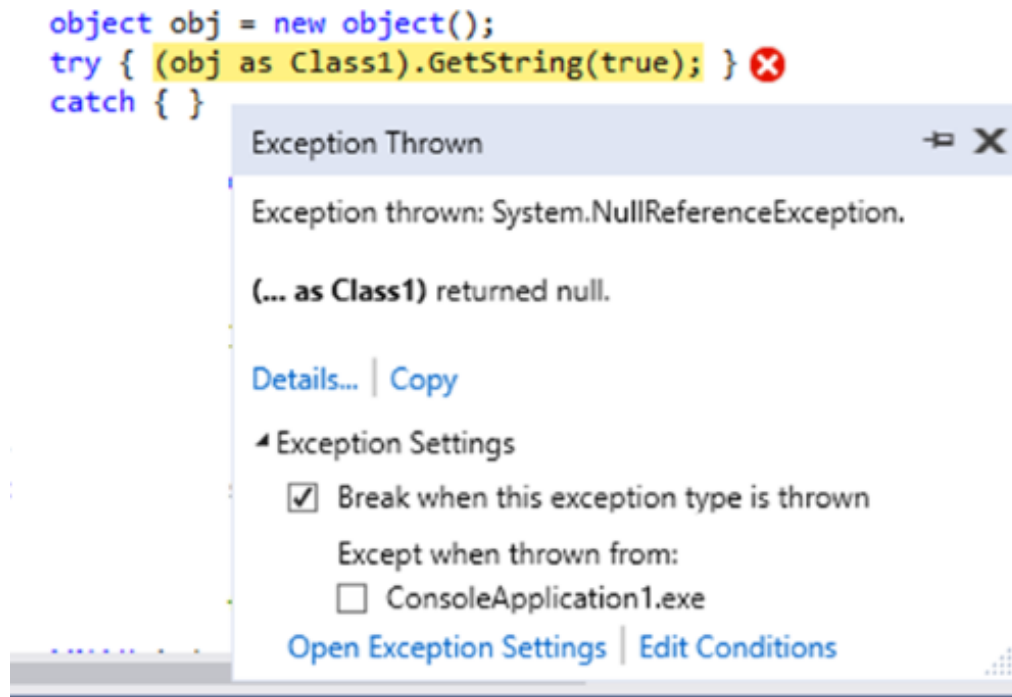
- Live Unit Testing
  - Tests > Live Unit Testing > Start

# VS2017 Streamlined Cloud Development

- Docker
- DevOps
- .NET Core and Linux

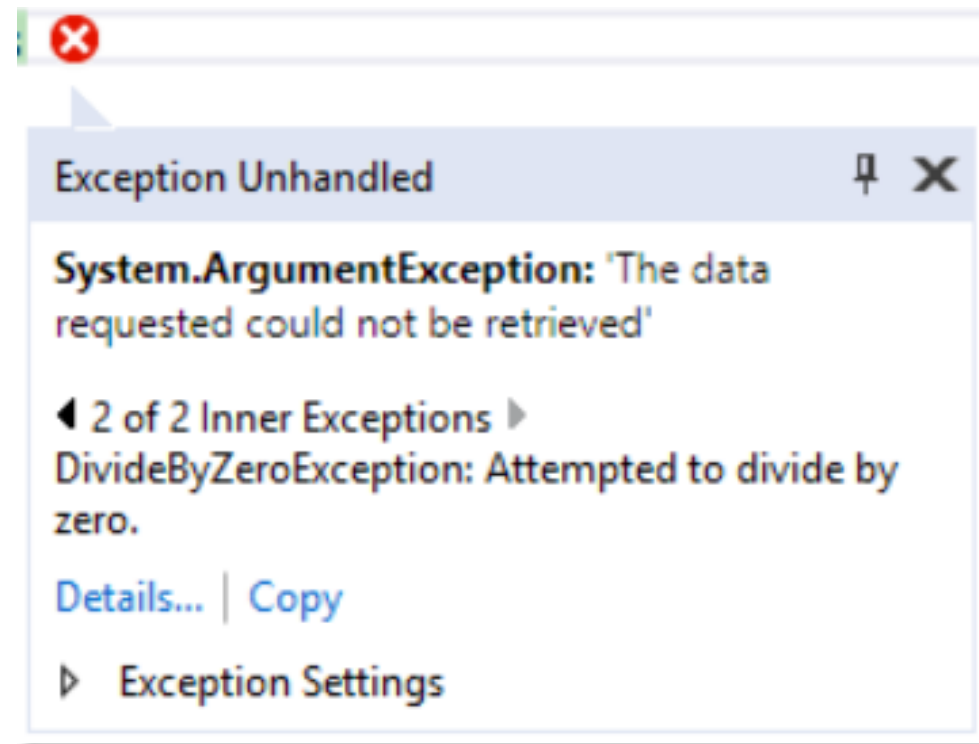
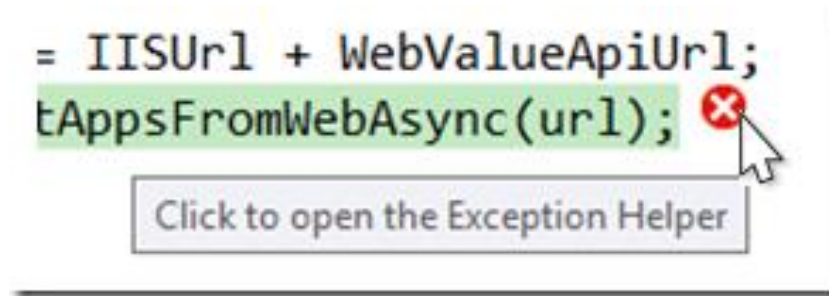
# New exception helper

- Breaking on the Line of Code – No matter if you are managed or native debugging, when you break on an exception the debugger will stop you on the line of code where that exception happens.
- Exception Information at a Glance – You will instantly be able to read the exception type and exception message in the Exception Helper, and whether the exception was thrown or unhandled.



# New exception helper

- Inner Exceptions right away – No longer will you have to drill into the exception details to see if there is an inner exception. We show it to you right when you break with the ability to navigate back through multiple inner exceptions.



# Redgate Data Tools

- To extend DevOps capabilities to SQL Server database development, Redgate Data Tools are now available in the following editions of Visual Studio 2017.
- Included with Visual Studio 2017 Enterprise:
  - [Redgate ReadyRoll Core](#)
  - [Redgate SQL Prompt Core](#)
- Included with all editions of Visual Studio 2017:
  - [Redgate SQL Search](#)



# C# 7 Features



# Agenda

- Simpler out variables
- Tuples
- Pattern Matching
- ref locals and returns
- Local Functions
- More expression-bodied members
- throw Expressions
- Generalized async return types
- Numeric literal syntax improvements

## Old out parameters

C#

 Copy




```
int numericResult;  
if (int.TryParse(input, out numericResult))  
    WriteLine(numericResult);  
else  
    WriteLine("Could not parse input");
```

## C# 7.0 out parameters

C#

 Copy

```
if (int.TryParse(input, out int result))  
    WriteLine(result);  
else  
    WriteLine("Could not parse input");
```

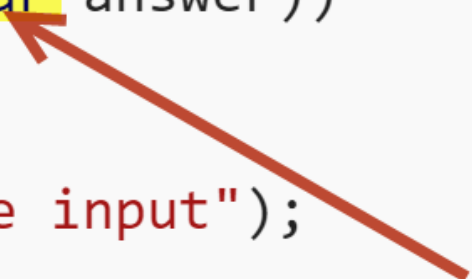


## C# 7.0 out parameters

C#

 Copy

```
if (int.TryParse(input, out var answer))  
    WriteLine(answer);  
else  
    WriteLine("Could not parse input");
```




## C# 7.0 out parameters scope 'leak'

C#

 Copy

```
if (!int.TryParse(input, out int result))  
{  
    return null;  
}  
  
return result;
```



# Old tuples

C#

VB

```
// Create a 7-tuple.  
var population = new Tuple<string, int, int, int, int, int, int>(      
    "New York", 7891957, 7781984,      
    7894862, 7071639, 7322564, 8008278);  
  
// Display the first and last elements.  
Console.WriteLine("Population of {0} in 2000: {1:N0}",      
    population.Item1, population.Item7);  
  
// The example displays the following output:  
//      Population of New York in 2000: 8,008,278
```

## C# 7.0 tuples

C#

 Copy

```
var letters = ("a", "b");
```

C#

 Copy

```
(string Alpha, string Beta) namedLetters = ("a", "b");
```

C#

 Copy

```
var alphabetStart = (Alpha: "a", Beta: "b");
```



## C# 7.0 tuples

C#

 Copy

```
(string First, string Second) firstLetters = (Alpha: "a", Beta: "b");
```

## C# 7.0 Tuples Min and Max

C#

 Copy

```
private static (int Max, int Min) Range(IEnumerable<int> numbers)
{
    int min = int.MaxValue;
    int max = int.MinValue;
    foreach(var n in numbers)
    {
        min = (n < min) ? n : min;
        max = (n > max) ? n : max;
    }
    return (max, min);
}
```

## C# 7.0 Tuples Min and Max

C#

 Copy

```
var range = Range(numbers);
```

C#

 Copy

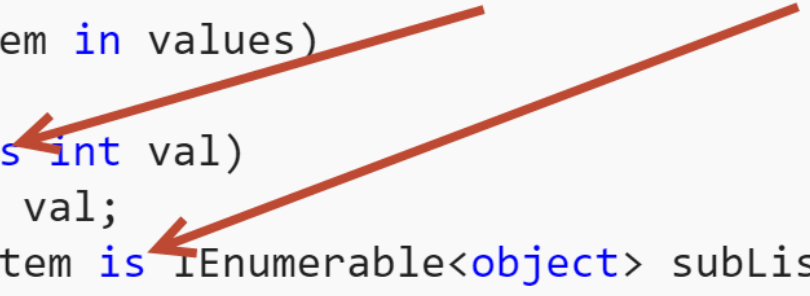
```
(int max, int min) = Range(numbers);
```

## C# 7.0 Pattern Matching with 'is'

C#

 Copy

```
public static int DiceSum2(IEnumerable<object> values)
{
    var sum = 0;
    foreach(var item in values)
    {
        if (item is int val)
            sum += val;
        else if (item is IEnumerable<object> subList)
            sum += DiceSum2(subList);
    }
    return sum;
}
```



# C# 7.0 Pattern Matching with 'switch'

C#

Copy

```
public static int DiceSum4(IEnumerable<object> values)
{
    var sum = 0;
    foreach (var item in values)
    {
        switch (item)
        {
            case 0:
                break;
            case int val:
                sum += val;
                break;
            case IEnumerable<object> subList when subList.Any():
                sum += DiceSum4(subList);
                break;
            case IEnumerable<object> subList:
                break;
            case null:
                break;
            default:
                throw new InvalidOperationException("unknown item type");
        }
    }
    return sum;
}
```

## C# 7.0 ref locals and returns

```
public ref int Find(int number, int[] numbers)
{
    for (int i = 0; i < numbers.Length; i++)
    {
        if (numbers[i] == number)
        {
            return ref numbers[i]; // return the storage location, not the value
        }
    }
    throw new IndexOutOfRangeException($"{nameof(number)} not found");
}

int[] array = { 1, 15, -39, 0, 7, 14, -12 };
ref int place = ref Find(7, array); // aliases 7's place in the array
place = 9; // replaces 7 with 9 in the array
WriteLine(array[4]); // prints 9
```

# C# 7.0 Local Functions - Iterative

C#

```
public static IEnumerable<char> AlphabetSubset3(char start, char end)
{
    if (start < 'a' || start > 'z')
        throw new ArgumentOutOfRangeException(paramName: nameof(start), message: "start must be a letter");
    if (end < 'a' || end > 'z')
        throw new ArgumentOutOfRangeException(paramName: nameof(end), message: "end must be a letter");

    if (end <= start)
        throw new ArgumentException($"{nameof(end)} must be greater than {nameof(start)}");

    return alphabetSubsetImplementation();

    IEnumerable<char> alphabetSubsetImplementation()
    {
        for (var c = start; c < end; c++)
            yield return c;
    }
}
```

# C# 7.0 Local Functions - Async

C#

 Copy

```
public Task<string> PerformLongRunningWork(string address, int index, string name)
{
    if (string.IsNullOrEmpty(address))
        throw new ArgumentException(message: "An address is required", paramName: nameof(address));
    if (index < 0)
        throw new ArgumentOutOfRangeException(paramName: nameof(index), message: "The index must be non-negative");
    if (string.IsNullOrEmpty(name))
        throw new ArgumentException(message: "You must supply a name", paramName: nameof(name));

    return longRunningWorkImplementation();

    async Task<string> longRunningWorkImplementation()
    {
        var interimResult = await FirstWork(address);
        var secondResult = await SecondStep(index, name);
        return $"The results are {interimResult} and {secondResult}. Enjoy.";
    }
}
```



## Old expression-bodied function members

C#

 Copy

```
public override string ToString() => $"{LastName}, {FirstName}";
```

C#

 Copy

```
public string FullName => $"{FirstName} {LastName}";
```

# C# 7.0 More expression-bodied members

C#

 Copy

```
// Expression-bodied constructor
public ExpressionMembersExample(string label) => this.Label = label;

// Expression-bodied finalizer
~ExpressionMembersExample() => Console.Error.WriteLine("Finalized!");

private string label;

// Expression-bodied get / set accessors.
public string Label
{
    get => label;
    set => this.label = value ?? "Default label";
}
```

## C# 7.0 Shorthand for expression-bodied members

```
// C#6 Destructor
```

```
~Person()
```

```
{
```


```
    Console.WriteLine("Destructor was called!");
```

```
}
```

```
~Person() => Console.Error.WriteLine("C# 7 Destructor was called!");
```

# C# 7.0 Throw expressions

C#

 Copy

```
public string Name
{
    get => name;
    set => name = value ??
        throw new ArgumentNullException(paramName: nameof(value), message: "New name must not be null");
}
```

# C# 7.0 Throw Expressions

```
class Person
{
    public string Name { get; }
    public Person(string name) => Name = name ?? throw new ArgumentNullException(name);
    public string GetFirstName()
    {
        var parts = Name.Split(" ");
        return (parts.Length > 0) ? parts[0] : throw new InvalidOperationException("No name!");
    }
    public string GetLastName() => throw new NotImplementedException();
}
```

## C# 7.0 Generalized async return types

C#

```
public ValueTask<int> CachedFunc()
{
    return (cache) ? new ValueTask<int>(cacheResult) : new ValueTask<int>(LoadCache());
}
private bool cache = false;
private int cacheResult;
private async Task<int> LoadCache()
{
    // simulate async work:
    await Task.Delay(100);
    cacheResult = 100;
    cache = true;
    return cacheResult;
}
```

# C# 7.0 Numeric literal syntax improvements

```
public const int One = 0b0001;  
public const int Two = 0b0010;  
public const int Four = 0b0100;  
public const int Eight = 0b1000;
```

Binary literal

```
public const int Sixteen = 0b0001_0000;  
public const int ThirtyTwo = 0b0010_0000;  
public const int SixtyFour = 0b0100_0000;  
public const int OneHundredTwentyEight = 0b1000_0000;
```

Digit separator

```
public const long BillionsAndBillions = 100_000_000_000;
```

```
public const double AvogadroConstant = 6.022_140_857_747_474e23;  
public const decimal GoldenRatio =  
1.618_033_988_749_894_848_204_586_834_365_638_117_720_309_179M;
```

# Things that didn't make the cut (but may later)

- Non nullable reference types
  - Proposal here: <https://gist.github.com/olmobrutall/31d2abafe0b21b017d56>
- Pattern matching and Record types
  - <https://github.com/dotnet/roslyn/blob/features/records/docs/features/records.md>
- Advanced pattern matching scenarios
  - <https://github.com/dotnet/roslyn/blob/features/patterns/docs/features/patterns.md>

<https://github.com/dotnet/roslyn/issues> + “Feature Request”



# Useful Links

- Microsoft docs: <https://docs.microsoft.com/en-us/>
- C# Get started guide: <https://www.microsoft.com/net/tutorials/csharp/getting-started/hello-world>
- New in C#: <https://docs.microsoft.com/en-us/dotnet/articles/csharp/whats-new/csharp-7>
- New in VS: <https://docs.microsoft.com/en-us/visualstudio/ide/whats-new-in-visual-studio>



Thank you!