Navigating Visual Studio

This document contains a potpourri for navigating along with useful tips for working with code in no special order.

# Keyboard

Visual Studio is an IDE with immense functionality and with that comes some getting used to in regards navigation.

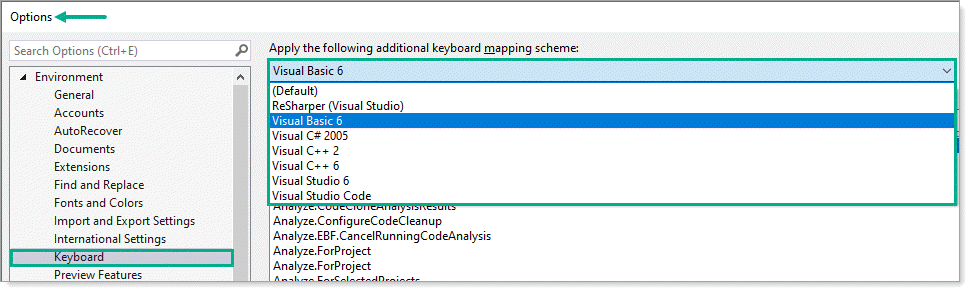
When Visual Studio starts for the first time there is a prompt to select keyboard mappings. This can be changed later by using from the IDE menu Tools → options → Keyboard.

In the screenshot below shows Karen’s setup which she selected back from Visual Basic days and has carried it forward which is merged in with a hybrid of ReSharper.

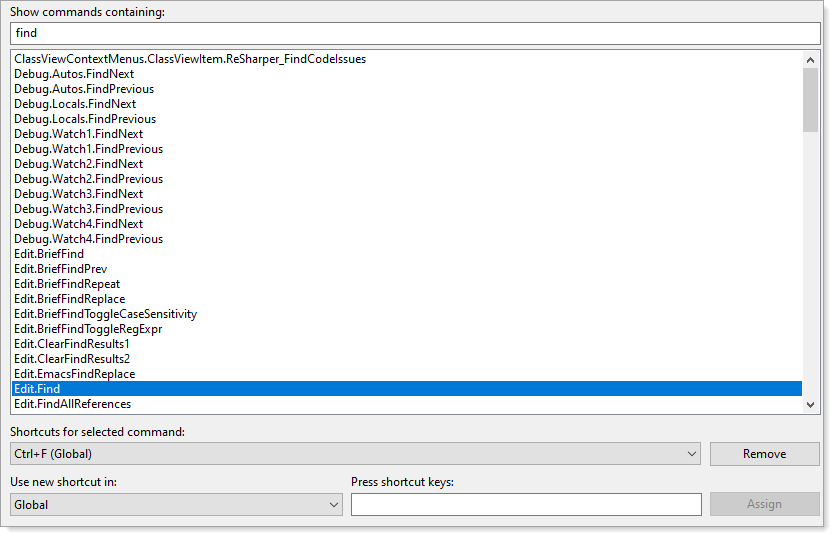
What should someone new to Visual Studio use? The recommendation is (Default).

Caveats

* Straying from (Default) means when there are instructions on the web to do a task via a shortcut means the equivalent shortcut under another mapping must be found.
* Should there be a standard for keyboard mappings? No, each developer has different preferences.



Customizing keyboard shortcuts within a keyboard mapping scheme e.g. (Default), Visual Basic 6 etc. type in the search input some text as shown below.



To change type in the input for Press shortcut keys. Be careful as selecting some shortcuts can conflict with other shortcuts especially with shortcuts that CTRL+A +B (hold down CTRL while pressing two other keys).

* Identify and customize keyboard shortcuts in Visual Studio [page](https://docs.microsoft.com/en-us/visualstudio/ide/identifying-and-customizing-keyboard-shortcuts-in-visual-studio?view=vs-2019)
* For (Default) keyboard mapping see the following [shortcut listing](https://docs.microsoft.com/en-us/visualstudio/ide/productivity-shortcuts?view=vs-2019).
* How to use the keyboard exclusively [page](https://docs.microsoft.com/en-us/visualstudio/ide/reference/how-to-use-the-keyboard-exclusively?view=vs-2019)

## Extensions

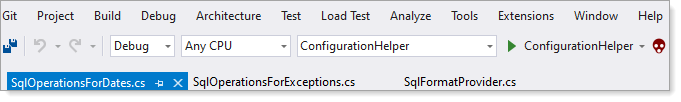
* [Keyboard Shortcuts Manager](https://marketplace.visualstudio.com/items?itemName=JustinClareburtMSFT.VSShortcutsManager) Provides menu options for backing up, restoring, and applying various keyboard shortcuts.
* [Show Keys](https://marketplace.visualstudio.com/items?itemName=MattLaceyLtd.ShowKeys) Show the default keyboard shortcuts on-screen when common commands are invoked. Intended for use during code demos or presentations to help the audience see what you're typing. VS Code has this natively.
* [ReSharper](https://www.jetbrains.com/help/resharper/Introduction__Index.html) The legendary .NET productivity tool: find and fix errors and code smells; navigate and refactor; run unit tests and write quality code faster. Much more than keyboard shortcuts.

## Window navigation

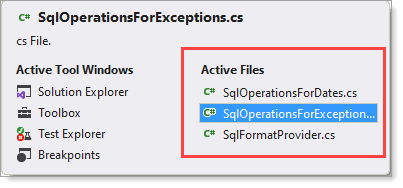
|  |
| --- |
| About customizing windows and tool windows along with saving layouts |

[Customize window layouts in Visual Studio documentation](https://docs.microsoft.com/en-us/visualstudio/ide/customizing-window-layouts-in-visual-studio?view=vs-2019)

### Switch between windows

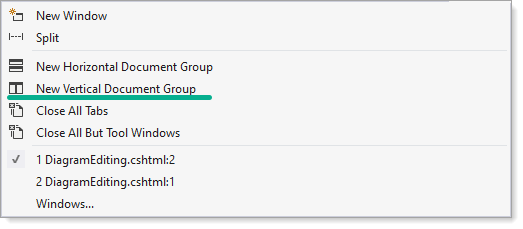


Press CTRL + TAB and CTRL + SHIFT + TAB to switch between the windows



### Split a window horizontally

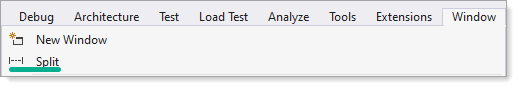
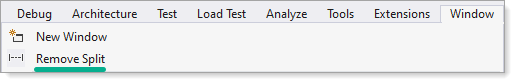
Visual Studio provides an option to split a window horizontally which can assist when working with a class with many lines of code.

* Open a file from Solution Explorer
* Select from the menu Windows → New window, the current file is duplicated.
* Select from the menu Windows → New Vertical Document Group  
  

Switch between these two windows CTRL+F6

See the following [Video](file:///C:\OED\Dotnetland\VS2019\LearningVisualStudio\Documents\Videos\HorizontalWindowSplit.mp4)

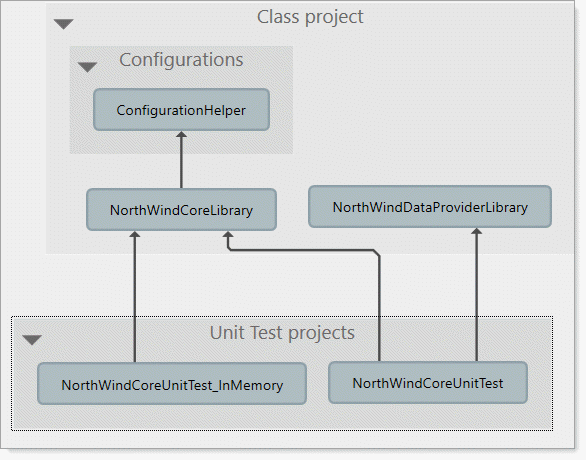
Split a window vertically

* Select from the menu Windows Split.  
  
* Remove split  
  

Use CTRL+F6 to navigate between upper and lower window.

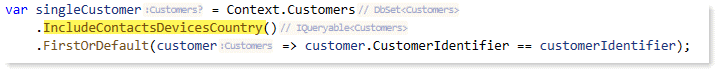
## Architectural view

When dealing with a large solution, using an Architectural view of a project can assist figuring out unfamiliar code, new to a developer or even for the developer who wrote the code. Click on a line to farther refine dependencies.

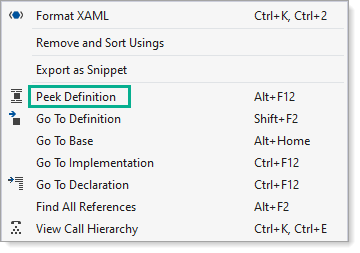


## Peek definition

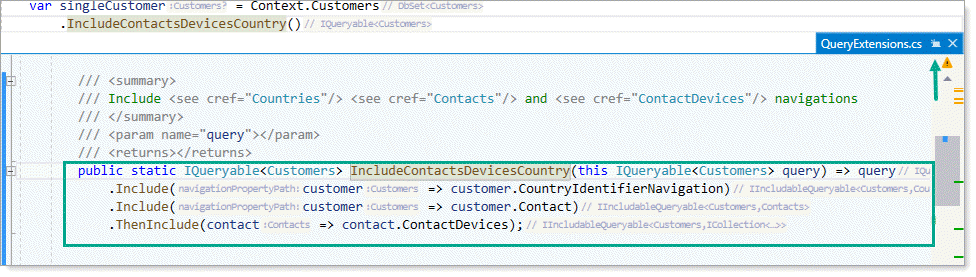
Visual Studio provides Peek Definition to view and edit code without switching away from the code that you're writing. For example, the following code, has IncludeContactsDevicesCountry which is unknown to the developer or is known but unsure where the source code is.



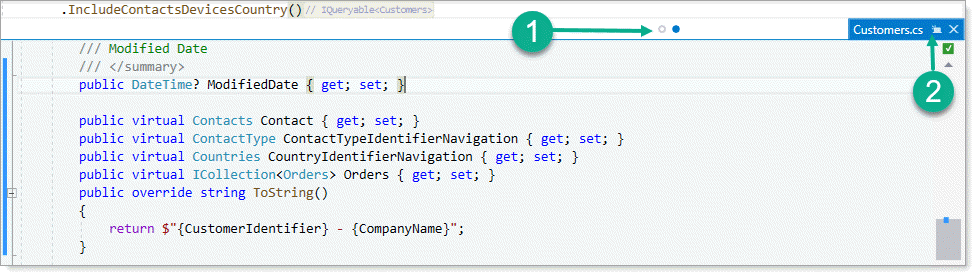
Right click on IncludeContactsDevicesCountry and select Peek Definition



We get the following



Now the developer wants to know about CountryIdentifierNavigation, right click and select Peek Definition and we get

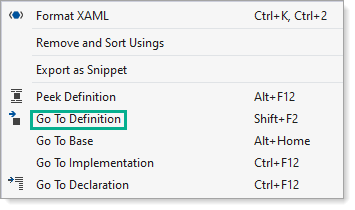


1. Provides switching between peek windows
2. Click this small button to open the current peek window in an editor window (known as promoting)

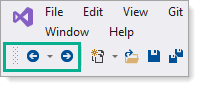
→ Video example where instead of using the context menu as shown above the shortcut key combination ALT+F12 is used.

[Peek video](file:///C:\OED\Dotnetland\VS2019\LearningVisualStudio\Documents\Videos\PeekDefinition1.mp4)

Also available, Go to definition which opens the source file in a new window.



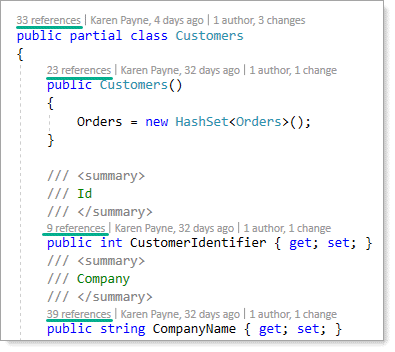
When traversing many source files or moving to different locations in a single source file a ninja tip is to use the forward and back buttons



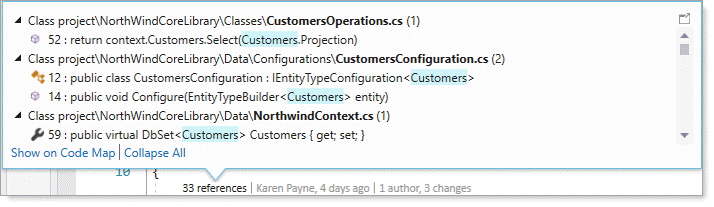
Find out references to a method or property.

**Notes**

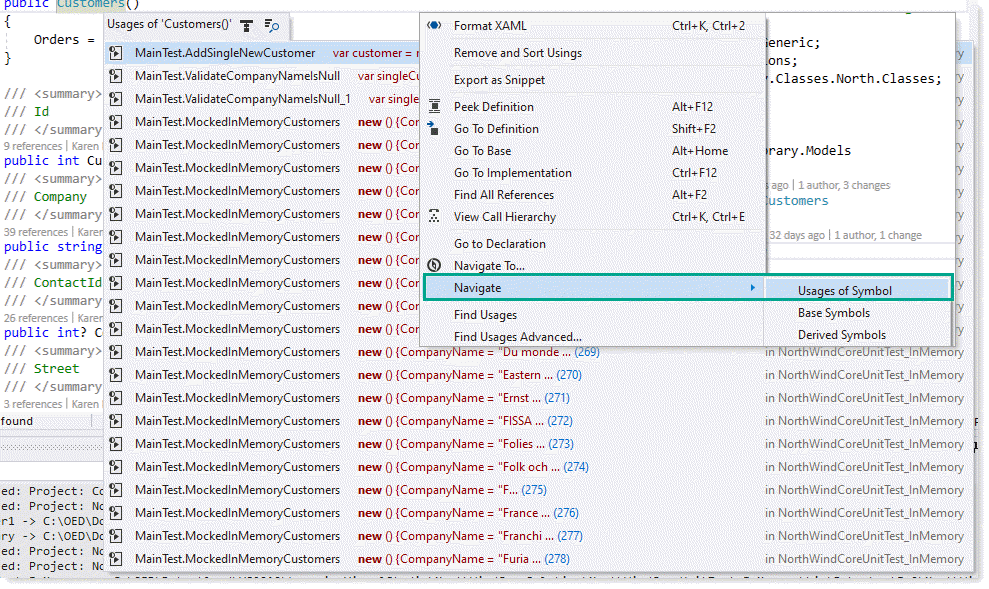
* The grey text with the green underline
  + Shows the reference/usage count.
  + Author when under source control
* May not be available except in Enterprise edition of Visual Studio



Shown below by single clicking references over the Customers class. Single click on an item with a line number to go to that code similar to peek definition.



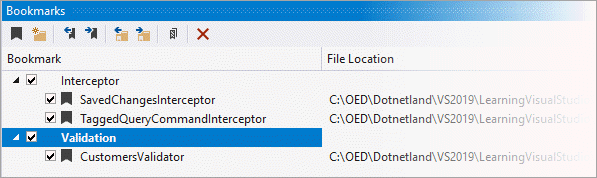
Take time to explore the above and also Use of symbol at the current cursor location as per below. As with references above, clicking on a usage will open the code in a new window.



# Bookmarks

You can use [bookmarks](https://docs.microsoft.com/en-us/visualstudio/ide/setting-bookmarks-in-code?view=vs-2019) to mark lines in your code so that you can quickly return to a specific location, or jump back and forth between locations. Bookmark commands and icons are available in two places: the Bookmark Window (View > Bookmark Window) and the text editor toolbar.

* When dealing with a lot of bookmarks create folders e.g.



# TODO List

Use Task List to track code comments that use tokens such as TODO and HACK, or custom tokens, and to manage shortcuts that take you directly to a predefined location in code. Click on the item in the list to go to its location in the source code.

Unfortunately a developer’s TODO list can not be shared with other developers.

Example for setting up via custom tokens

[TestCleanup]

public void TestCleanup()

{

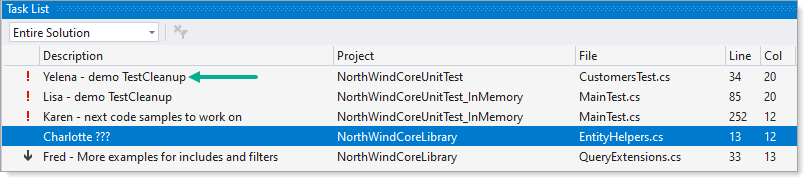
    if (TestContext.TestName == nameof(CustomerCount))

    {

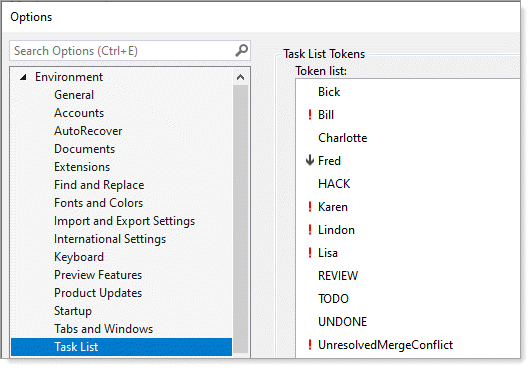
        // Yelena - demo TestCleanup

    }

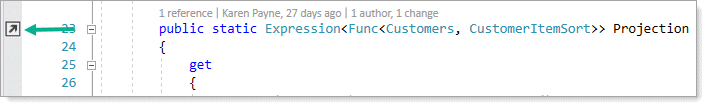
}



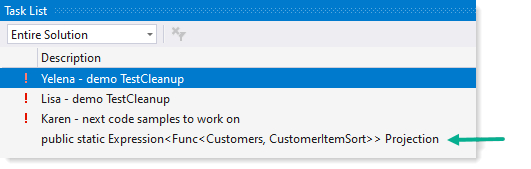
TODO items can be added, edited and removed under Tools menu → Options →Environment → Task List



* A task list item can be entered without using any of the tokens above (this has been around but not really documented), pressing CTRL+K+H will add a unnamed item e.g.



While in the task list, note the difference from two images above, no icon.



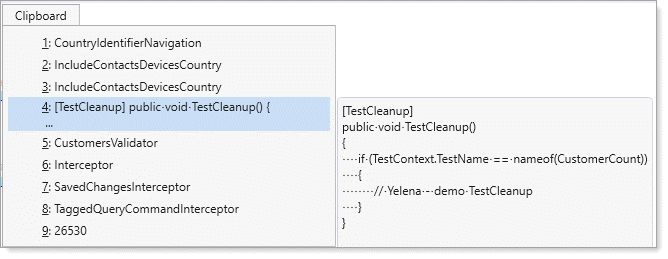
Use CTRL+K+H to remove the shortcut while in the editor, not in the Task List window.

# Visual Studio clipboard

To copy/cut/paste use Windows shortcuts e.g. CTRL+C to copy, CTRL+V to paste etc.

## Clipboard history

Press CTRL+SHIFT+V to display the clipboard history. If as shown below there is a lot of text there is a preview. Select an item with a single click to insert at the current cursor location.

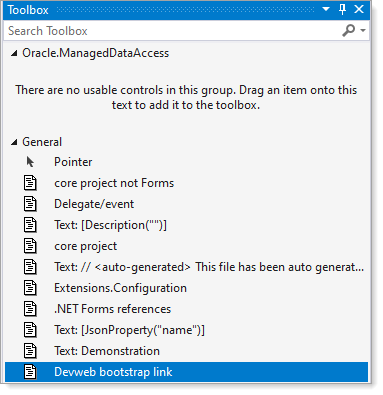


## Ninja secret

Find yourself needing specific text but also must perform regular copy and paste which loses the text, rather than resort to the clipboard history do the following.

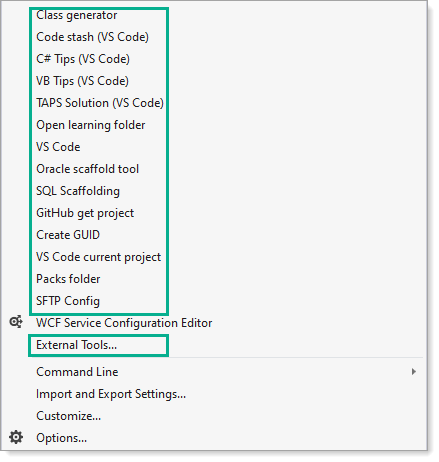
With the Toolbox window open, select the specific text and drag to the Toolbox which places the text in showing a preview of the text. To use the text in another location, select the location, double click on the text to insert into the code or drag the text from the toolbox into the code.

Super-secret Ninja trick, right click on the text and give it a meaningful name if you expect to use it for a long time. Below shows Karen’s items.



# External tools

TODO-



# Code snippets

Code snippets are small blocks of reusable code that can be inserted in a code file using a right-click menu (context menu) command or a combination of hotkeys. They typically contain commonly used code blocks such as try-finally or if-else blocks, but they can be used to insert entire classes or methods.

Most common used are to create statements e.g. if, if-else, try-catch, foreach, for etc.

Example using a simple if statement

private void ConnectionOnStateChange(object sender, StateChangeEventArgs e)

{

    if (e.CurrentState == ConnectionState.Broken || e.CurrentState == ConnectionState.Closed)

    {

    }

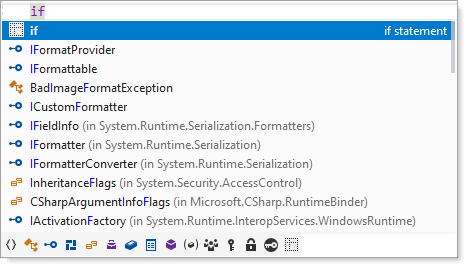
}

Type if and a window appears, press TAB to get the following with the cursor positioned in the brackets. Note that a) you may not get the dropdown or if it does appear pressing TAB quickly will dismiss the dropdown. Karen is unsure as she has a custom extension installed.

if ()

{

}



→ [Video](Videos/If_tab.mp4)

There are also surround with functionality which after selecting a block of code followed by right clicking there will be options to surround the code.

**Additionally**

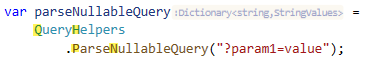
* [DPack](https://marketplace.visualstudio.com/items?itemName=SergeyM.DPack-10329) extension (free) has enhanced surround with.
* [Resharper](https://www.jetbrains.com/help/resharper/Templates__Applying_Templates__Surrounding_Code_Fragments_with_Templates.html) add-in/extension take surround with to a higher level
* [Snippet Designer](https://github.com/mmanela/SnippetDesigner) – [Market place](https://marketplace.visualstudio.com/items?itemName=vs-publisher-2795.SnippetDesigner)

Notes: For the occasional coder there is no need for Resharper while using Visual Studio as a primary IDE Resharper is a must have.

## Camel case helpers

By typing upper cased characters to quick get at class names, properties and methods.

Example, we want to use QueryHelpers.ParseQuery to write the following. We can type this all out while

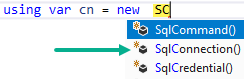


Typing **QH** gives use QueryHelpers, type a period followed by **PN** to get ParseQuery. Another example is to get a connection class of type SqlConnection.

We want

using var cn = new SqlConnection() { ConnectionString = \_connectionString };

Type **SC**



→ [StringBuilder video](Videos/CamelCasing1.mp4)

## IntelliSense

IntelliSense is a code-completion aid that includes a number of features: List Members, Parameter Info, Quick Info, and Complete Word. These features help you to learn more about the code you're using, keep track of the parameters you're typing, and add calls to properties and methods with only a few keystrokes.

Each language e.g. C#, JavaScript, HTML, CSS for instance have IntelliSense but different flavors.

[Microsoft documentation](https://docs.microsoft.com/en-us/visualstudio/ide/visual-csharp-intellisense?view=vs-2019)

√ We will be using IntelliSense intensively.

√ I’m not getting into this here as having Resharper installed I cannot fully write up on IntelliSense as with Resharper IntelliSense is supercharged.

When dealing with ASP.NET Core there are [Tag Helpers](https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro?view=aspnetcore-5.0).

Tag Helpers enable server-side code to participate in creating and rendering HTML elements in Razor files. For example, the built-in ImageTagHelper can append a version number to the image name. Whenever the image changes, the server generates a new unique version for the image, so clients are guaranteed to get the current image (instead of a stale cached image). There are many built-in Tag Helpers for common tasks - such as creating forms, links, loading assets and more - and even more available in public GitHub repositories and as NuGet packages. Tag Helpers are authored in C#, and they target HTML elements based on element name, attribute name, or parent tag. For example, the built-in LabelTagHelper can target the HTML <label> element when the LabelTagHelper attributes are applied. If you're familiar with HTML Helpers, Tag Helpers reduce the explicit transitions between HTML and C# in Razor views.

Tag Helpers can be seen as a cross between IntelliSense and code snippets as Tag Helpers will generate code as do code snippets and both use IntelliSense.

# Refactoring code

Refactoring is the process of modifying code in order to make it easier to maintain, understand, and extend, but without changing its behavior. Visual Studio provides a good deal of refactoring helpers.

For a complete list see the following [Microsoft docs](https://docs.microsoft.com/en-us/visualstudio/ide/refactoring-in-visual-studio?view=vs-2019)

## Below are a few important refactors

**Convert anonymous type to class.**

To start with, what is an anonymous type? The [official documentation](https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/anonymous-types).

In lay terms an anonymous type is local to where it was created such as a method. In the following code snippet data is read from a SQL-Server database into a strong typed list followed by grouping by category and product name from the following method GetProductsWithProjection where how data is retrieved is not relevant.

public static async Task AnonymousExample()

{

    List<Product> products = await GetProductsWithProjection();

    var localList = products

        .GroupBy(product => (product.CategoryName, product.ProductName))

        .Select(item => new

        {

            CategoryName = item.Key.CategoryName,

            ProductName = item.Key.ProductName

        })

        .ToList();

}

By hovering over localList variable we see it’s an anonymous type which is constrained to the method AnonymousExample. This is fine if we want to iterate the data and place the data into a concrete container.

For sharing the information out of the method we need to create a concrete class

public class GroupItem

{

    public string CategoryName { get; set; }

    public string ProductName { get; set; }

}

Now let’s rewrite the method AnonymousExample so data can be shared outside of the method.

public static async Task<List<GroupItem>> StrongTypedExample()

{

    var products = await GetProductsWithProjection();

    List<GroupItem> groupItems = products

        .GroupBy(product => (product.CategoryName, product.ProductName))

        .Select(item => new GroupItem

        {

            CategoryName = item.Key.CategoryName,

            ProductName = item.Key.ProductName

        }).ToList();

    return groupItems;

}

Both examples are fairly lame and done this way to keep the concept of anonymous types clear. Both code samples were taken from a useful method. Not getting into the weeds here as we will in a later class but note two concrete classes are used here.

public static async Task<List<Product>> GetProductsWithProjectionGroupByCategory()

{

    List<Product> productList = new();

    await Task.Run(async () =>

    {

        var products = await GetProductsWithProjection();

        productList = products.GroupBy(product => new CategoryProduct {CategoryName = product.CategoryName, ProductName = product.ProductName})

            .Select(product => new Product()

            {

                ProductId = products.FirstOrDefault().ProductId,

                CategoryName = product.FirstOrDefault().CategoryName,

                CategoryId = product.FirstOrDefault().CategoryId,

                ProductName = product.FirstOrDefault().ProductName,

                SupplierName = products.FirstOrDefault().SupplierName

            })

            .ToList();

    });

    return productList;

}

In the following [video](Videos/AnonymousToType/AnonymousProject/AnonymousProject.mp4) a LINQ query returns an Anonymous type of int and string which means it’s scope is limited to the method while using Resharper using a few mouse clicks creates a class to represent int and string into a new class.

**Move type to namespace**

|  |
| --- |
| The [namespace keyword](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/namespace) is used to declare a scope that contains a set of related objects. You can use a namespace to organize code elements and to create globally unique types. |

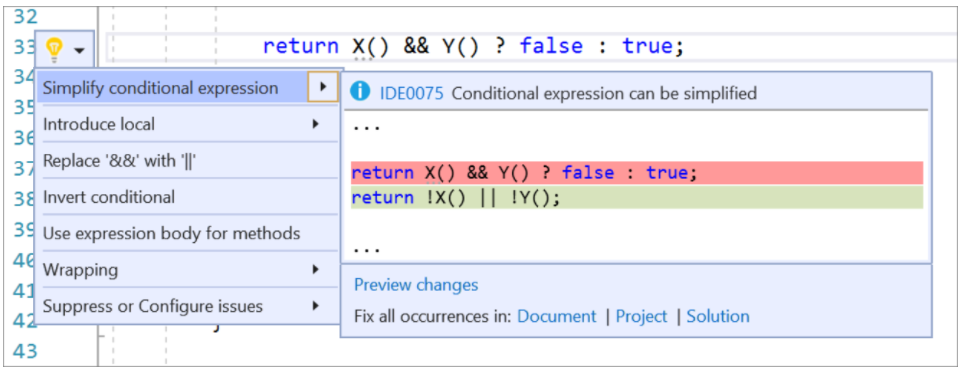
Novice developers will work with a single namespace (Karen will dig into this) while an experienced developer will have many namespaces. Suppose (and this has happened in at least two C# projects at OED) a developer copies code from one project to another without changing the namespace? This creates both confusion and makes maintaining code sometime impossible along with for novice developers compiler errors which cause the inability to build a project.

This is where [Move type to namespace](https://docs.microsoft.com/en-us/visualstudio/ide/reference/move-type-to-namespace?view=vs-2019) is a critical refactoring.

## Simplify conditional expression refactoring

Visual Studio may suggest a refactors as per below. Usually Visual Studio gets it right although sometimes not so until you understand what is right or wrong make sure to test the refactor out. There are other type of refactor recommendation, treat them as not always good until you understand what is good or bad.

**Karen’s note**: 99.9999 percent of the time I know if a refactor suggestion is good or bad which comes only from experience.



[See the docs](https://docs.microsoft.com/en-us/visualstudio/ide/reference/simplify-conditional-expression?view=vs-2019) for the above and other refactors.

# Generate a class or type

One of Karen’s favorite’s ¯\*\_(ツ)\_*/¯

Imagine writing code and a class is needed, usually this means creating a new class file and as needed add properties. Let’s look at a simple example for reading information from a file in a class method which needs to return data read from a file. We write the code for returning account information yet there is no Account class yet.

See Microsoft docs for other, similar [quick actions](https://docs.microsoft.com/en-us/visualstudio/ide/quick-actions?view=vs-2019).

public class AccountOperations

{

    private static string \_fileName = "Accounts.csv";

    public static List<Account> Read()

    {

        List<Account> accounts = new List<Account>();

        if (File.Exists(\_fileName))

        {

            var lines = File.ReadAllLines(\_fileName);

            if (lines.All(line => line.Contains(",")))

            {

                var parts = lines.Select(line => line.Split(',')).ToArray();

                if (parts.All(item => bool.TryParse(item[1], out \_)) &&

                    parts.All(item => !string.IsNullOrWhiteSpace(item[0])) &&

                    parts.All(item => int.TryParse(item[0], out \_)))

                {

                    accounts = parts.Select(data => new Account()

                    {

                        Id = Convert.ToInt32(parts[0]),

                        Active = Convert.ToBoolean(parts[1])

                    }).ToList();

                }

            }

        }

        return accounts;

    }

    public static void Write(List<Account> accounts)

    {

        var sb = new StringBuilder();

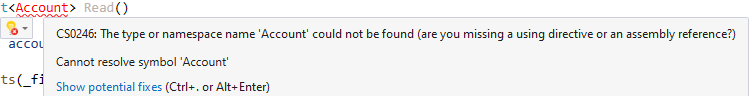
        accounts.ForEach(account => sb.AppendLine(account.Line));

        File.WriteAllText(\_fileName, sb.ToString());

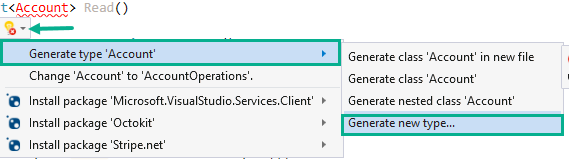
    }

}

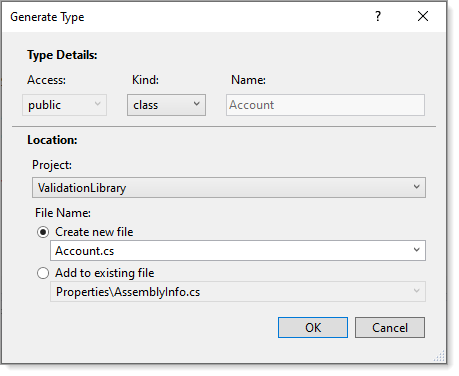
To create the Account class in a non-existing folder, hover the mouse of an instance of Account and a lightbulb appears



Click on the down arrow of the lightbulb, select Generate type ‘Account’ →Generate new type …

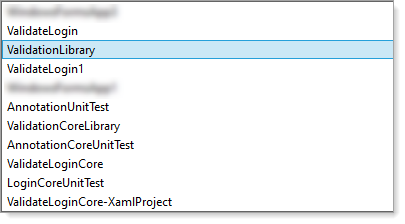


This shows the following dialog

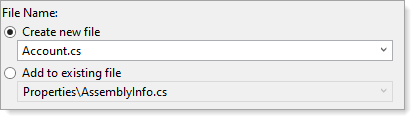


From here access is public which is correct, kind is class, again what we want.

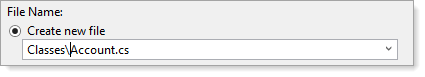
Next is Location for creating the Account class in the current project or another project in the current project.



Once a project has been selected the next step is should the file be placed into the root of a project, a sub-folder or a new folder?



If we want to place the Account class file in an existing folder named Classes we prefix Account with Classes



The Account class file will be created in the Classes folder. Suppose instead we used Classs\Account.cs, Visual Studio will create the folder and now we can move the class file (which we will get into later).

Now Account is a known type, no more red.

public static List<Account> Read()

{

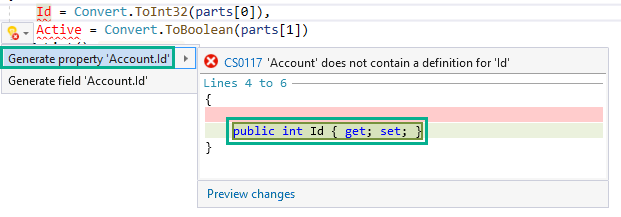
    List<Account> accounts = new List<Account>();

Next we need to handle any properties e.g.

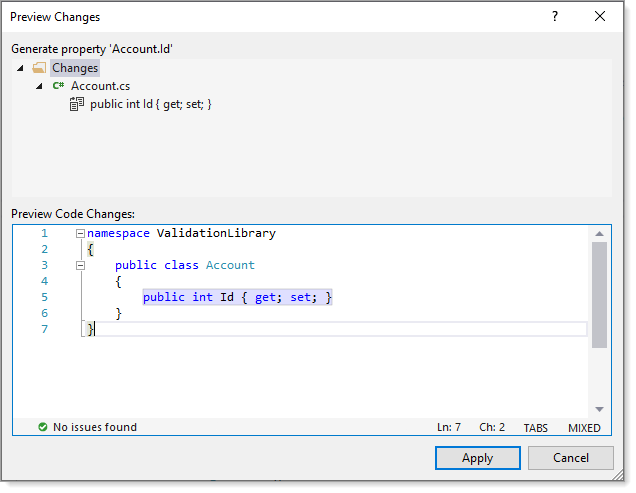
Id = Convert.ToInt32(parts[0]),

Active = Convert.ToBoolean(parts[1])

Since both properties are typed from the assignment e.g. Id is cast to a int. Hove over Id and get the same lightbulb as with the missing class to generate this property.



Note the Preview changes link above, to verify click on Preview changes to get the following, click Apply to create the propery



Do the same for Active property and Line.

public class Account

{

    public int Id { get; set; }

    public bool Active { get; set; }

    public string Line { get; set; }

}

Now the Line property is partially correct, it’s a string but we want to place Id and Active property values into this property which only the developer knows and must code it themselves.

public class Account

{

    public int Id { get; set; }

    public bool Active { get; set; }

    public string Line => $"{Id},{Active}";

}

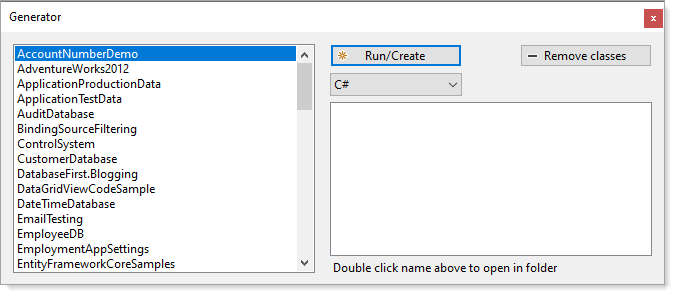
This is another way to do the above and is only worth using when the class will reside in the same folder of other classes, otherwise without doing this several times can be a pain (Karen uses both methods to generate code).

Option 3

When working with Entity Framework Core, there is a reverse engineer process to generate not only classes (known as models) but also all relationships defined in a database. Note SQL-Server works great while let’s put it this way, Oracle is on again, off again. Karen has a utility to assist in a private GitHub repository which she can provide source code for.

Option 4

Karen has a utility she created to create classes from a database when not using Entity Framework core shown below. Full source code is available in the following [GitHub repository](https://github.com/karenpayneoregon/classes-from-sqlserver).



## Generating classes from XML and JSON

Visual Studio has the capability to generate XML or JSON from a data source. Example, a Json file needs to be read into a list. The Json is online at the following [site](https://jsonplaceholder.typicode.com/users).

A single record

{

  "id": 1,

  "name": "Leanne Graham",

  "username": "Bret",

  "email": "Sincere@april.biz",

  "address": {

    "street": "Kulas Light",

    "suite": "Apt. 556",

    "city": "Gwenborough",

    "zipcode": "92998-3874",

    "geo": {

      "lat": "-37.3159",

      "lng": "81.1496"

    }

  },

  "phone": "1-770-736-8031 x56442",

  "website": "hildegard.org",

  "company": {

    "name": "Romaguera-Crona",

    "catchPhrase": "Multi-layered client-server neural-net",

    "bs": "harness real-time e-markets"

  }

}

Create a new test project.

Copy the record to the Windows clipboard followed by opening an existing class file.

* From the Visual Studio menu → Edit → Pasted Special → Paste JSON as class
* Four classes are generated, Rootobject, Address, Geo and Company
* Next since each property name is starts with lower case we want the first character to be uppercased. Add the following attribute to each property. Also for Rootobject rename to User

User example

public class User

{

    [JsonPropertyName("id")]

    public int Id { get; set; }

    [JsonPropertyName("name")]

    public string Name { get; set; }

    [JsonPropertyName("username")]

    public string UserName { get; set; }

    [JsonPropertyName("email")]

    public string Email { get; set; }

    [JsonPropertyName("address")]

    public Address Address { get; set; }

    [JsonPropertyName("phone")]

    public string Phone { get; set; }

    [JsonPropertyName("website")]

    public string Website { get; set; }

    [JsonPropertyName("company")]

    public Company Company { get; set; }

}

Add the following code to a test method

[TestMethod]

public async Task GetAllUser()

{

    using HttpClient client = new() { BaseAddress = new Uri("https://jsonplaceholder.typicode.com") };

    List<User> userList = await client.GetFromJsonAsync<List<User>>("users");

    Assert.AreEqual(userList.Count,10);

}

Run the test then upon passing add the following test method and ensure the test passes

[TestMethod]

[TestTraits(Trait.Json)]

public async Task GetSingleUser()

{

    using HttpClient client = new() {BaseAddress = new Uri("https://jsonplaceholder.typicode.com")};

    User user = await client.GetFromJsonAsync<User>("users/1");

    Assert.IsTrue(user.Name == "Leanne Graham");

    Assert.IsTrue(user.Company.Name == "Romaguera-Crona");

    Assert.IsTrue(user.Address.Geo.Latitude == "81.1496");

    Assert.IsTrue(user.Address.Geo.Longitude == "-37.3159");

}

All of the above can be done locally too as done in the following [GitHub repository](https://github.com/karenpayneoregon/unit-test-json) which can be cloned in Visual Studio or via GitDesktop.

Next steps

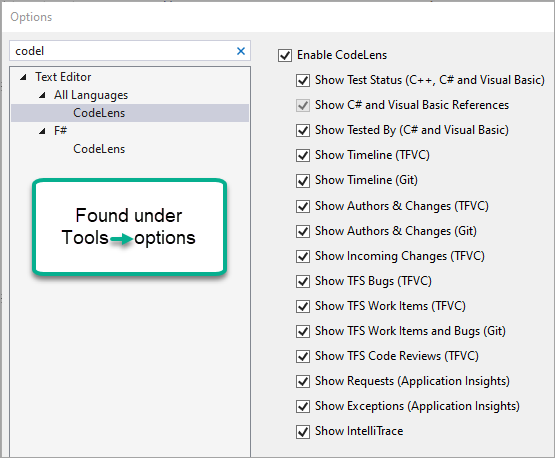
* Move the four classes to their own files under a new folder named Classes
* Override ToString (Karen will explain)
* Rerun test
* Create a new class project
  + Move the four classes to this new project
  + In the test project reference the new class project

## Informational

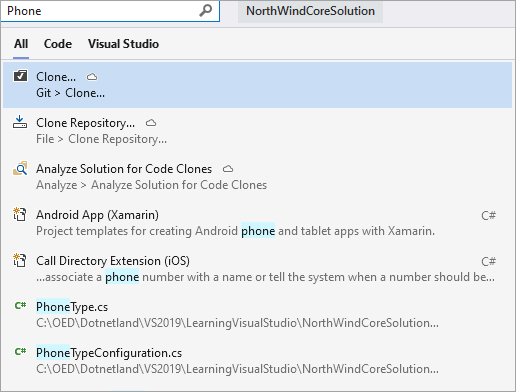
### CodeLens

→ [Video](file:///C:\OED\Dotnetland\VS2019\LearningVisualStudio\Documents\Videos\CodeLens1.mp4)

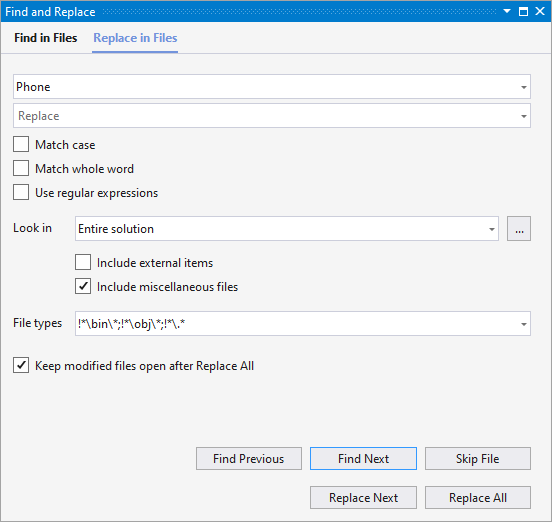
CodeLens lets you stay focused on your work while you find out what happened to your code–without leaving the editor. You can find references to a piece of code, changes to your code, linked bugs, work items, code reviews, and unit tests.

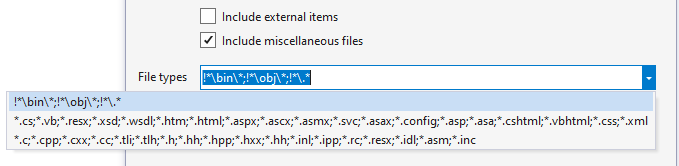


# Find and replace

* [Find and replace text](https://docs.microsoft.com/en-us/visualstudio/ide/finding-and-replacing-text?view=vs-2019)
* [Use regular expressions in Visual Studio](https://docs.microsoft.com/en-us/visualstudio/ide/using-regular-expressions-in-visual-studio?view=vs-2019)
* [Find/Command box](https://docs.microsoft.com/en-us/visualstudio/ide/find-command-box?view=vs-2019)  
  
* [Find in Files](https://docs.microsoft.com/en-us/visualstudio/ide/find-in-files?view=vs-2019)

Resharper has an enhanced find

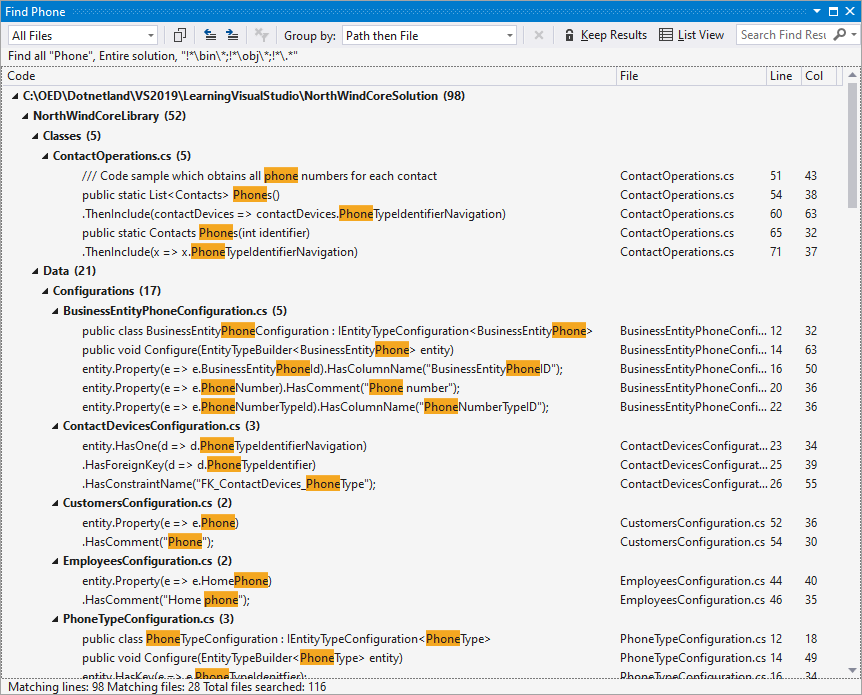




Highlight across multiple lines

By pressing (together) SHIFT+ALT+Arrow keys text can be selected followed by acted on. In the following video the Import statements are wrong, require slight adjustment to remove \_vb. Before the operations text in red could not be found because the namespace had been changed.

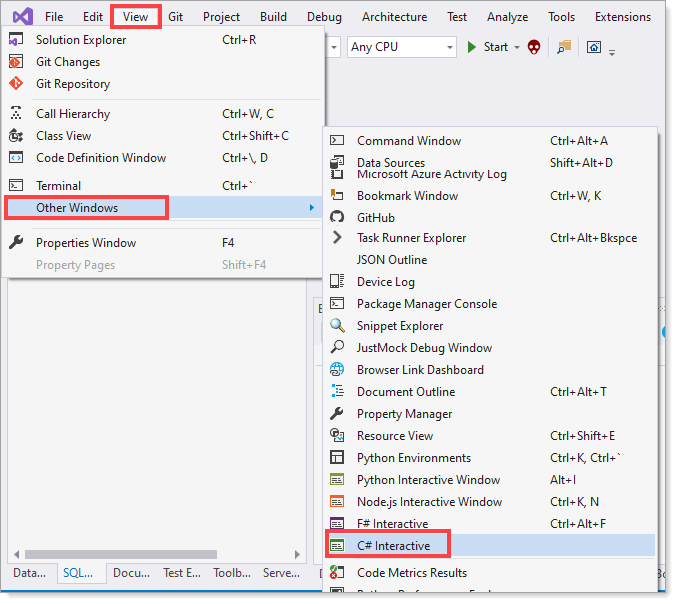
[Video](file:///C:\OED\Dotnetland\VS2019\LearningVisualStudio\Documents\Videos\ShiftAltArrows.mp4)



Writing code snippets to test a potential piece of code for inclusion in a project.

|  |
| --- |
| The C# Interactive window is a REPL Editor, i.e. Read-Evaluate-Print-Loop |

Example, we want a list of month names. Rather than write a method write out the code with C# Interactive. Open the interactive window.



Write code as seen in this [video](file:///C:\OED\Dotnetland\VS2019\LearningVisualStudio\Documents\Videos\Interactive1.gif). Once satisfied a method can be created e.g.

public class Helpers

{

    public static List<string> MonthNames() => Enumerable.Range(1, 12).

        Select((index) => DateTimeFormatInfo.CurrentInfo.GetMonthName(index)).ToList();

}