Table of Contents for This Appendix

Additional Tips from Chapter 1

Additional Tips from Chapter 2

Additional Tips from Chapter 3

Additional Tips from Chapter 4

Additional Tips from Chapter 5

Additional Tips from Chapter 6

Additional Tips from Chapter 7

Appendix B

Additional Tips

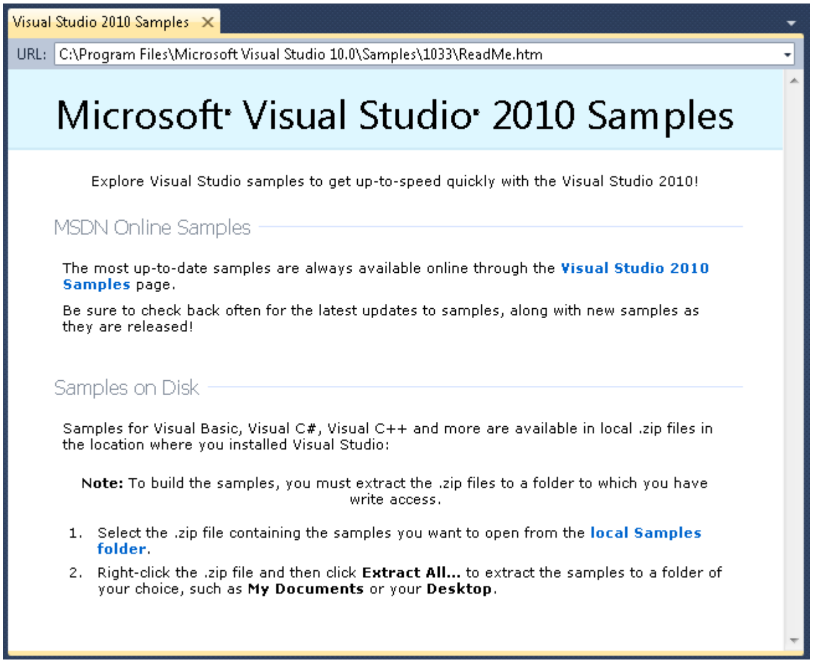
Additional Tips from Chapter 1

AX.01 Getting Help Samples

WINDOWS	Alt,H, L
MENU	Help | Samples
COMMAND	Help.Samples
VERSIONS	2008,2010
CODE	vstipEnv0002

This is another one of those things that is always there but that developers tend to forget about. You can get sample code from within Visual Studio itself. Just select Help | Samples from your menu bar.

What you see next is version-specific—for example, in Visual Studio 2010, you will see the page shown below:


Visual Studio 2010 Samples
URL: C:\Program Files\Microsoft Visual Studio 10.0\Samples\1033\ReadMe.htm

Microsoft® Visual Studio® 2010 Samples

Explore Visual Studio samples to get up-to-speed quickly with the Visual Studio 2010!

MSDN Online Samples

The most up-to-date samples are always available online through the **Visual Studio 2010 Samples** page.

Be sure to check back often for the latest updates to samples, along with new samples as they are released!

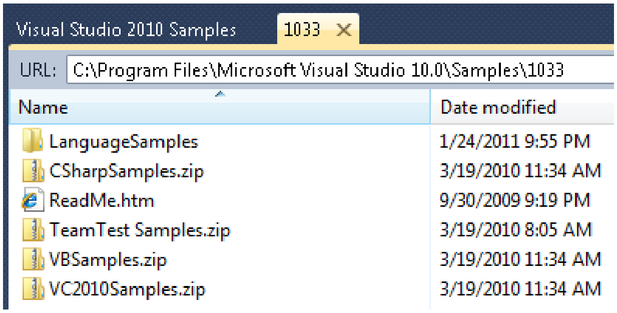
Samples on Disk

Samples for Visual Basic, Visual C#, Visual C++ and more are available in local .zip files in the location where you installed Visual Studio:

Note: To build the samples, you must extract the .zip files to a folder to which you have write access.

1. Select the .zip file containing the samples you want to open from the **local Samples folder**.
2. Right-click the .zip file and then click **Extract All...** to extract the samples to a folder of your choice, such as **My Documents** or your **Desktop**.

Click the Local Samples Folder link, and you see the folder shown in the following image:



URL: C:\Program Files\Microsoft Visual Studio 10.0\Samples\1033

Name	Date modified
LanguageSamples	1/24/2011 9:55 PM
CSharpSamples.zip	3/19/2010 11:34 AM
ReadMe.htm	9/30/2009 9:19 PM
TeamTest Samples.zip	3/19/2010 8:05 AM
VBSamples.zip	3/19/2010 11:34 AM
VC2010Samples.zip	3/19/2010 11:34 AM

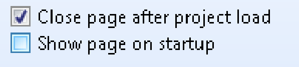
From here, you just unzip the code samples for the language you are interested in.

AX.02 Make the Start Page Go Away

WINDOWS	Alt,V, G
MENU	View | Start Page
COMMAND	View.StartPage
VERSIONS	2010
CODE	vstipTool0002

Does the Start Page appearing every time you start Visual Studio annoy you? You can set up Visual Studio so that the Start Page doesn't load when you start Visual Studio. Just look in the lower-left corner, and clear the Show Page On Startup check box.

From now on, the Start Page shows up only when you want it to.



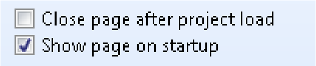
☑ Close page after project load
☐ Show page on startup

AX.03 Bringing Back the Start Page

WINDOWS	Alt,V, G
MENU	View | Start Page
COMMAND	View.StartPage
VERSIONS	2010
CODE	vstipTool0001

You might have noticed that when you open up a Solution or Project, the Start Page goes away. This is the new default behavior in Visual Studio 2010.

To change it, look in the lower-left corner of the new and improved Start Page, and clear the Close Page After Project Load check box. From now on, the Start Page sticks around until you close it yourself.

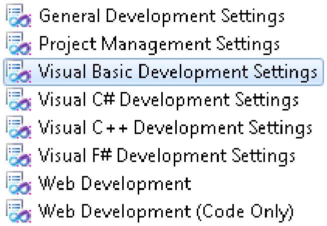


☐ Close page after project load
☑ Show page on startup

AX.04 Show All Settings with Visual Basic

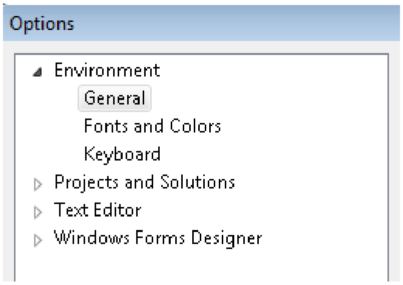
WINDOWS	Alt,T, O
MENU	Tools | Options
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
LANGUAGES	VB
CODE	vstipEnv0042

Did you choose the Visual Basic settings during your install?

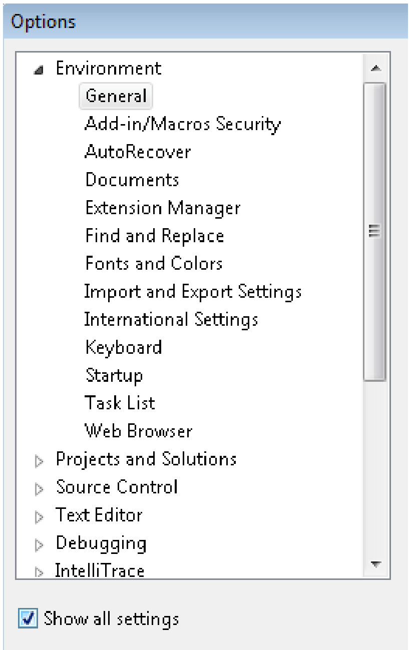


General Development Settings
Project Management Settings
Visual Basic Development Settings
Visual C# Development Settings
Visual C++ Development Settings
Visual F# Development Settings
Web Development
Web Development (Code Only)

If so, you might notice, when you open the Tools | Options menu, that the options are not all there:



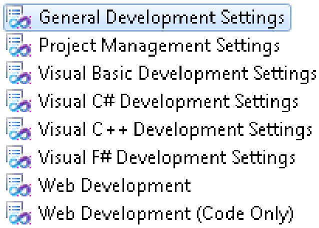
To bring them back, just select Show All Settings at the bottom-left of the dialog box, and all the available settings will reappear:



AX.05 Find Your Development Settings

VERSIONS	2005, 2008, 2010
CODE	vstipEnv0020

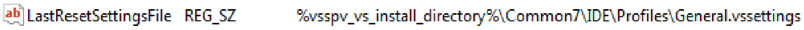
Development settings determine quite a bit when you use Visual Studio. For example, they determine how you see the installed templates and what options you see initially in the Tools | Options dialog box. You probably remember the following choices when you installed Visual Studio:


General Development Settings
Project Management Settings
Visual Basic Development Settings
Visual C# Development Settings
Visual C++ Development Settings
Visual F# Development Settings
Web Development
Web Development (Code Only)

If you happen to forget what choice you made, you can quickly get a reminder by going to the registry key HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\<version>\Profile.

Warning Don’t make any changes in the Registry for this tip; just view the data.

So, for Visual Studio 2010, the path would be HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\10.0\Profile, and then you would look at the “LastResetSettingsFile” string:



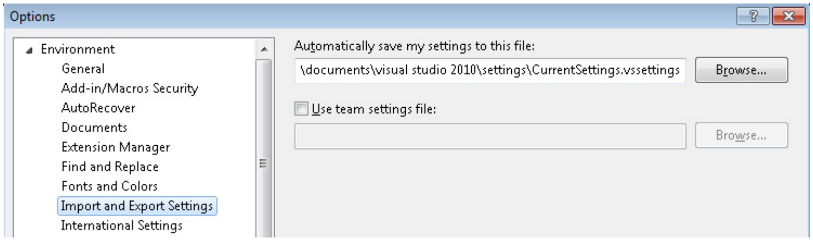
LastResetSettingsFile REG_SZ %vsspv_vs_install_directory%\Common7\IDE\Profiles\General.vssettings

Assuming you haven’t reset the settings via Tools | Import And Export Settings lately, this value should have the name of the settings file you chose at first launch. In my case, as shown in the preceding graphic, I chose the “General Development Settings” (General.vssettings) option.

AX.06 Settings Automatically Saved On Exit

WINDOWS	Alt,T, I
MENU	Tools | Options | Import and Export Settings
COMMAND	Tools.ImportandExportSettings
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0026

Visual Studio automatically saves your settings every time it is closed. To see where this file is located (or to change the location), go to Tools | Options | Import And Export Settings and locate the Automatically Save My Settings To This File area:



Automatically save my settings to this file: \documents\visual studio 2010\settings\CurrentSettings.vssettings

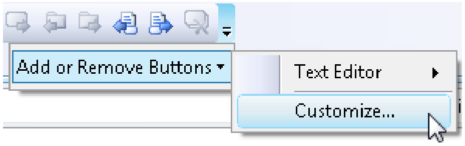
☐ Use team settings file:

The nice thing is that you can use this to “undo” any changes you have made during a session. So if you made some changes to Visual Studio but don’t want to keep them, you can import the CurrentSettings.vssettings (default name) file to bring back your settings from the last time Visual Studio was closed.

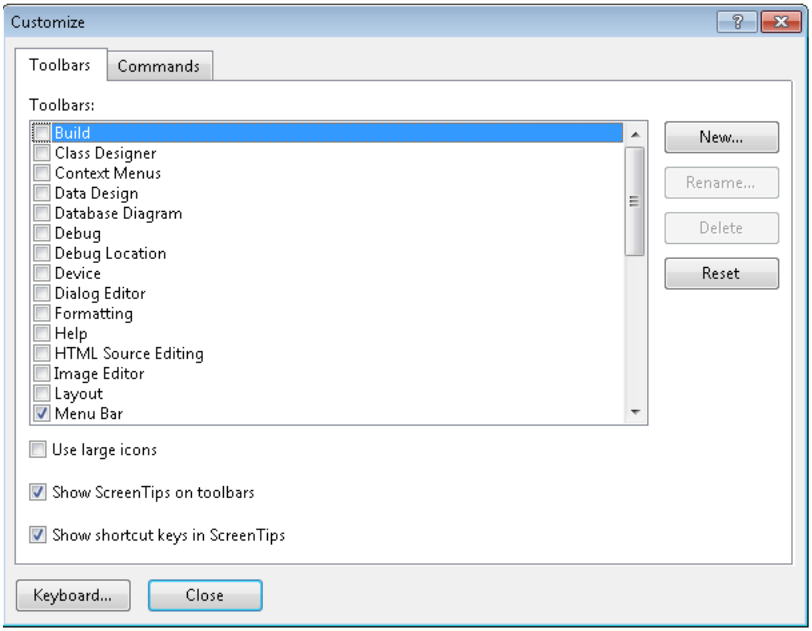
AX.07 Customize Your Toolbars in Visual Studio 2008: Toolbars Tab

WINDOWS	Alt,T, C
MENU	Tools | Customize
COMMAND	Tools.Customize
VERSIONS	2008
CODE	vstipEnv0032

You can customize any toolbar in Visual Studio 2008. Just click the drop-down arrow to the right of any toolbar, click Add Or Remove Buttons, and choose Customize:



Whichever option you choose opens the Customize dialog box:



Toolbars:

☐ Build
☐ Class Designer
☐ Context Menus
☐ Data Design
☐ Database Diagram
☐ Debug
☐ Debug Location
☐ Device
☐ Dialog Editor
☐ Formatting
☐ Help
☐ HTML Source Editing
☐ Image Editor
☐ Layout
☑ Menu Bar

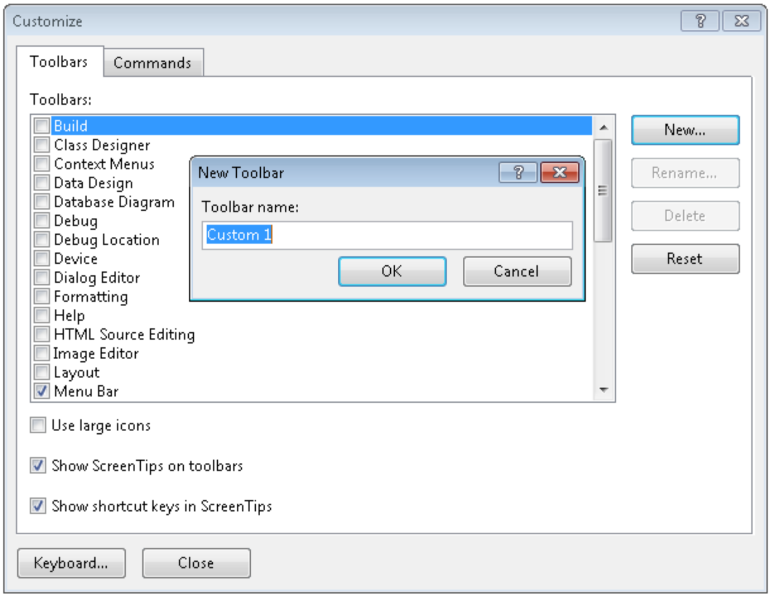
☐ Use large icons

☑ Show ScreenTips on toolbars

☑ Show shortcut keys in ScreenTips

Custom Toolbars

Notice that the Toolbars tab lists all the available toolbars. When you click New to create a customized toolbar, you are prompted to give the new toolbar a name:



Toolbars:

- ☐ Build
- ☐ Class Designer
- ☐ Context Menus
- ☐ Data Design
- ☐ Database Diagram
- ☐ Debug
- ☐ Debug Location
- ☐ Device
- ☐ Dialog Editor
- ☐ Formatting
- ☐ Help
- ☐ HTML Source Editing
- ☐ Image Editor
- ☐ Layout
- ☑ Menu Bar

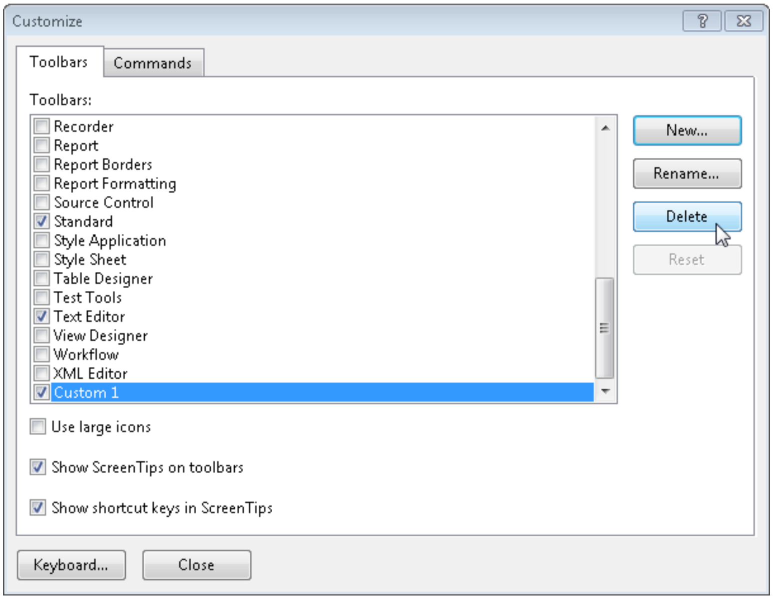
Toolbar name: Custom 1

☐ Use large icons

☑ Show ScreenTips on toolbars

☑ Show shortcut keys in ScreenTips

After you name it, you can delete the custom toolbar by clicking Delete, or you can rename it by clicking Rename:



Toolbars:

- ☐ Recorder
- ☐ Report
- ☐ Report Borders
- ☐ Report Formatting
- ☐ Source Control
- ☑ Standard
- ☐ Style Application
- ☐ Style Sheet
- ☐ Table Designer
- ☐ Test Tools
- ☑ Text Editor
- ☐ View Designer
- ☐ Workflow
- ☐ XML Editor
- ☑ Custom 1

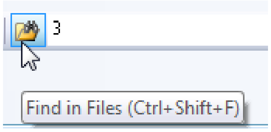
☐ Use large icons

☑ Show ScreenTips on toolbars

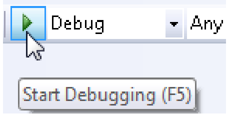
☑ Show shortcut keys in ScreenTips

Additionally, you can take advantage of the following options at the bottom of the dialog box:

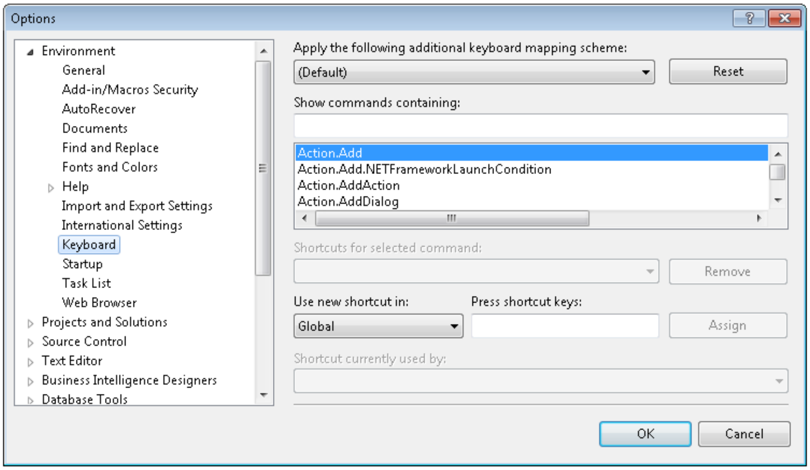
- **Use Large Icons** Shows larger icons on toolbars in your environment.
- **Show ScreenTips On Toolbars** Indicates whether you see a tooltip for the toolbar items:



- **Show Shortcut Keys In ScreenTips** Shows the shortcut key combinations in the tooltips for items that have them.



Clicking the Keyboard button at the bottom of the Customize dialog box is just the same as going to Tools | Options | Keyboard (see vstipTool0063, “Keyboard Shortcuts: Creating New Shortcuts,” on page 127 for details):



Apply the following additional keyboard mapping scheme: (Default)

Show commands containing:

Action.Add
Action.Add.NETFrameworkLaunchCondition
Action.AddAction
Action.AddDialog

Shortcuts for selected command:

Use new shortcut in: Global

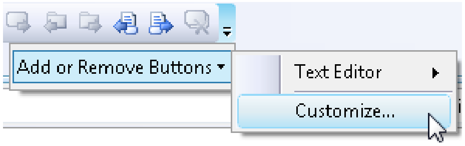
Press shortcut keys:

Shortcut currently used by:

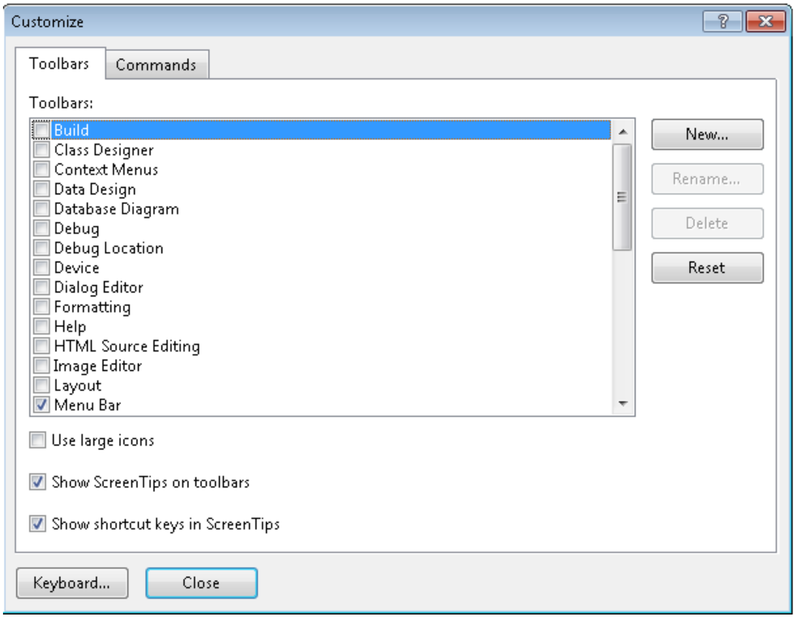
AX.08 Customize Your Toolbars in Visual Studio 2008: Commands Tab

WINDOWS	Alt,T, C
MENU	Tools | Customize
COMMAND	Tools.Customize
VERSIONS	2008
CODE	vstipEnv0033

You can customize any toolbar in Visual Studio 2008. Just click the drop-down arrow to the right of any toolbar, click Add Or Remove Buttons, and choose Customize:



Alternatively, you can go to Tools | Customize on the menu bar. Whichever option you choose opens the Customize dialog box:



Toolbars:

- ☐ Build
- ☐ Class Designer
- ☐ Context Menus
- ☐ Data Design
- ☐ Database Diagram
- ☐ Debug
- ☐ Debug Location
- ☐ Device
- ☐ Dialog Editor
- ☐ Formatting
- ☐ Help
- ☐ HTML Source Editing
- ☐ Image Editor
- ☐ Layout
- ☑ Menu Bar

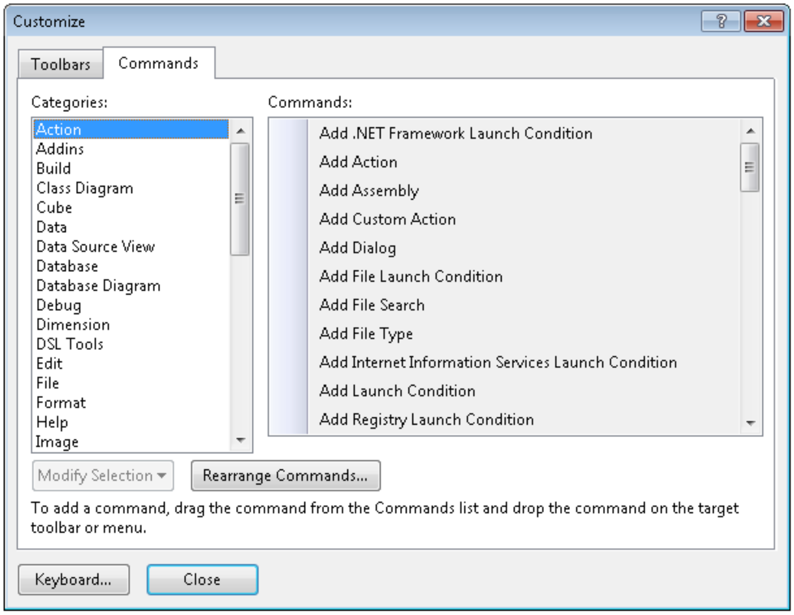
☐ Use large icons

☑ Show ScreenTips on toolbars

☑ Show shortcut keys in ScreenTips

In this case, let's look at the Commands tab:

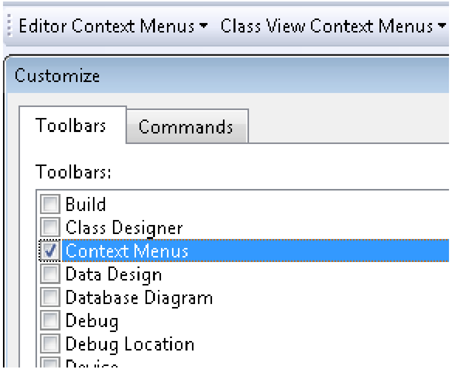
Note To learn more about the Toolbars tab, see vstipEnv0032, "Customize Your Toolbars in Visual Studio 2008 Toolbars Tab," on page A12.



To add a command, drag the command from the Commands list and drop the command on the target toolbar or menu.

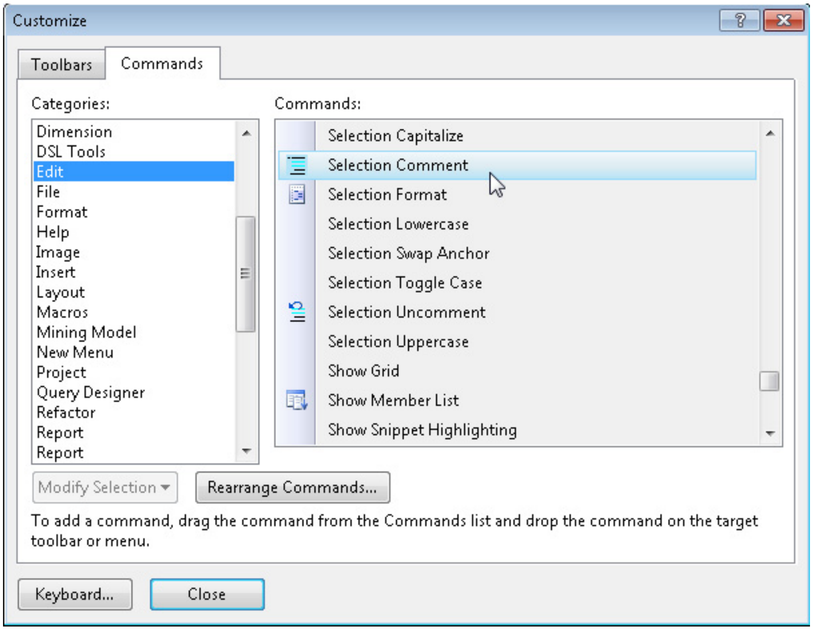
The best way to learn how to customize menus and toolbars is to work through an example. For our purposes, we want to add the ability to select some code, right-click, and comment or uncomment the code.

First, we have to go back to the Toolbars tab and pick the menu or toolbar we want to modify. For our example, let's choose Context Menus:



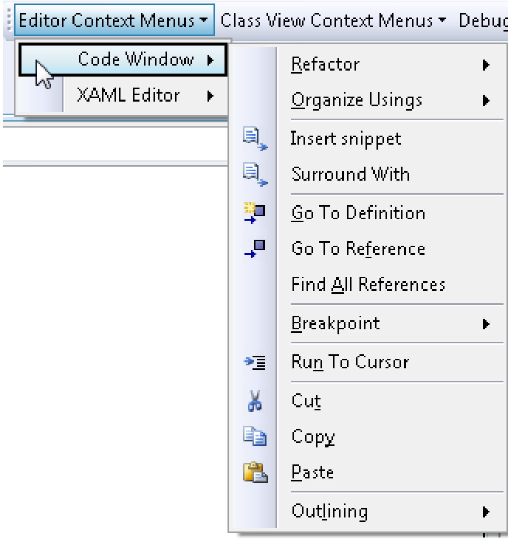
☐ Build
☐ Class Designer
☑ Context Menus
☐ Data Design
☐ Database Diagram
☐ Debug
☐ Debug Location

Now we click the Commands tab and locate the items we want to add. In this case, we dig a bit and find Selection Comment and Selection Uncomment in the Edit category:

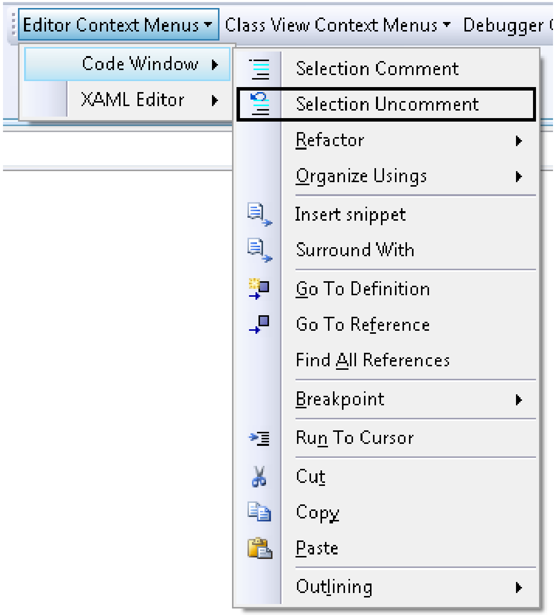


To add a command, drag the command from the Commands list and drop the command on the target toolbar or menu.

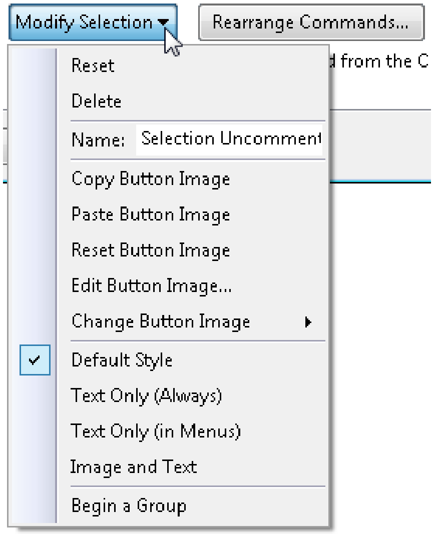
Now we have to see which menu we want to modify. Let's navigate to Editor Context Menus | Code Window to see where we want our items to go:



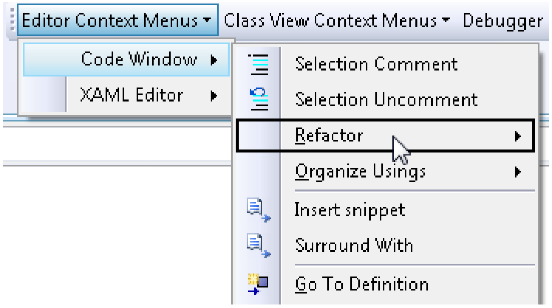
Now we drag the items where we want them on the menu:



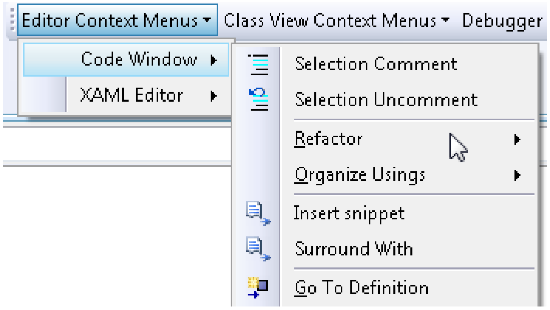
If we want, we can click Modify Selection to reset, delete, modify the name, modify the button image, change the way the item is displayed, or begin a new group:



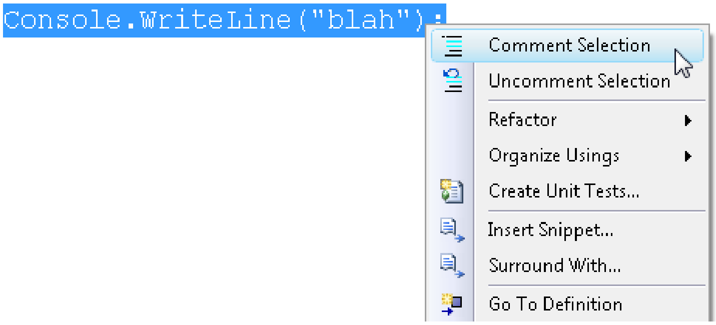
We would like the new buttons to be in their own group, so we click the item just *below* where we want our group to be:



Now we click Modify Selection and choose Begin A Group to get a new group line:



Close the Customize dialog box, select some code, and right-click to see whether our items show up:

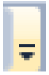


```
Console.WriteLine("blah");
```

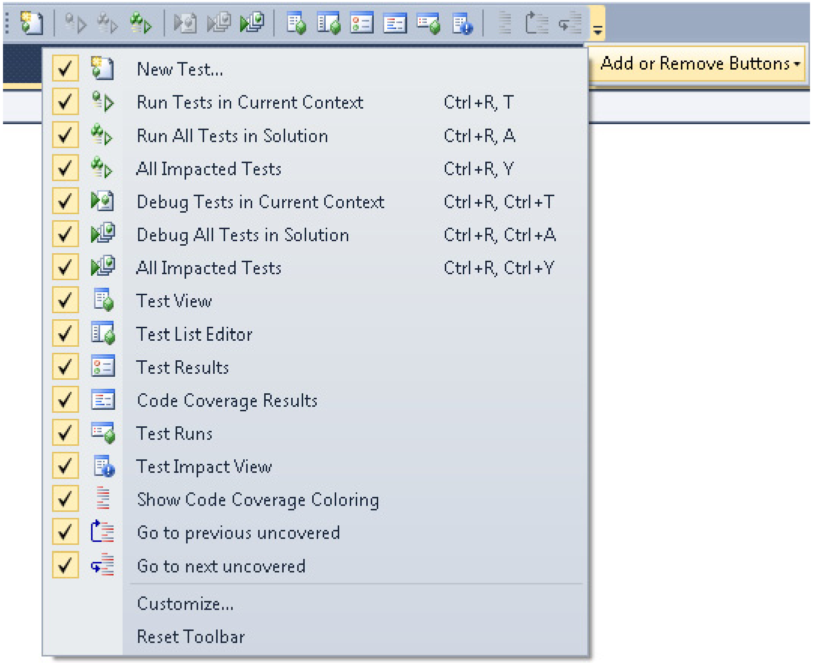
AX.09 Hide or Show Default Buttons on a Toolbar

VERSIONS	2005, 2008, 2010
CODE	vstipEnv0028

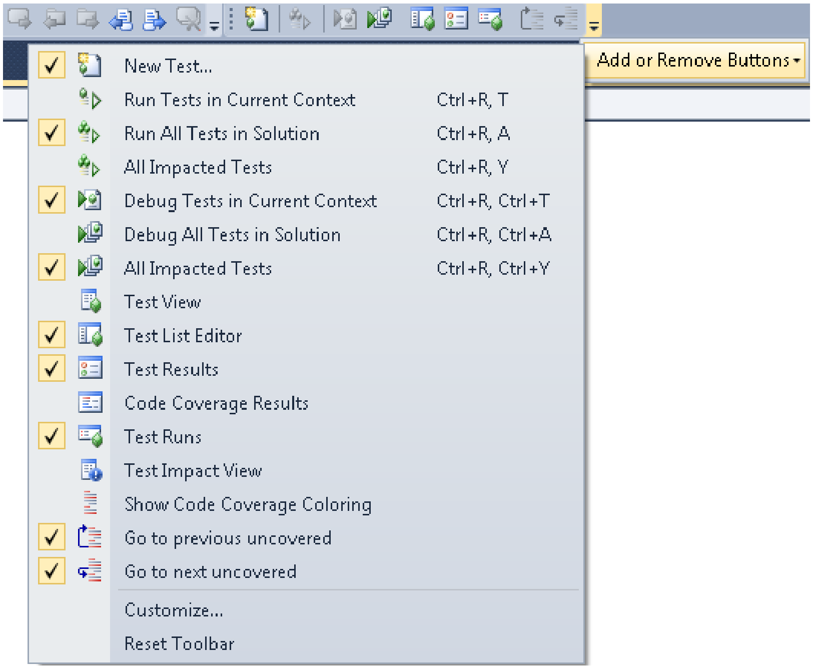
You can pick and choose which default buttons you want to show on any toolbar. Just click the drop-down arrow to the right of the toolbar:



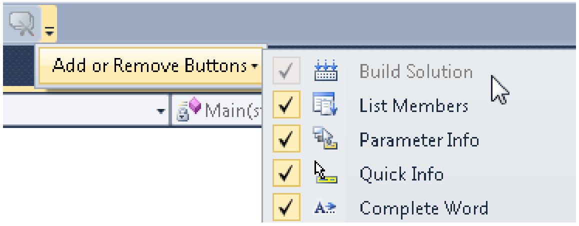
Then click Add or Remove Buttons:


Add or Remove Buttons
☑ New Test...
☑ Run Tests in Current Context Ctrl+R, T
☑ Run All Tests in Solution Ctrl+R, A
☑ All Impacted Tests Ctrl+R, Y
☑ Debug Tests in Current Context Ctrl+R, Ctrl+T
☑ Debug All Tests in Solution Ctrl+R, Ctrl+A
☑ All Impacted Tests Ctrl+R, Ctrl+Y
☑ Test View
☑ Test List Editor
☑ Test Results
☑ Code Coverage Results
☑ Test Runs
☑ Test Impact View
☑ Show Code Coverage Coloring
☑ Go to previous uncovered
☑ Go to next uncovered
Customize...
Reset Toolbar

As you can see, this gives you a list of the buttons currently being viewed. You can uncheck some of them to create a customized look for your toolbar:


Add or Remove Buttons
✓ New Test...
Run Tests in Current Context Ctrl+R, T
✓ Run All Tests in Solution Ctrl+R, A
All Impacted Tests Ctrl+R, Y
✓ Debug Tests in Current Context Ctrl+R, Ctrl+T
Debug All Tests in Solution Ctrl+R, Ctrl+A
✓ All Impacted Tests Ctrl+R, Ctrl+Y
Test View
✓ Test List Editor
✓ Test Results
Code Coverage Results
✓ Test Runs
Test Impact View
Show Code Coverage Coloring
✓ Go to previous uncovered
✓ Go to next uncovered
Customize...
Reset Toolbar

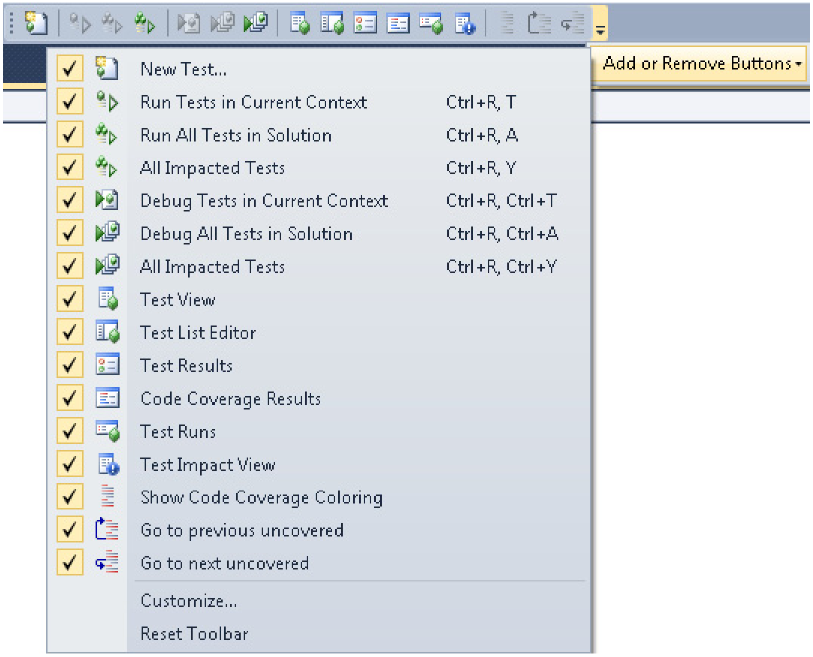
Unfortunately, if you have added customized buttons, they can't be hidden in this way:



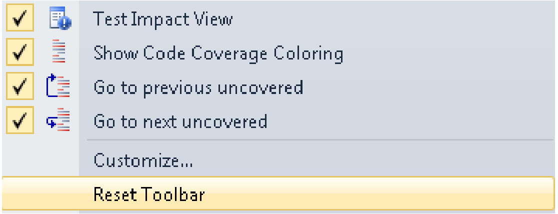
AX.10 Reset Toolbars

VERSIONS	2005, 2008, 2010
CODE	vstipEnv0029

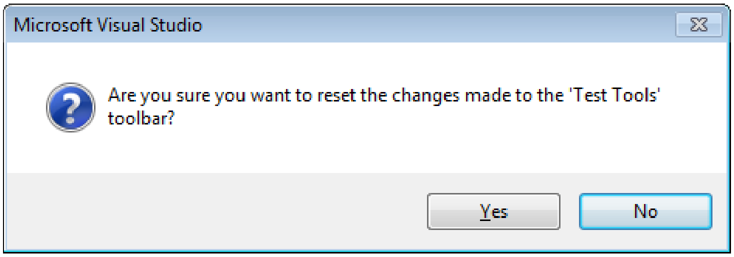
You can reset any toolbar to its default settings. Just click the drop-down arrow to the right of any toolbar, and then click Add or Remove Buttons:


Add or Remove Buttons
New Test...
Run Tests in Current Context Ctrl+R, T
Run All Tests in Solution Ctrl+R, A
All Impacted Tests Ctrl+R, Y
Debug Tests in Current Context Ctrl+R, Ctrl+T
Debug All Tests in Solution Ctrl+R, Ctrl+A
All Impacted Tests Ctrl+R, Ctrl+Y
Test View
Test List Editor
Test Results
Code Coverage Results
Test Runs
Test Impact View
Show Code Coverage Coloring
Go to previous uncovered
Go to next uncovered
Customize...
Reset Toolbar

Click Reset Toolbar:



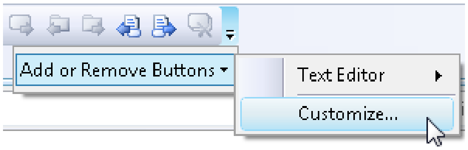
You now see the following dialog box:



Are you sure you want to reset the changes made to the 'Test Tools' toolbar?

Click Yes to remove any custom buttons and to reset the toolbar to its default settings.

Note In Visual Studio 2008, you have to go through an extra menu to get to the Reset Toolbar option:

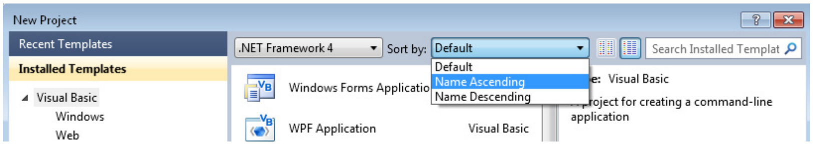


Additional Tips from Chapter 2

AX.11 Sorting Templates in the New Project Dialog Box

DEFAULT	Ctrl+Shift+N
VISUAL BASIC 6	Ctrl+Shift+N; Ctrl+N
VISUAL C# 2005	Ctrl+Shift+N
VISUAL C++ 2	Ctrl+Shift+N
VISUAL C++ 6	Ctrl+Shift+N
VISUAL STUDIO 6	Ctrl+N
WINDOWS	Alt,F, N, P (new project); Alt,F, D, N (add new project)
MENU	File | New Project; File | Add New Project
COMMAND	File.NewProject; File.AddNewProject
VERSIONS	2010
CODE	vstipProj0003

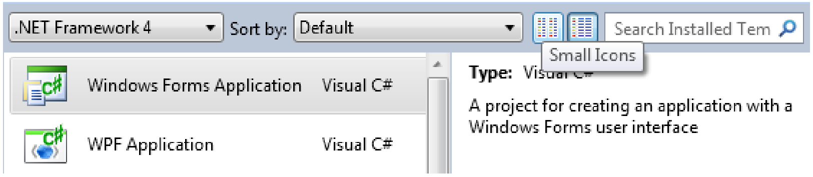
Ever just want to have an alphabetical list of templates in the New Project dialog box? Just use the new Sort By drop-down list:



AX.12 Toggle Icon Size in the New Project Dialog Box

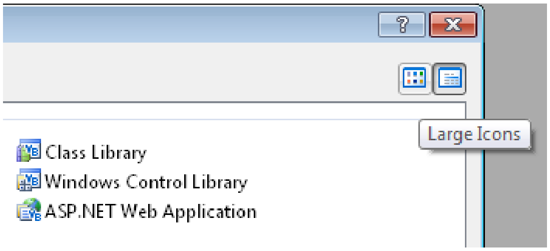
DEFAULT	Ctrl+Shift+N
VISUAL BASIC 6	Ctrl+Shift+N; Ctrl+N
VISUAL C# 2005	Ctrl+Shift+N
VISUAL C++ 2	Ctrl+Shift+N
VISUAL C++ 6	Ctrl+Shift+N
VISUAL STUDIO 6	Ctrl+N
WINDOWS	Alt, F, N, P (new project); Alt, F, D, N (add new project)
MENU	File | New | Project
COMMAND	File.NewProject
VERSIONS	2005,2008,2010
CODE	vstipProj0007

When you create a new project (or add a new item) in Visual Studio, you can change the icon size from small to medium (called *large* in Visual Studio 2008). To do this, you need to find the buttons that change the icon to your desired size and then click them. Unfortunately, these buttons appear in different places, depending on your Visual Studio version. In Visual Studio 2010, for example, the buttons are located toward the middle-right of the New Project or Item dialog box, next to the Sort By field:



A project for creating an application with a Windows Forms user interface

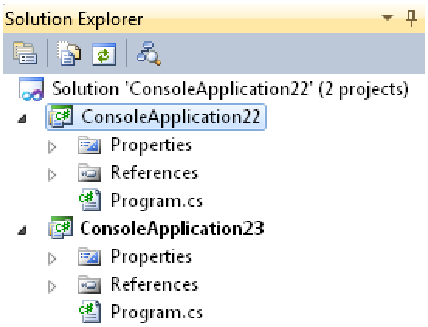
In Visual Studio 2005 and Visual Studio 2008, the buttons are at the far right, as shown in the following illustration:



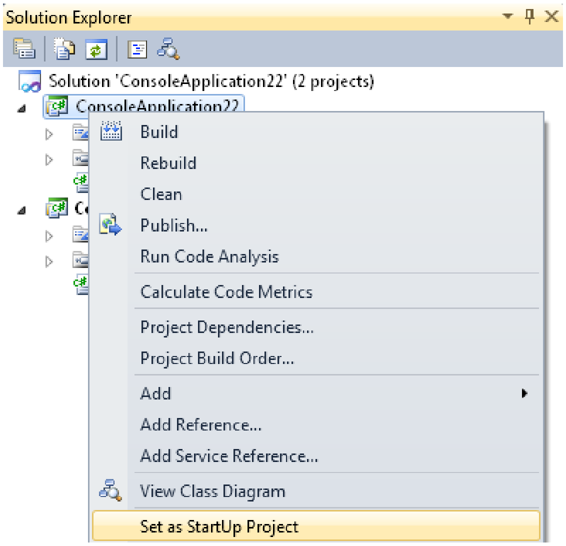
AX.13 Choosing the StartUp Project

WINDOWS	Alt,P, A (with project selected in Solution Explorer); Shift+F10, A (with project selected in Solution Explorer)
MENU	Project | Set as StartUp Project; [Right-Click a project in Solution Explorer] | Set as StartUp Project
COMMAND	Project.SetasStartUpProject
VERSIONS	2005,2008,2010
CODE	vstipEnv0014

When you work with multiple projects, one is usually the StartUp Project. That's the one that starts up first when you start (with or without debugging). It's easy to spot the current StartUp Project because its name appears in bold type in Solution Explorer:



To quickly change the Startup Project, just right-click a project in Solution Explorer and choose Set As StartUp Project from the context menu:

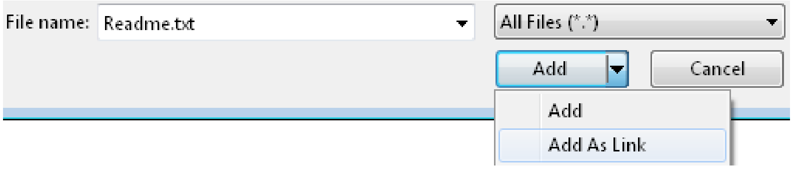


AX.14 Linked Items in Projects

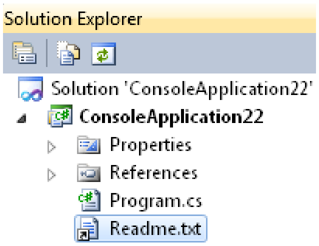
DEFAULT	Shift+Alt+A; Ctrl+Shift+D
VISUAL BASIC 6	Ctrl+D; Shift+Alt+A
VISUAL C# 2005	Shift+Alt+A
VISUAL C++ 2	Shift+Alt+A
VISUAL C++ 6	Shift+Alt+A
VISUAL STUDIO 6	Shift+Alt+A; Ctrl+Shift+D
WINDOWS	Alt, P, G
MENU	Project | Add Existing Item
COMMAND	Project.AddExistingItem
VERSIONS	2005,2008,2010
CODE	vstipProj0022

You sometimes have a shared resource that you want to include in your project. Traditionally, you would add the existing item (Shift+Alt+A), and Visual Studio would make a copy for you.

However, did you know you can just link to the item instead? You would typically do this if you had a shared resource on, say, a network drive that you want to include but don't want to have a copy in your project. To do this, go to the Add Existing Item dialog box (Shift+Alt+A) and choose Add As Link from the Add drop-down list.


File name: Readme.txt

Now a link to the file is added to your project rather than a copy:



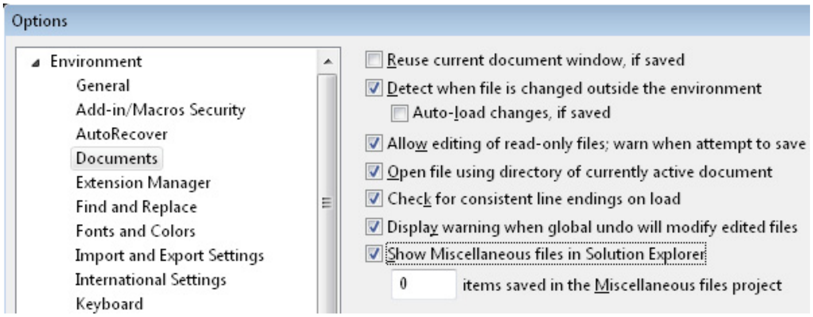
You can tell a file is a link because it has an arrow indicator in the icon, as shown in the preceding graphic—much like the arrow you see on shortcut icons in Windows. The usual caveats apply here, just as they do to any shortcut. For example, for Visual Studio to use the file, you need to make sure that the path to the linked file is accessible.

AX.15 Using the Miscellaneous Files Project

WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | Documents
COMMAND	Tools.Options
VERSIONS	2008,2010
CODE	vstipProj0012

Did you know that Solution Explorer can display a Miscellaneous Files project for files that you do not want to permanently associate with a project or solution? Maybe you still want to track and be able to open certain files quickly—but not associate them with the project or solution. For example, you can create and edit files by using the Visual Studio editors without creating a project. You can also work on files that you want to use temporarily—such as files in which you keep development notes while you work on your solution.

To get the Miscellaneous Files folder to show up, go to Tools | Options | Environment | Documents and select the Show Miscellaneous Files In Solution Explorer option:

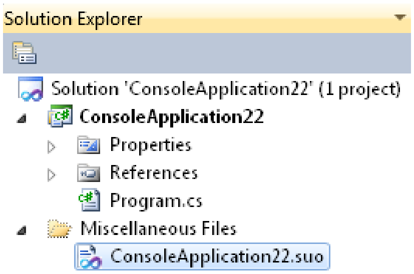


☐ Reuse current document window, if saved
☑ Detect when file is changed outside the environment
  ☐ Auto-load changes, if saved
☑ Allow editing of read-only files; warn when attempt to save
☑ Open file using directory of currently active document
☑ Check for consistent line endings on load
☑ Display warning when global undo will modify edited files
☑ Show Miscellaneous files in Solution Explorer
  0 items saved in the Miscellaneous files project

Pay particular attention to the Items Saved In The Miscellaneous Files Project field, which is initially set to zero. Leaving this value set to zero shows only extra files you currently have open; however, setting it to another number indicates how many files you want to remember at any given time. If you aren't sure about what this value should be, to get started, I suggest setting this value to at least five.

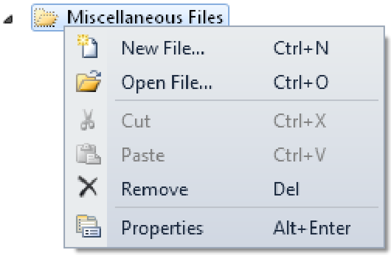
> **Note** The Miscellaneous Files Project does not show up until you open up at least one file that goes in it.

After you open a file that doesn't belong to the solution, the folder shows up, as shown in the following illustration:



With the folder now showing, you can right-click and open, create, or remove files:

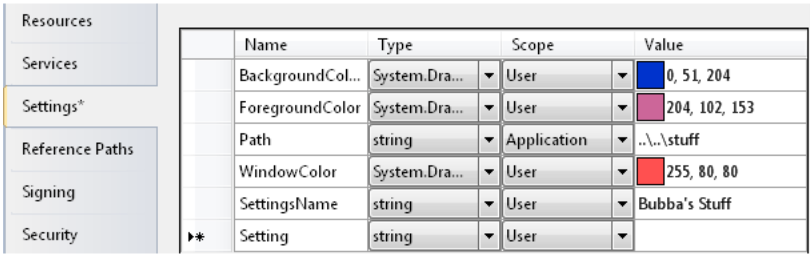
Note Removing files from this folder doesn't delete them permanently.



AX.16 Change the Order of Your Application Settings

VERSIONS	2008,2010
LANGUAGES	C#, VB
CODE	vstipProj0024

When working in your project properties, you might sometimes find yourself using a variety of application settings (I'm assuming you know how to create and use application settings; if not, you can find more information at *http://msdn.microsoft.com/en-us/library/a65txexh.aspx*):

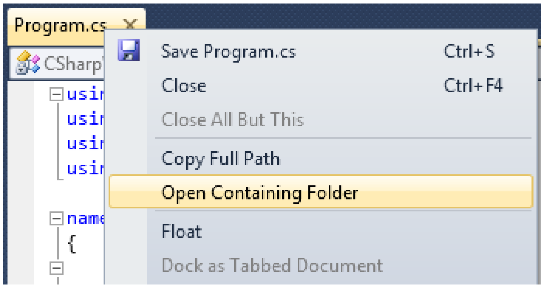


	Name	Type	Scope	Value
	BackgroundCol...	System.Dra...	User	🟦 0, 51, 204
	ForegroundColor	System.Dra...	User	🟪 204, 102, 153
	Path	string	Application	..\..\stuff
	WindowColor	System.Dra...	User	🟥 255, 80, 80
	SettingsName	string	User	Bubba's Stuff
▸*	Setting	string	User	

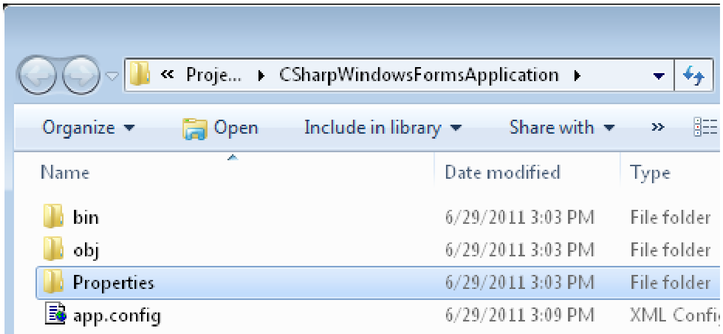
But what if you want to organize these settings? For example, suppose you want the WindowColor setting to appear with the other color settings. You can change the sequence to your preference.

Warning Changing a setting requires you to manually edit your settings file—which could get you into trouble if you make a mistake, so do this at your own risk. Also, make a copy of your settings file before editing it so that you can recover if a problem occurs.

First, save any pending changes, and close the Properties window. Now open up the project location:

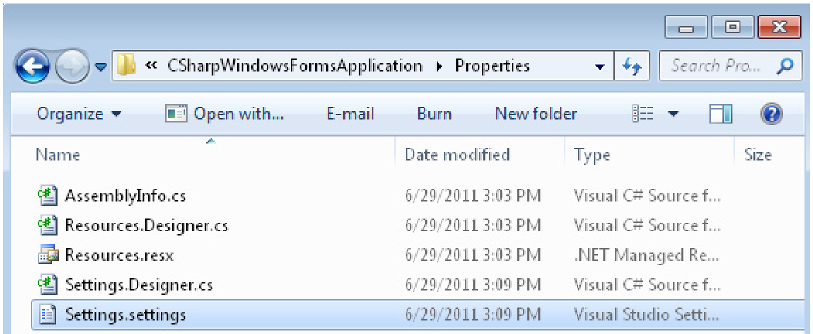


Locate your Properties folder:



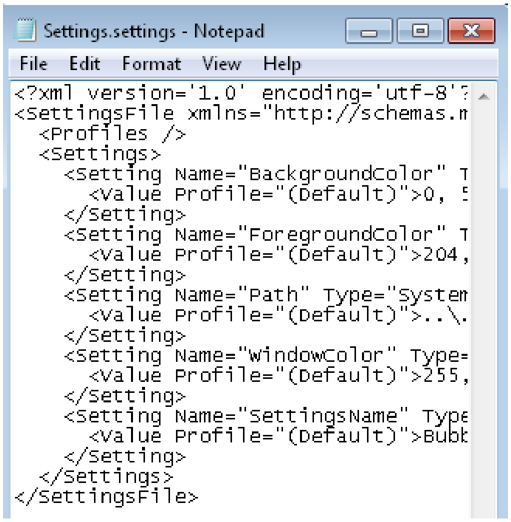
Name	Date modified	Type
bin	6/29/2011 3:03 PM	File folder
obj	6/29/2011 3:03 PM	File folder
Properties	6/29/2011 3:03 PM	File folder
app.config	6/29/2011 3:09 PM	XML Confi

Locate your Settings file, named Settings.settings in this example:



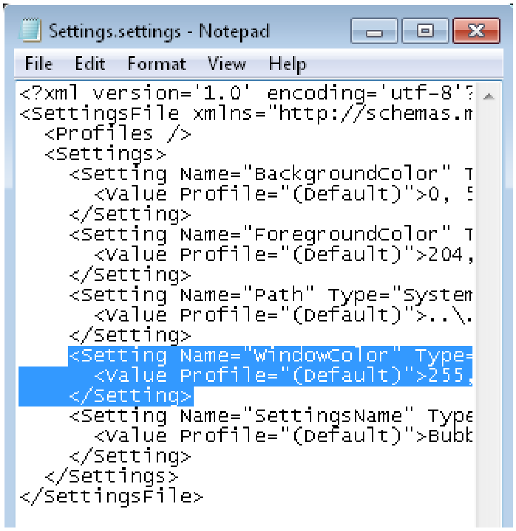
Name	Date modified	Type	Size
AssemblyInfo.cs	6/29/2011 3:03 PM	Visual C# Source f...	
Resources.Designer.cs	6/29/2011 3:03 PM	Visual C# Source f...	
Resources.resx	6/29/2011 3:03 PM	.NET Managed Re...	
Settings.Designer.cs	6/29/2011 3:09 PM	Visual C# Source f...	
Settings.settings	6/29/2011 3:09 PM	Visual Studio Setti...	

Open the Settings.settings file with a plain text editor such as Notepad. Notice that the file is an XML file:



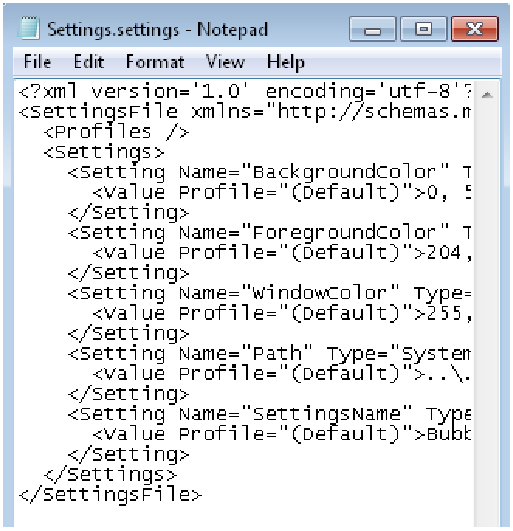
```
<?xml version='1.0' encoding='utf-8'?
<SettingsFile xmlns="http://schemas.m
  <Profiles />
  <Settings>
    <Setting Name="BackgroundColor" T
      <Value Profile="(Default)">0, 5
    </Setting>
    <Setting Name="ForegroundColor" T
      <Value Profile="(Default)">204,
    </Setting>
    <Setting Name="Path" Type="System
      <Value Profile="(Default)">..\.
    </Setting>
    <Setting Name="WindowColor" Type=
      <Value Profile="(Default)">255,
    </Setting>
    <Setting Name="SettingsName" Type
      <Value Profile="(Default)">Bubb
    </Setting>
  </Settings>
</SettingsFile>
```

As you see in the preceding illustration, each setting resides in a <Setting> element. You just need to rearrange the elements in the order you want them. In this case, select the <Setting> element whose name attribute is WindowColor, as shown in the following illustration:



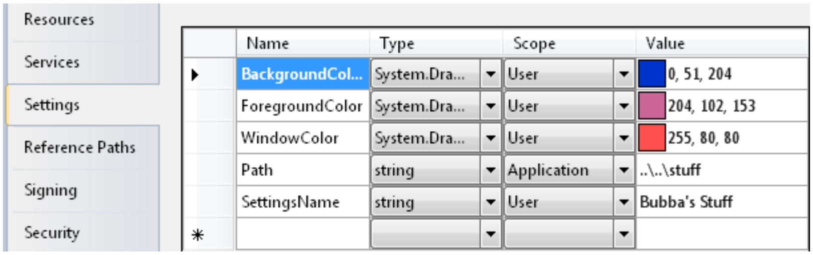
```
<?xml version='1.0' encoding='utf-8'?
<SettingsFile xmlns="http://schemas.m
  <Profiles />
  <Settings>
    <Setting Name="BackgroundColor" T
      <Value Profile="(Default)">0, 5
    </Setting>
    <Setting Name="ForegroundColor" T
      <Value Profile="(Default)">204,
    </Setting>
    <Setting Name="Path" Type="System
      <Value Profile="(Default)">..\.
    </Setting>
    <Setting Name="WindowColor" Type=
      <Value Profile="(Default)">255,
    </Setting>
    <Setting Name="SettingsName" Type
      <Value Profile="(Default)">Bubb
    </Setting>
  </Settings>
</SettingsFile>
```

Next just cut and paste it where you want the setting to show up. For this example, place it just after the ForegroundColor <Setting> element:



```
<?xml version='1.0' encoding='utf-8'?
<SettingsFile xmlns="http://schemas.m
  <Profiles />
  <Settings>
    <Setting Name="BackgroundColor" T
      <Value Profile="(Default)">0, 5
    </Setting>
    <Setting Name="ForegroundColor" T
      <Value Profile="(Default)">204,
    </Setting>
    <Setting Name="WindowColor" Type=
      <Value Profile="(Default)">255,
    </Setting>
    <Setting Name="Path" Type="System
      <Value Profile="(Default)">..\.
    </Setting>
    <Setting Name="SettingsName" Type
      <Value Profile="(Default)">Bubb
    </Setting>
  </Settings>
</SettingsFile>
```

Save your changes and close the file. Now, when you go back into your Project properties, you'll see the settings arranged in their new order:

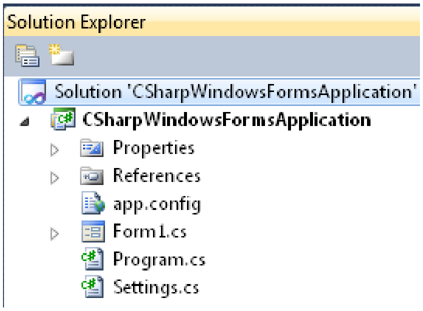


	Name	Type	Scope	Value
▶	BackgroundCol...	System.Dra...	User	🟦 0, 51, 204
	ForegroundColor	System.Dra...	User	🟪 204, 102, 153
	WindowColor	System.Dra...	User	🟥 255, 80, 80
	Path	string	Application	..\..\stuff
	SettingsName	string	User	Bubba's Stuff
*				

AX.17 Hide or Show the Solution File in Solution Explorer

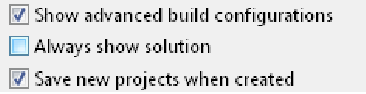
WINDOWS	Alt,T, O
MENU	Tools | Options | Projects and Solutions | General
COMMAND	Tools.Options
VERSIONS	2005,2008,2010
CODE	vstipProj0008

If you don't like seeing the solution file in Solution Explorer, you can easily hide it (or show it if you have it hidden). First, take a look at the default view of Solution Explorer, with the solution file showing:

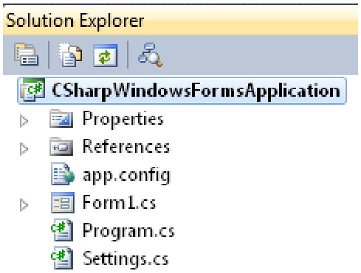

Solution Explorer

Solution 'CSharpWindowsFormsApplication'
- CSharpWindowsFormsApplication
  - Properties
  - References
  - app.config
  - Form1.cs
  - Program.cs
  - Settings.cs

To hide the solution file, select Tools | Options | Projects And Solutions | General, and clear the Always Show Solution option:


☑ Show advanced build configurations
☐ Always show solution
☑ Save new projects when created

The result is shown below:

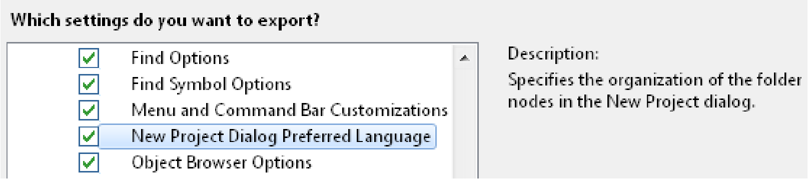


Note This feature works only when you have just one project in the solution; if you have multiple projects in your solution, Visual Studio ignores this setting and shows you the solution node.

AX.18 New Project Dialog Preferred Language

WINDOWS	Alt,T, I
MENU	Tools | Import and Export Settings
COMMAND	Tools.ImportandExportSettings
VERSIONS	2005,2008,2010
CODE	vstipEnv0041

As you're examining items that Visual Studio can export, you might come across the New Project Dialog Preferred Language option, under General Settings:

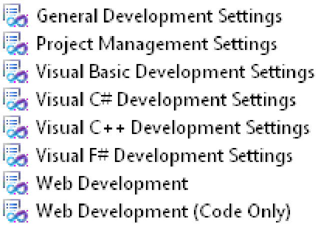

Which settings do you want to export?

- ☑ Find Options
- ☑ Find Symbol Options
- ☑ Menu and Command Bar Customizations
- ☑ New Project Dialog Preferred Language
- ☑ Object Browser Options

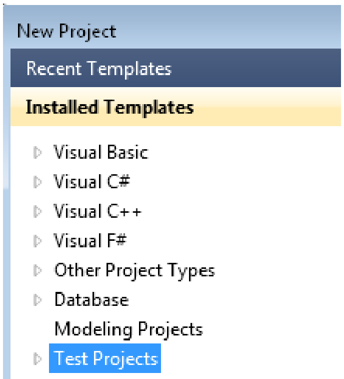
Description:
Specifies the organization of the folder nodes in the New Project dialog.

You might wonder what that means. The best way to explain is to show you a little bit more about the inner workings of Visual Studio.

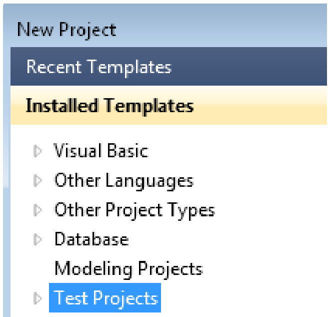
When you installed Visual Studio, you might recall having chosen your preferred development settings:


General Development Settings
Project Management Settings
Visual Basic Development Settings
Visual C# Development Settings
Visual C++ Development Settings
Visual F# Development Settings
Web Development
Web Development (Code Only)

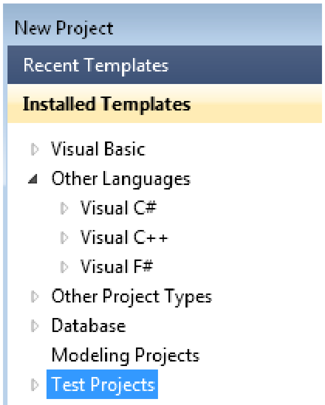
Some of these options are language-specific, and some are not. If you chose General Development Settings, for example, you have no preferred language in the New Project dialog box, so you see a list of languages:



Without a preferred language, the New Project dialog box groups all the language-specific templates into separate nodes, so you can easily select the one you want. However, if you chose a development setting option associated with a particular language, such as VB, you see something different in the New Project dialog box:



As you can see, the other top-level language nodes have disappeared, because you already indicated that Visual Basic is your preferred language. Instead, you'll see an Other Languages node that represents all the other languages:


New Project
Recent Templates
Installed Templates
- Visual Basic
- Other Languages
  - Visual C#
  - Visual C++
  - Visual F#
- Other Project Types
- Database
- Modeling Projects
- Test Projects

So, to come full circle, when you export the New Project Dialog Preferred Language item, it saves this structure for you, which you can then later import.

AX.19 Optimizing Your Project Code

DEFAULT	Alt+Enter (with project selected in Solution Explorer)
VISUAL BASIC 6	Alt+Enter (with project selected in Solution Explorer)
VISUAL C# 2005	Alt+Enter (with project selected in Solution Explorer)
VISUAL C++ 2	Alt+Enter (with project selected in Solution Explorer)
VISUAL C++ 6	Alt+F7; Alt+Enter (with project selected in Solution Explorer)
WINDOWS	Alt,P, P
MENU	Project | [Project Name] Properties
COMMAND	Project.Properties
VERSIONS	2008,2010
LANGUAGES	C#, C++, VB
CODE	vstipProj0014

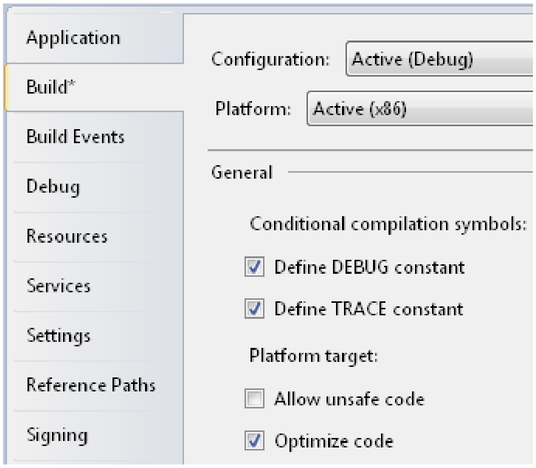
In vstipDebug0032 ("Understanding Just My Code," page A215), we touched on optimization. When optimization is turned off (the default setting for Debug builds), it factors into the code being considered "yours" for the purposes of determining what is "Just My Code." Generally, you won't turn this on for Debug builds. If you do, debug symbols are not generated and you can't step through your code.

When you create a Release build, Visual Studio turns optimization on by default. So what is optimization? According to the documentation, the optimization option “enables or disables optimizations performed by the compiler to make your output file smaller, faster, and more efficient.”

It’s useful to know where to find this option.

C#

In C#, you’ll find the Optimize Code option in the Project properties on the Build tab:



Configuration: Active (Debug)

Platform: Active (x86)

Conditional compilation symbols:

☑ Define DEBUG constant

☑ Define TRACE constant

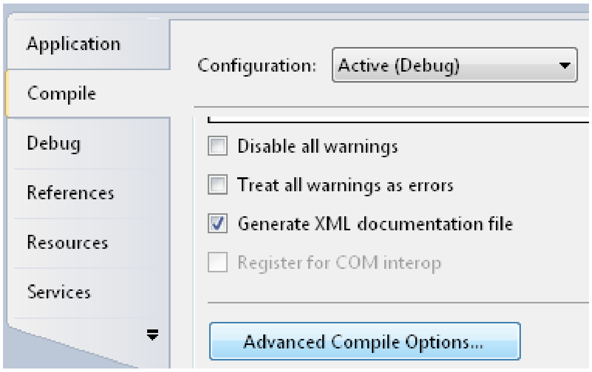
Platform target:

☐ Allow unsafe code

☑ Optimize code

VB

In VB, it is also in the Project properties, but on the Compile tab, and you need to click Advanced Compile Options to find it:



Configuration: Active (Debug)

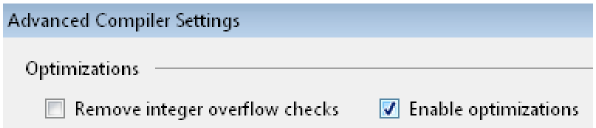
☐ Disable all warnings

☐ Treat all warnings as errors

☑ Generate XML documentation file

☐ Register for COM interop

Next, locate the Enable Optimizations option:



Advanced Compiler Settings

Optimizations

☐ Remove integer overflow checks ☑ Enable optimizations

C++

While VB and C# have only a single option to control optimization, C++ supports a number of more specific optimizations. For example, you can choose to optimize for application speed or for program size. Enabling the optimizations setting from the Project properties enables full optimization (/Ox).

You can get a sense of the full range of /O features in the following list at *http://msdn.microsoft.com/en-us/library/k1ack8f1.aspx*:

- /O1 optimizes code for minimum size.
- /O2 optimizes code for maximum speed.
- /Ob controls inline function expansion.
- /Od disables optimization, speeding compilation and simplifying debugging.
- /Og enables global optimizations.
- /Oi generates intrinsic functions for appropriate function calls.
- /Os tells the compiler to favor optimizations for size over optimizations for speed.
- /Ot (a default setting) tells the compiler to favor optimizations for speed over optimizations for size.
- /Ox selects full optimization.
- /Oy suppresses the creation of frame pointers on the call stack for quicker function calls.

Finally

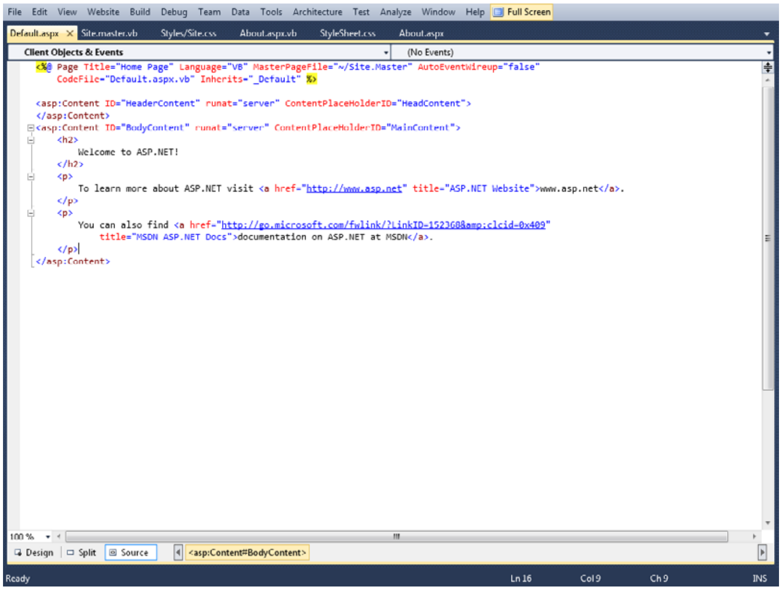
Optimization does a lot of cool things that are great for a shipping application—but not for one you are currently working on and debugging. It's easy to get deep into what the various optimization options actually do; Eric Lippert, of Microsoft, wrote an excellent article on this subject, titled "What Does the Optimize Switch Do," which you can find at *http://blogs.msdn.com/b/ericlippert/archive/2009/06/11/what-does-the-optimize-switch-do.aspx*.

Additional Tips from Chapter 3

AX.20 Full Screen Mode

DEFAULT	Shift+Alt+Enter
VISUAL BASIC 6	Shift+Alt+Enter
VISUAL C# 2005	Shift+Alt+Enter
VISUAL C++ 2	Shift+Alt+Enter
VISUAL C++ 6	Shift+Alt+Enter
VISUAL STUDIO 6	Shift+Alt+Enter
WINDOWS	Alt,V, U
MENU	View | Full Screen
COMMAND	View.FullScreen
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0024

You can quickly switch from any current window state to Full Screen Mode by pressing Shift+Alt+Enter:



```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master" AutoEventWireup="false"
    CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www.asp.net" title="ASP.NET Website">www.asp.net</a>.
    </p>
    <p>
        You can also find <a href="http://go.microsoft.com/fwlink/?LinkID=152368&amp;clcid=0x409"
            title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
    </p>
</asp:Content>
```

By default, this action hides the toolbars and takes up as much of the screen as possible. The advantage, of course, is that you get more real estate to work with when you need more room.

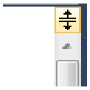
To come out of Full Screen Mode, just press Shift+Alt+Enter again and you are returned to normal view.

AX.21 Split Your Windows Horizontally

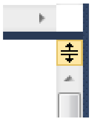
WINDOWS	Alt,W, P (toggle split and remove)
MENU	Window | Split; Window | Remove Split
COMMAND	Window.Split (toggle split and remove)
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0004

In Visual Studio, you can split your windows horizontally. This feature has been available in Microsoft products for quite a while. Just go to Window | Split on the menu bar, or you can use the following mouse technique.

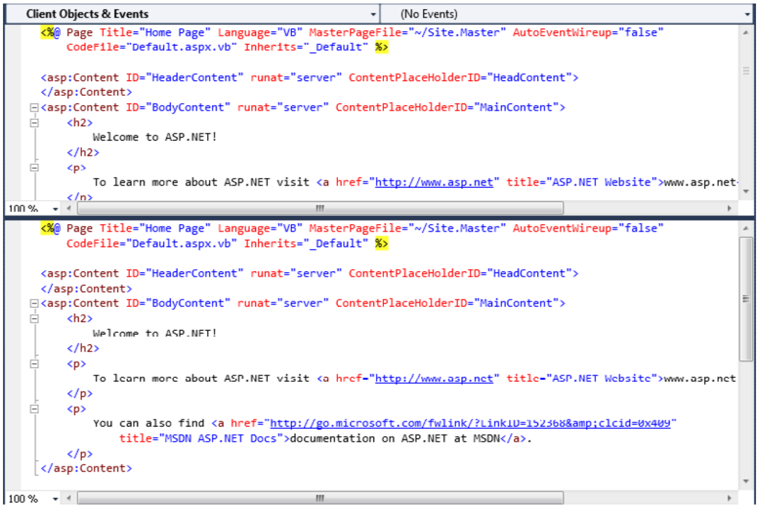
Go the upper-right corner of a document window, and look for the splitter control, as shown in the following illustration:



Click and drag the control down to begin the split process:



Now you have a horizontal split, so you can do things like see different sections of your document at the same time:



```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master" AutoEventWireup="false"
    CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www.asp.net" title="ASP.NET Website">www.asp.net
    </p>
```

```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master" AutoEventWireup="false"
    CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www.asp.net" title="ASP.NET Website">www.asp.net
    </p>
    <p>
        You can also find <a href="http://go.microsoft.com/fwlink/?LinkID=152368&amp;clcid=0x409"
            title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
    </p>
</asp:Content>
```

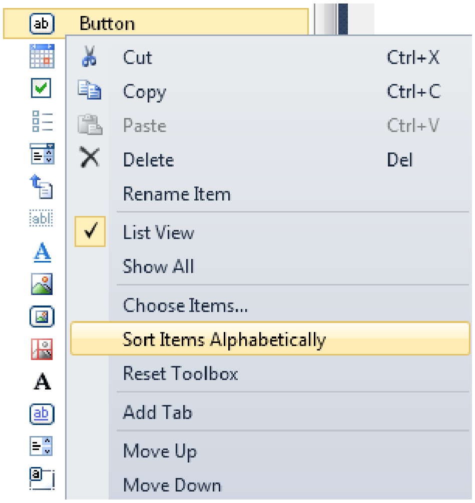
To remove a split, select Window | Remove Split—or just double-click the line separating the two sections.

AX.22 Sorting Items in the Toolbox

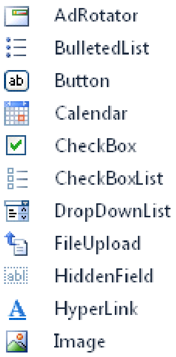
WINDOWS	Shift,F10, O (with the Toolbox selected)
MENU	[Right Click the Toolbox] | Sort Items Alphabetically
COMMAND	Tools.SortItemsAlphabetically
VERSIONS	2005, 2008, 2010
CODE	vstipTool0052

Warning As far as I know, there is no easy way to undo the sorting action other than resetting the Toolbox, so make sure you really want the Toolbox items listed alphabetically before you apply this tip.

If you don't like the default sort order for items in the Toolbox, you can right-click the Toolbox and then choose to sort the items alphabetically:



Selecting this option sorts the items by name, assuming they weren't sorted that way already:

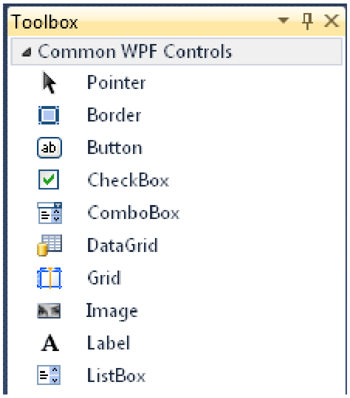


AdRotator
BulletedList
Button
Calendar
CheckBox
CheckBoxList
DropDownList
FileUpload
HiddenField
HyperLink
Image

AX.23 Icon vs. List View in the Toolbox

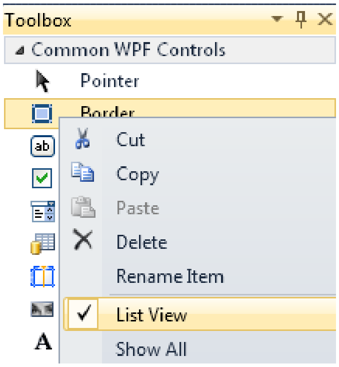
WINDOWS	Shift,F10, L (with the Toolbox selected will toggle List View)
MENU	[Right Click the Toolbox] | List View (toggle)
COMMAND	Tools.ListView
VERSIONS	2005, 2008, 2010
CODE	vstipTool0053

The default view of items in the Toolbox is List View, as shown in the following illustration:

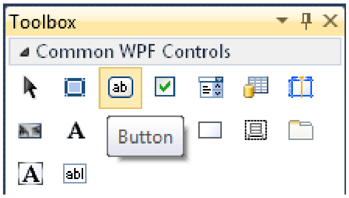


Toolbox
Common WPF Controls
Pointer
Border
Button
CheckBox
ComboBox
DataGrid
Grid
Image
Label
ListBox

You can change this to the Icon View if you prefer, by right-clicking the Toolbox and clicking List View, which turns off the List View option:

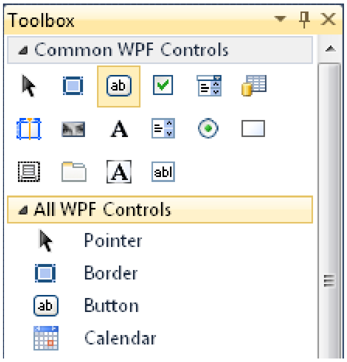


This produces the result shown in the following illustration:



To get the List View back again, right-click in Icon View, and click List View again.

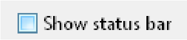
This option applies only on the current Toolbox tab, so you can have your tabs organized differently based on your preference:



AX.24 Hide the Status Bar

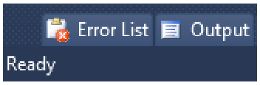
WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0024

Removing the Status Bar gives you a little extra space at the bottom of your screen. Without it you cannot get status messages, so remove it only if you are sure you don't need it. To remove the Status Bar, go to Tools | Options | Environment | General and clear the Show Status Bar option:

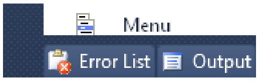


☐ Show status bar

Before (status bar is where the word "Ready" is located):



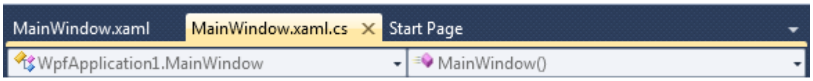
After (no status bar):



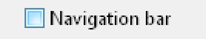
AX.25 Remove the Navigation Bar

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | [All Languages or Specific Language]
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0028

If you find you don't use the navigation bar, shown in the following illustration, and want a little extra space, how can you make it go away?

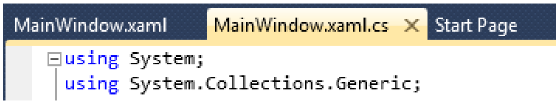


Just go to Tools | Options | Text Editor | [All Languages or Specific Language], and clear the Navigation Bar check box:



☐ Navigation bar

The following illustration shows the result:

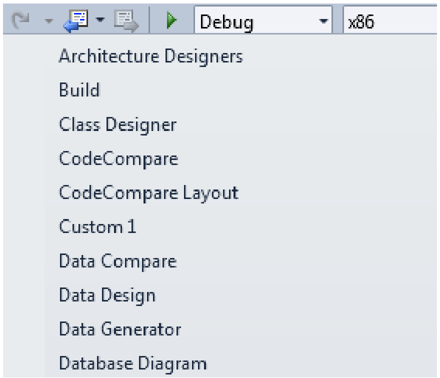


```
using System;
using System.Collections.Generic;
```

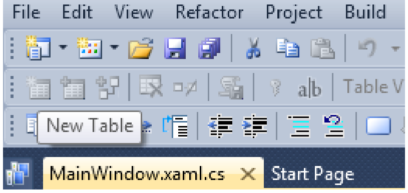
AX.26 Show Any Toolbar

WINDOWS	Alt,V, T, [Up or Down] Arrow
MENU	View | Toolbars | [Toolbar]
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0025

You can show any toolbar at any time. Just right-click an existing toolbar, and left-click the one you want to see:



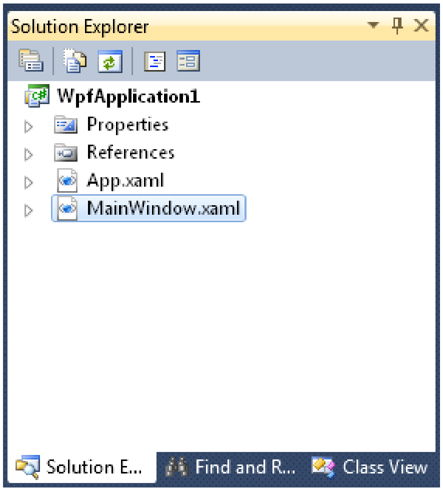
Not all toolbars will have their buttons available, because it depends on your context. For example, the Database Diagram Toolbar (shown in the following illustration) buttons are not available until you are actually working on a diagram:



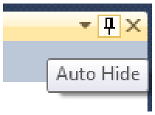
AX.27 Changing Auto-Hide Behavior for Tool Windows

WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0035

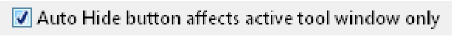
By default, you could easily have several tool windows in a tab group, as shown here:



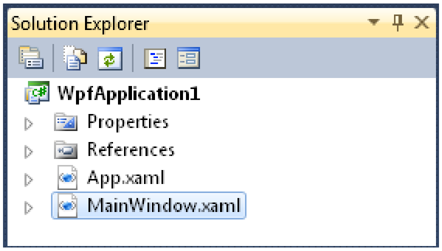
So to clear your workspace, you click the auto-hide button:



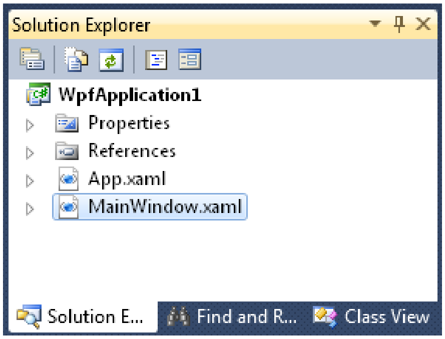
Now all the tool windows are hidden together. Next you can go to Tools | Options | Environment | General and select the Auto Hide Button Affects Active Tool Window Only option:


☑ Auto Hide button affects active tool window only

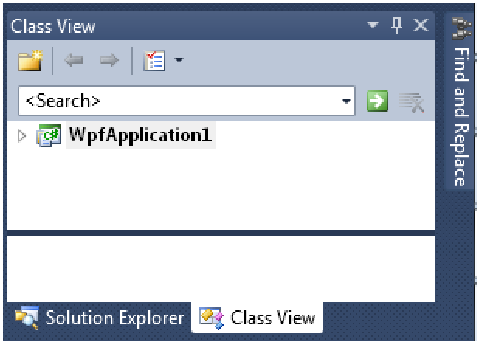
Now you no longer get tabs at the bottom when you interact with a tool window:



The tool windows have to be docked individually to get the tabs again:



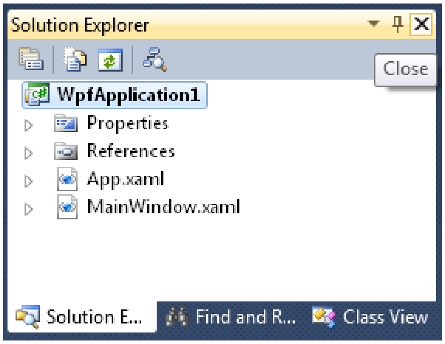
Now, when you auto-hide a window, only that window is hidden; the rest of the tab group remains visible:



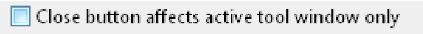
AX.28 Closing a Tool Window Tab Group

WINDOWS	Shift,Esc
MENU	Tools | Options | Environment | General
COMMAND	Window.CloseToolWindow
VERSIONS	2005, 2008, 2010
CODE	vstipTool0040

By default, when you click the Close button for a tool window in a tab group, it normally closes only the current tool window:



However, you can change this behavior by going to Tools | Options | Environment | General and clearing the Close Button Affects Active Tool Window Only check box, as shown in the following illustration:


☐ Close button affects active tool window only

Now, closing any one tool window in the tab group causes all tool windows in that group to be closed.

AX.29 Copy and Paste with the Command Prompt

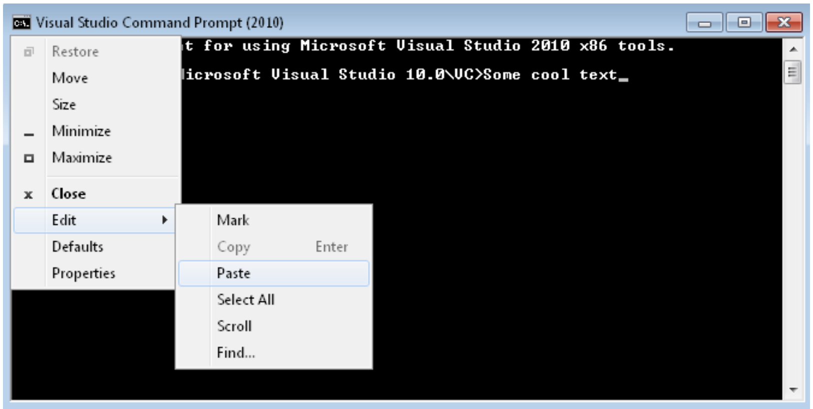
WINDOWS	Enter (copy)
MENU	System Menu | Edit | [Mark, Copy, Paste, Select All]
CODE	vstipTool0057

In a previous tip, we looked at how to work with the command prompt history. In this tip, we examine how to copy and paste text to and from the Command Prompt window.

Pasting

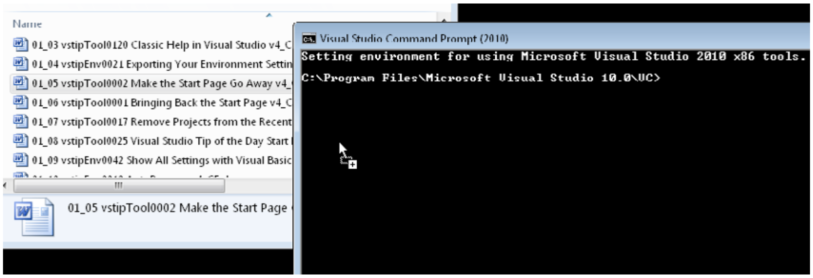
Visual Studio offers a rich set of tools to help you work with text in the command prompt windows. For example, you can copy text and paste it by going to System Menu | Edit | Paste, as shown in the following illustration.

Note The System menu is the icon in the upper-left corner of the window.



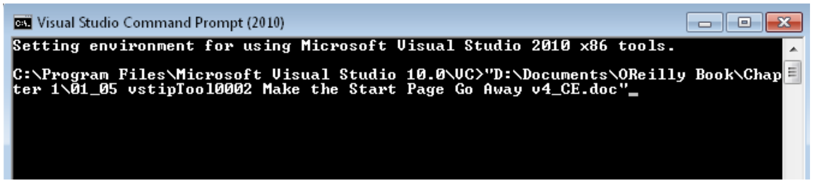
```
t for using Microsoft Visual Studio 2010 x86 tools.
licrosoft Visual Studio 10.0\VC>Some cool text_
```

If you have a folder or file path you want to use, you can always find the folder or file in Windows Explorer and drag the item to the command prompt, as shown in the following illustration:


Name
01_03 vstipTool0120 Classic Help in Visual Studio v4_C
01_04 vstipEnv0021 Exporting Your Environment Settin
01_05 vstipTool0002 Make the Start Page Go Away v4_
01_06 vstipTool0001 Bringing Back the Start Page v4_C
01_07 vstipTool0017 Remove Projects from the Recent
01_08 vstipTool0025 Visual Studio Tip of the Day Start
01_09 vstipEnv0042 Show All Settings with Visual Basic
01_05 vstipTool0002 Make the Start Page
Visual Studio Command Prompt (2010)

```
Setting environment for using Microsoft Visual Studio 2010 x86 tools.
C:\Program Files\Microsoft Visual Studio 10.0\VC>
```

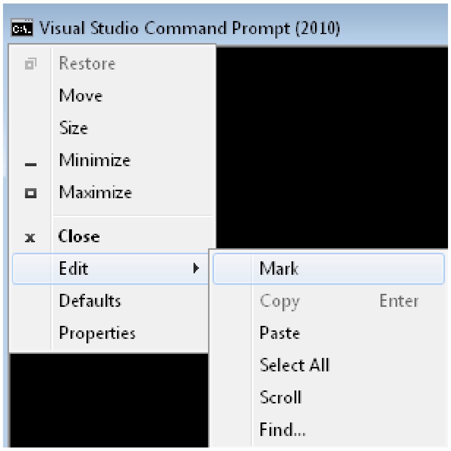
The preceding action pastes the full path to the folder or file in the window:



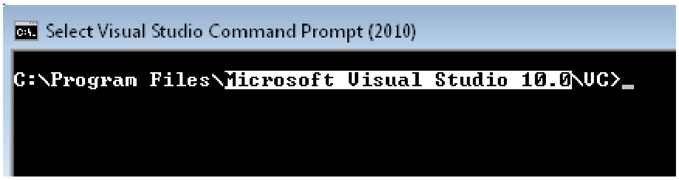
```
Setting environment for using Microsoft Visual Studio 2010 x86 tools.

C:\Program Files\Microsoft Visual Studio 10.0\VC>"D:\Documents\OReilly Book\Chap
ter 1\01_05 vstipTool0002 Make the Start Page Go Away v4_CE.doc"_
```

Copying

You can copy text from the command prompt by marking the text. Go to System Menu | Edit | Mark, as shown in the following illustration:

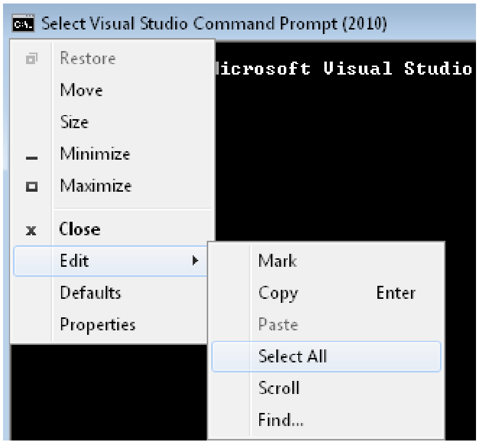


Highlight the text you want by clicking and dragging over that text:



```
C:\Program Files\Microsoft Visual Studio 10.0\VC>_
```

If you want, you can always go to System Menu | Edit | Select All:



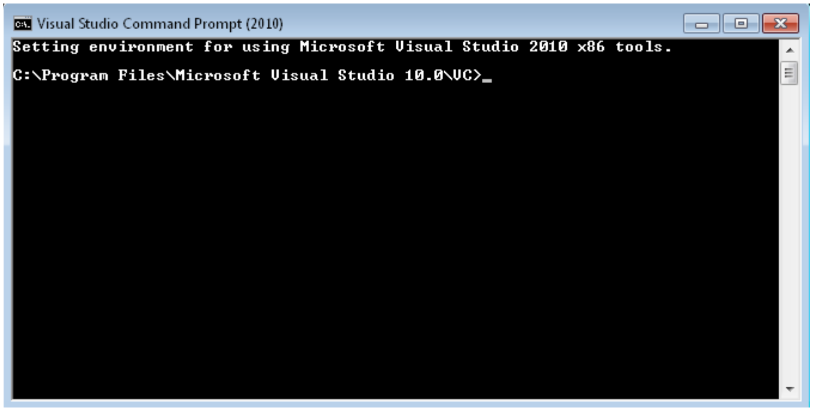
```
icrosoft Visual Studio
```

Press Enter to copy the text to the clipboard, and then paste the text where you want.

AX.30 Customize the Command Prompt

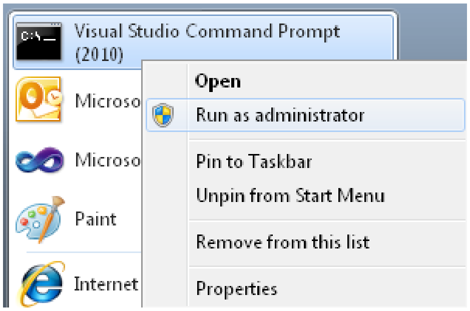
WINDOWS	Alt,Space, P
MENU	System Menu | Properties
CODE	vstipTool0058

If you are going to use the command prompt, you might want to customize the look and feel:

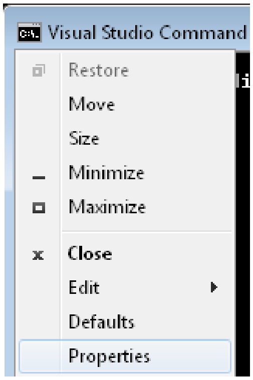


```
Setting environment for using Microsoft Visual Studio 2010 x86 tools.

C:\Program Files\Microsoft Visual Studio 10.0\VC>_
```

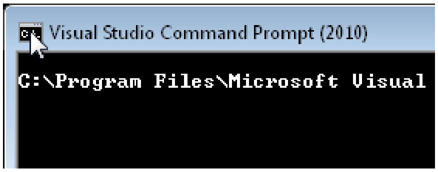
Note If you want to customize the Command window, you need to run it as Administrator. To do this, just right-click the command prompt icon and choose Run As Administrator:



Let's review some basic settings you might want to take advantage of early on. First go to System Menu | Properties:



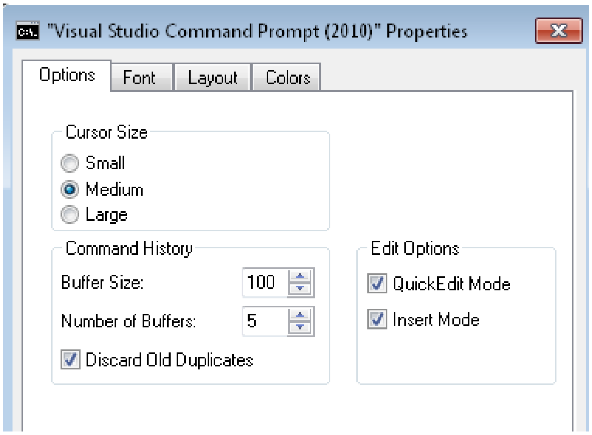
Note Just as a reminder, the System menu is the icon at the upper-left corner of the window:



```
C:\Program Files\Microsoft Visual
```

Options

On the Options tab, notice that we have several items of interest:



Cursor Size

○ Small
● Medium
○ Large

Command History

Buffer Size: 100
Number of Buffers: 5
☑ Discard Old Duplicates

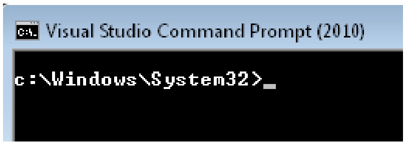
Edit Options

☑ QuickEdit Mode
☑ Insert Mode

Cursor Size

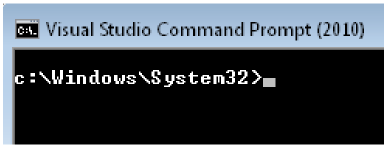
Represents the size of the flashing cursor.

Small



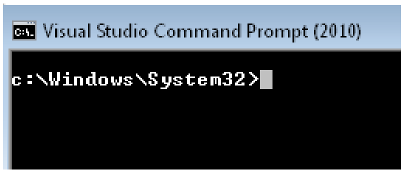
```
c:\Windows\System32>_
```

Medium



```
c:\Windows\System32>
```

Large



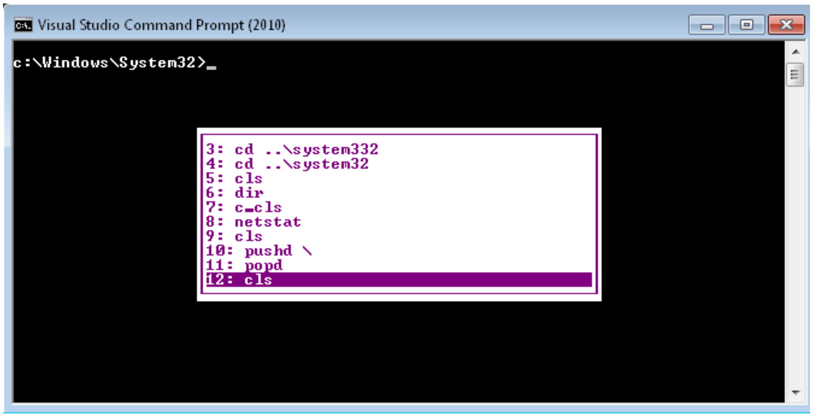
```
c:\Windows\System32>█
```

Command History

Deals with how many commands are remembered by setting the buffer size (number of commands to remember) times the number of buffers. The default is 200 commands, but this value can be changed to 999 for each value for a max total of 998,001 commands that can be remembered. I usually set Buffer Size to 100 and Number Of Buffers to 5, for a total of 500 commands remembered.

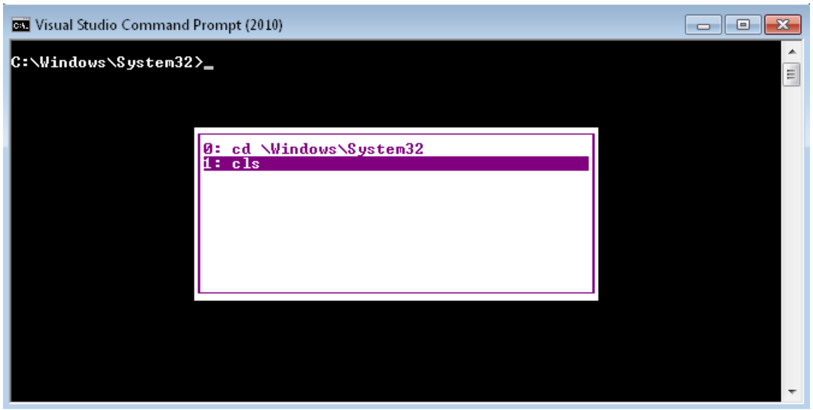
I usually select the Discard Old Duplicates option. Some people leave this off for running the same command several times, so you might want to leave this off in some cases. Selecting this option removes duplicates so that they don't show up several times in the command history.

Before:



```
c:\Windows\System32>_

3: cd ..\system332
4: cd ..\system32
5: cls
6: dir
7: c▬cls
8: netstat
9: cls
10: pushd \
11: popd
12: cls
```

After:

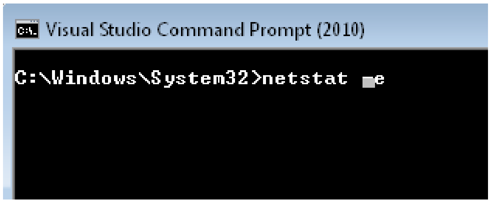


```
C:\Windows\System32>_

0: cd \Windows\System32
1: cls
```

Edit Options

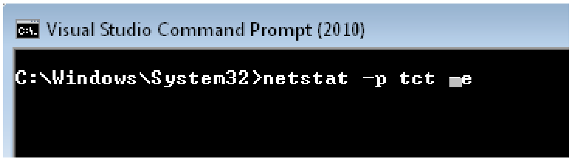
Define how you can do certain actions. Quick Edit Mode allows you to use your mouse to work with text in the window, so I suggest you turn it on. Insert Mode is on by default and simply means that if you move the cursor into a command (using the arrow keys), you can type and it inserts (instead of overwrites) the text after it.

Before:



```
C:\Windows\System32>netstat _e
```

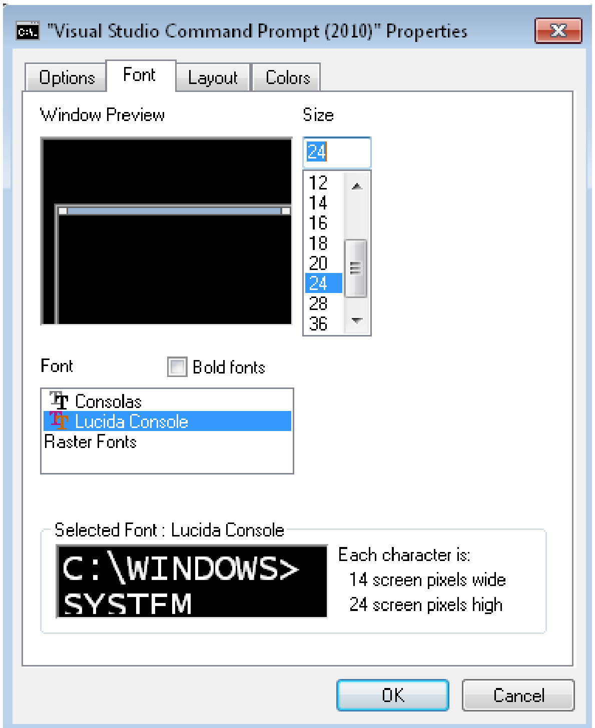
After:



```
C:\Windows\System32>netstat -p tct _e
```

Font

The Font tab is pretty self-explanatory. It allows you to pick various font properties for the command window. Play with these to suit your needs:



Size: 24

Font: Lucida Console

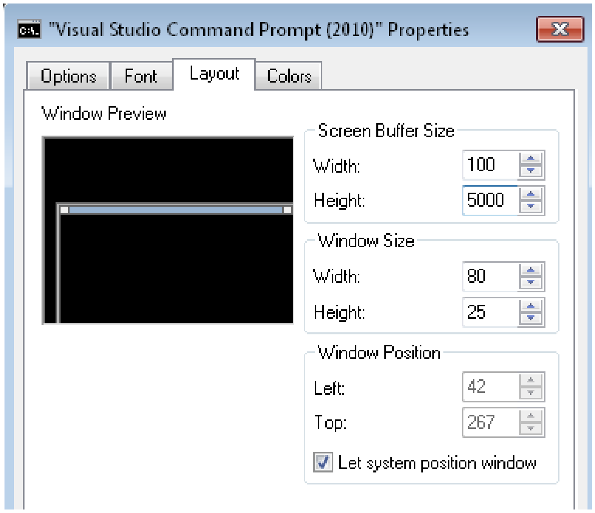
☐ Bold fonts

Selected Font : Lucida Console

Each character is:
14 screen pixels wide
24 screen pixels high

Layout

The Layout tab is very important. Let's look at some of these options:



Screen Buffer Size
Width: 100
Height: 5000

Window Size
Width: 80
Height: 25

Window Position
Left: 42
Top: 267

☑ Let system position window

Screen Buffer Size

Allows you to set how wide and tall you want the window information to be. Width should be less than the window size width you anticipate. Because I tend to maximize my command windows, I set this to 100. Height is how many lines you want to be able to scroll back to. I set this to 5000 just because I like large buffer sizes. Most folks tend to set this to around 1000 or so.

Window Size

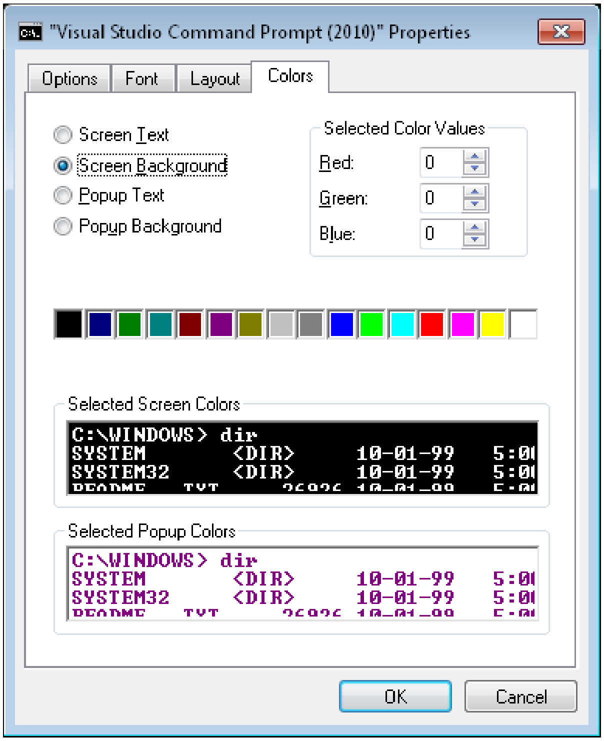
Sets the initial size of the command window. You can play with these settings to find a suitable size for you. Because I maximize my command windows, I just leave this setting alone.

Window Position

The position you want the window to start in. I would leave this setting be, but if you want to change it, set Left and Top to 0 and then increase from there to suit your taste.

Colors

The Colors tab allows you to change two major aspects of the color scheme for the command window: one for the regular window and one for the pop-up window you get when you press F7:


"Visual Studio Command Prompt (2010)" Properties
Options
Font
Layout
Colors
Selected Screen Colors
C:\WINDOWS> dir
SYSTEM <DIR> 10-01-99 5:0
SYSTEM32 <DIR> 10-01-99 5:0
Selected Popup Colors
C:\WINDOWS> dir
SYSTEM <DIR> 10-01-99 5:0
SYSTEM32 <DIR> 10-01-99 5:0
OK
Cancel

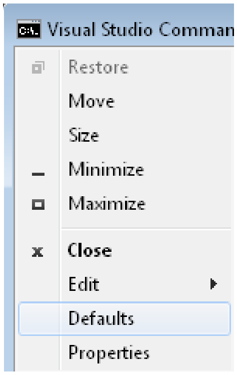
○ Screen Text
● Screen Background
○ Popup Text
○ Popup Background

Selected Color Values
Red: 0
Green: 0
Blue: 0

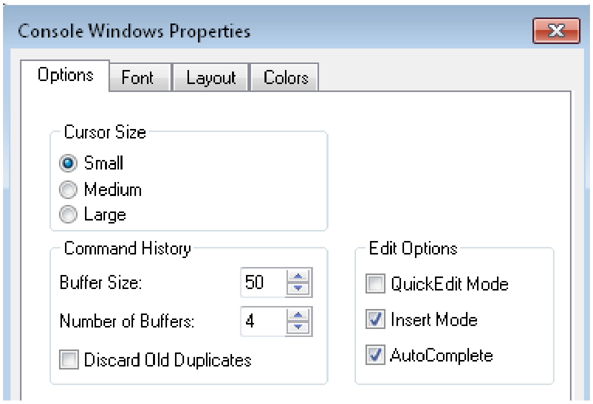
These settings are completely personal preferences, so try adjusting them and looking at the previews at the bottom to determine what schemes suit you. I tend to favor the old school “green screen” colors on my Command windows (green text with a black background).

Resetting Back to Defaults

If you totally mess up your settings, you can always get all the default settings back by going to System Menu | Defaults:



After clicking Default, you see the properties with *all tabs set to their default settings*. Just click OK, and you have all your defaults back:



Cursor Size
● Small
○ Medium
○ Large

Command History
Buffer Size: 50
Number of Buffers: 4
☐ Discard Old Duplicates

Edit Options
☐ QuickEdit Mode
☑ Insert Mode
☑ AutoComplete

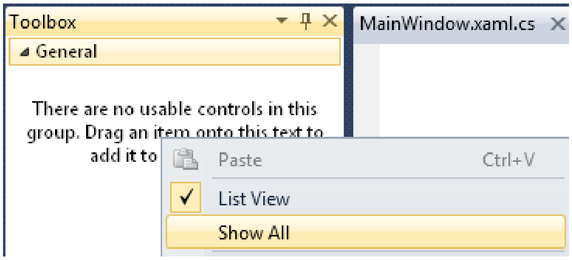
Note When you restore the default settings, notice the AutoComplete option that wasn't enabled before. This option allows for tab completion of file names. (See vstipTool0056, "Command Prompt Tab Completion," page 105.) You should always leave this setting on.

AX.31 Show All Toolbox Controls

WINDOWS	Shift,F10, S (with Toolbox selected)
MENU	Right-click | Show All
COMMAND	Tools.ShowAll
VERSIONS	2005, 2008, 2010
CODE	vstipTool0060

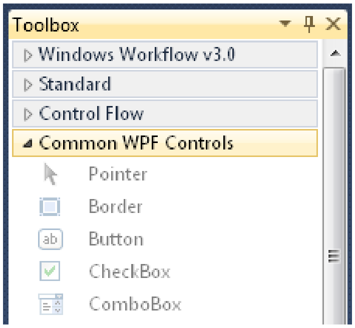
Sometimes when you install controls for the Toolbox, you might not see the controls because they are in the wrong context. For example, a WPF control wouldn't show up in the Toolbox while you are typing code. When the controls don't show up, you might think the add-in failed to install the Toolbox controls. This tip shows how you can confirm everything installed in the Toolbox.

To see all the controls that are installed, right-click the Toolbox and select Show All, as shown in the following illustration:



There are no usable controls in this group. Drag an item onto this text to add it to

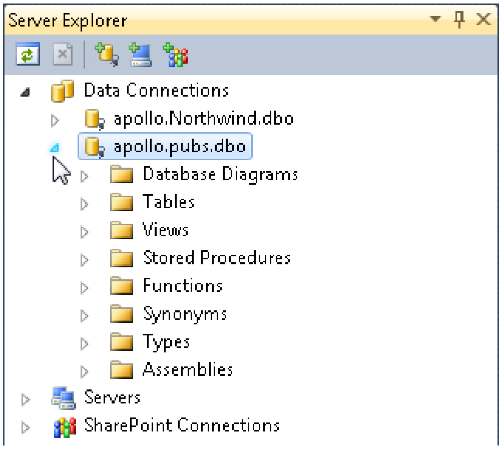
Now all the controls are visible, regardless of context:



AX.32 Server Explorer: Data Connections

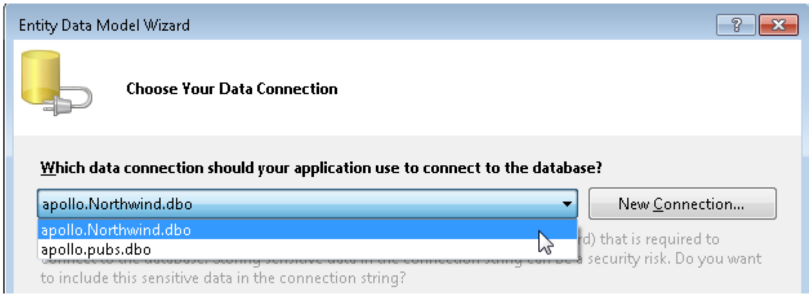
DEFAULT	Ctrl+Alt+S (view server explorer)
VISUAL BASIC 6	Ctrl+Alt+S (view server explorer)
VISUAL C# 2005	Ctrl+Alt+S; Ctrl+W, L; Ctrl+W, Ctrl+L (view server explorer)
VISUAL C++ 2	Ctrl+Alt+S (view server explorer)
VISUAL C++ 6	Ctrl+Alt+S (view server explorer)
VISUAL STUDIO 6	Ctrl+Alt+S (view server explorer)
WINDOWS	Alt,V, V (view server explorer); Alt,T, D (connect to database)
MENU	View | Server Explorer; Tools | Connect to Database
COMMAND	View.ServerExplorer; Tools.ConnecttoDatabase
VERSIONS	2005, 2008, 2010
CODE	vstipTool0121

Server Explorer (Ctrl+Alt+S) is the server management tool window that comes with Visual Studio. One of the things you can use this window for is to open data connections:



Data Connections in Other Areas

The data connections you have in Server Explorer can be used in other areas, such as ADO.NET Entity Data Models:



Which data connection should your application use to connect to the database?

apollo.Northwind.dbo

apollo.Northwind.dbo
apollo.pubs.dbo

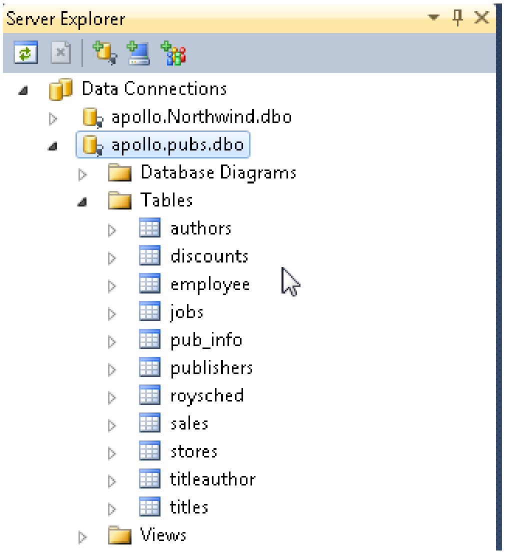
d) that is required to
security risk. Do you want
to include this sensitive data in the connection string?

When you go to create Entity Data Models that are generated from a database, you can choose from the data connections you already have or you can create a new connection to be added to the list.

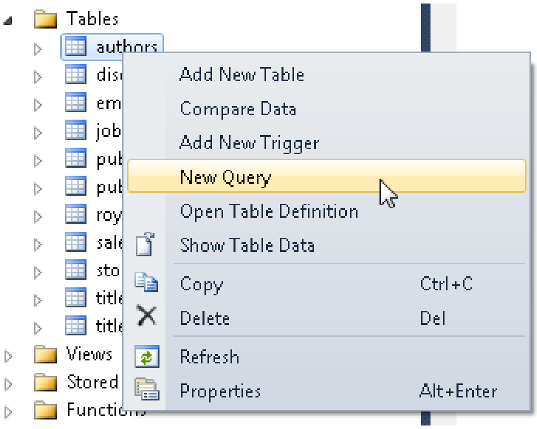
Data Connections in Server Explorer

There is an incredible amount of power and control that is available here. Let's look at a couple of examples.

Tables

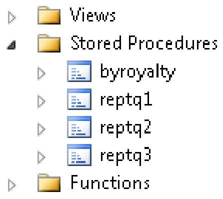


At the most basic level, you can list out the tables in the database. Additionally, you can right-click any table and see most of the features you can leverage, as shown in the following illustration:

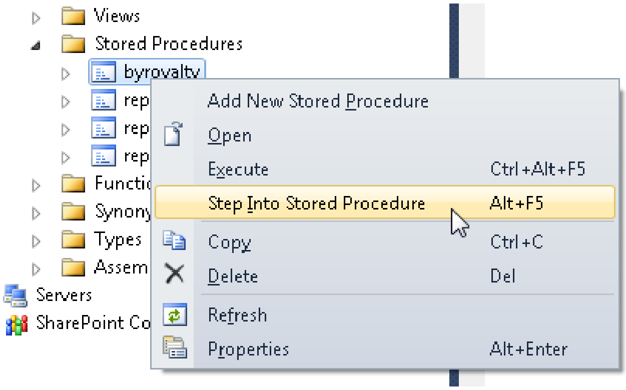


Notice that you can add a new table, create a new query, or just show the data in the table, among other tasks.

Stored Procedures



One of the great database features in Visual Studio is that not only can you create and edit stored procedures, but you can step into them as well:



Finally

Don’t take the Database Connection features in Server Explorer for granted. These features make a great deal of power available to you, and you should invest some time to see whether these are helpful for your work.

AX.33 Server Explorer: Server Event Logs

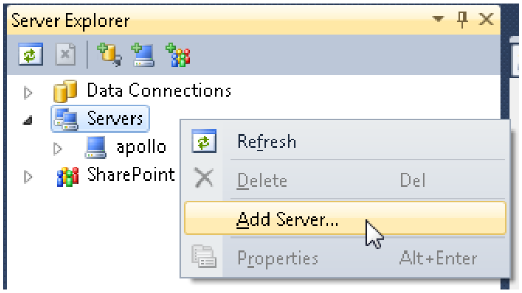
DEFAULT	Ctrl+Alt+S
VISUAL BASIC 6	Ctrl+Alt+S
VISUAL C# 2005	Ctrl+Alt+S; Ctrl+W, L; Ctrl+W, Ctrl+L
VISUAL C++ 2	Ctrl+Alt+S
VISUAL C++ 6	Ctrl+Alt+S
VISUAL STUDIO 6	Ctrl+Alt+S
WINDOWS	Alt,V, V
MENU	View | Server Explorer
COMMAND	View.ServerExplorer
VERSIONS	2005, 2008, 2010
CODE	vstipTool0122

Most people believe that Server Explorer is the tool window we use for data connections (see vstipTool0121, "Server Explorer Data Connections," page A58), and they assume that's pretty much where the experience ends.

This unsung hero really does a lot more if you let it. For example, the Servers section comes with a power that you might not even know existed—working with servers.

Adding Servers

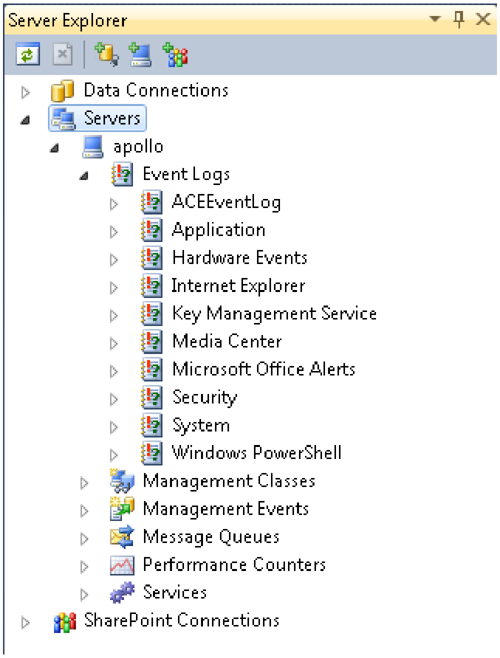
To get started, Server Explorer comes with the local machine already in the list of Servers, plus you can add additional servers as needed:



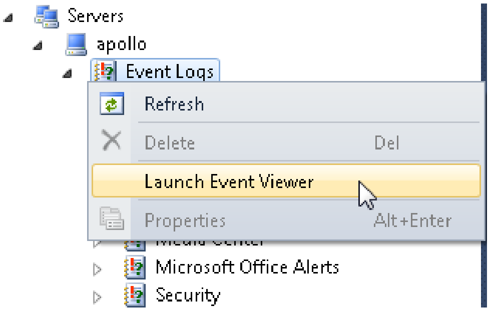
Don't be fooled by the term "Server"; it really means any computer that you want to connect to for information. In these examples, my "server" is a Microsoft Windows 7 Professional–based machine.

Event Logs

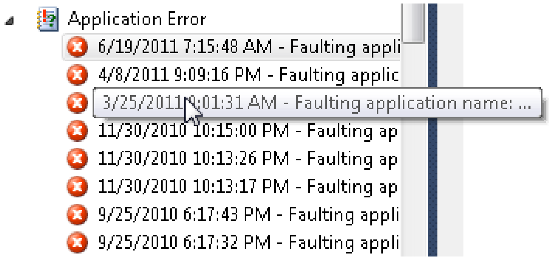
After you have the server that you want, you can start working with features. The Event Logs section is a perfect example. The following illustration shows what is on my machine for Event Logs in Server Explorer:



You can use this area to start the Event Viewer if you need it:



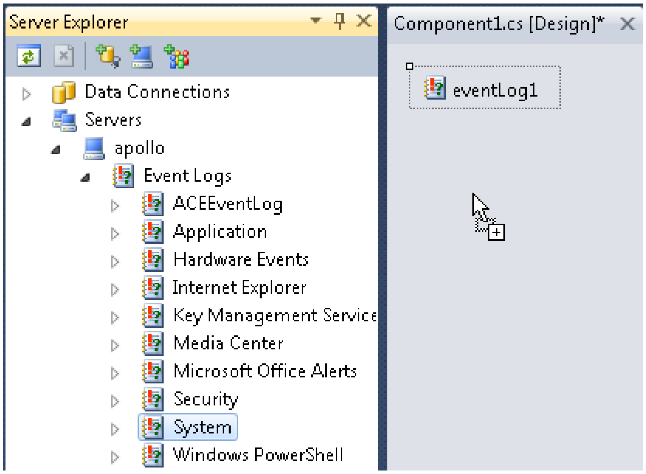
Or you can take a quick view of events in any of the various categories by drilling down into the tree:


Application Error
6/19/2011 7:15:48 AM - Faulting appli
4/8/2011 9:09:16 PM - Faulting applic
3/25/2011 9:01:31 AM - Faulting application name: ...
11/30/2010 10:15:00 PM - Faulting ap
11/30/2010 10:13:26 PM - Faulting ap
11/30/2010 10:13:17 PM - Faulting ap
9/25/2010 6:17:43 PM - Faulting appli
9/25/2010 6:17:32 PM - Faulting appli

Click and Drag Components

Event log components can even be dragged onto a Windows form or component class design surface so that you can manipulate them:

Note For more information, see the topic "How to: Add Items from Server Explorer," at *http://msdn.microsoft.com/en-us/library/84s2c1k0.aspx*.



When the component is in place, you can write code to perform various actions. In the following example, an entry is written to the event log:

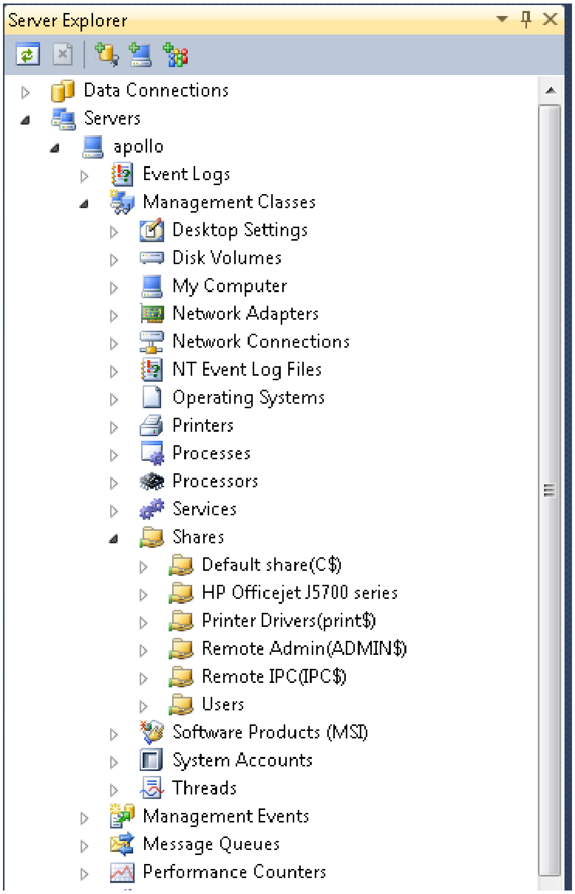
```
public void WriteToLog()
{
    eventLog1.WriteEntry("Log entry!");
}
```

This is a great way to have event log information available to the user. Taking time to learn Server Explorer can definitely bring you benefits in your daily work.

AX.34 Server Explorer: Server Management Classes

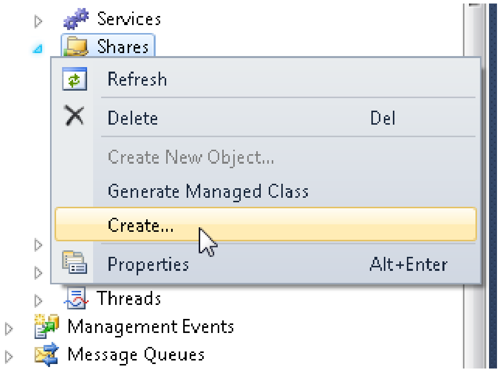
DEFAULT	Ctrl+Alt+S
VISUAL BASIC 6	Ctrl+Alt+S
VISUAL C# 2005	Ctrl+Alt+S; Ctrl+W, L; Ctrl+W, Ctrl+L
VISUAL C++ 2	Ctrl+Alt+S
VISUAL C++ 6	Ctrl+Alt+S
VISUAL STUDIO 6	Ctrl+Alt+S
WINDOWS	Alt,V, V
MENU	View | Server Explorer
COMMAND	View.ServerExplorer
VERSIONS	2005, 2008, 2010
CODE	vstipTool0123

The management classes in Server Explorer give you a view into the system. For example, you can use management classes to see network shares, as shown in the following illustration:

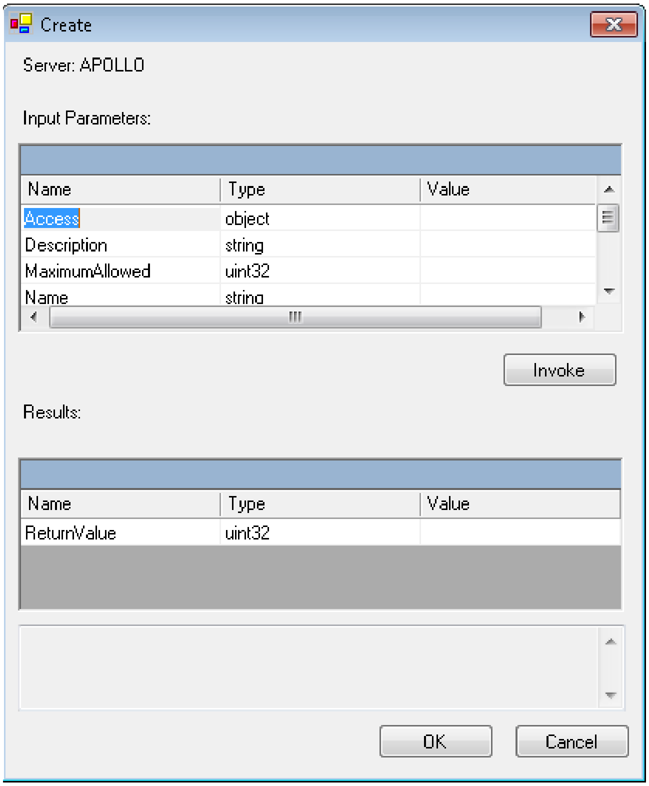


Create

You can also right-click a node and choose Create from the menu to make a new instance:



Unfortunately, you get a really ugly and unfriendly dialog box to use when you do this, as shown in the following illustration:



Server: APOLLO

Input Parameters:

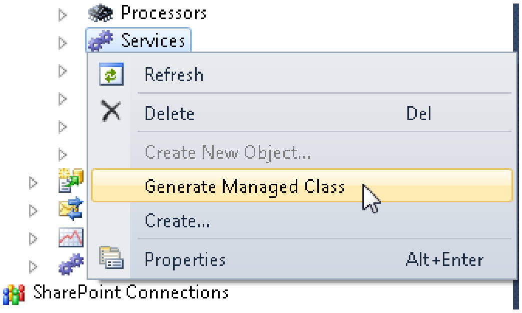
Name	Type	Value
Access	object	
Description	string	
MaximumAllowed	uint32	
Name	string	

Results:

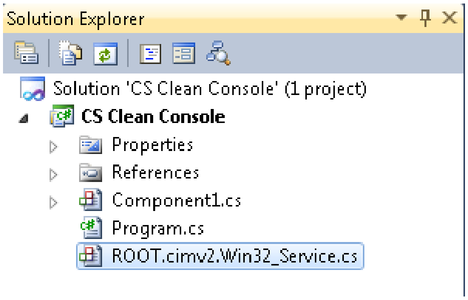
Name	Type	Value
ReturnValue	uint32	

Generate Classes

One better way to work with the management classes is to right-click one and then generate a class from the menu:



This gives you a class you can use as you see fit:

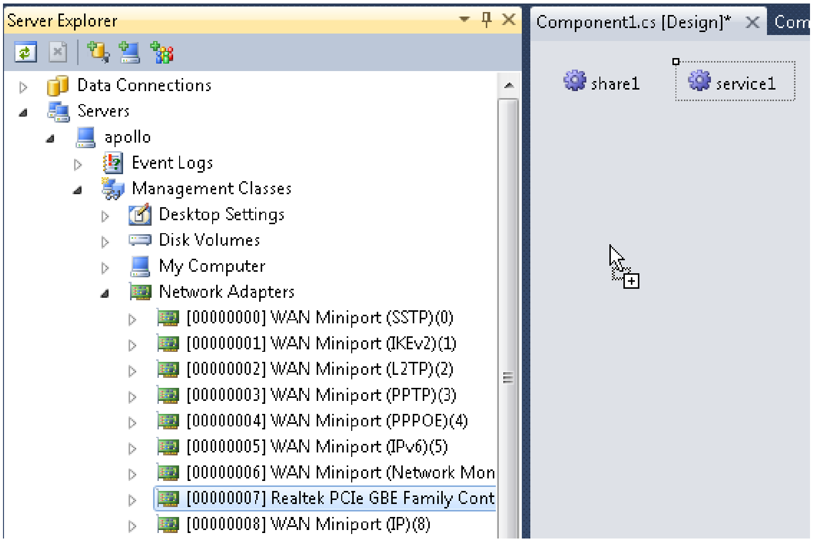


```
// Private property to hold the WMI namespace in which the class resides.
private static string CreatedWmiNamespace = "ROOT\\cimv2";

// Private property to hold the name of WMI class which created this class.
private static string CreatedClassName = "Win32_Service";
```

Click and Drag Components

The management classes can also be dragged onto a Windows form or component class design surface so that you can manipulate them.

Note For more information, see the topic "How to: Add Items from Server Explorer," at *http://msdn.microsoft.com/en-us/library/84s2c1k0.aspx*.

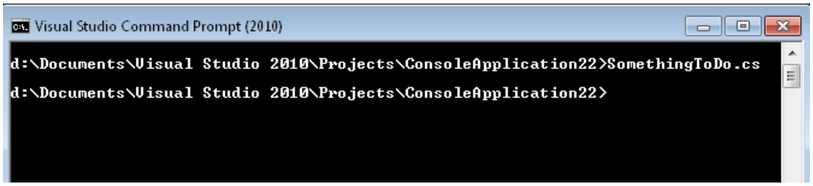

Server Explorer
Data Connections
Servers
apollo
Event Logs
Management Classes
Desktop Settings
Disk Volumes
My Computer
Network Adapters
[00000000] WAN Miniport (SSTP)(0)
[00000001] WAN Miniport (IKEv2)(1)
[00000002] WAN Miniport (L2TP)(2)
[00000003] WAN Miniport (PPTP)(3)
[00000004] WAN Miniport (PPPOE)(4)
[00000005] WAN Miniport (IPv6)(5)
[00000006] WAN Miniport (Network Mon
[00000007] Realtek PCIe GBE Family Cont
[00000008] WAN Miniport (IP)(8)
Component1.cs [Design]*
Com
share1
service1

AX.35 Window Layouts: File View

VERSIONS	2005, 2008, 2010
CODE	vstipEnv0053

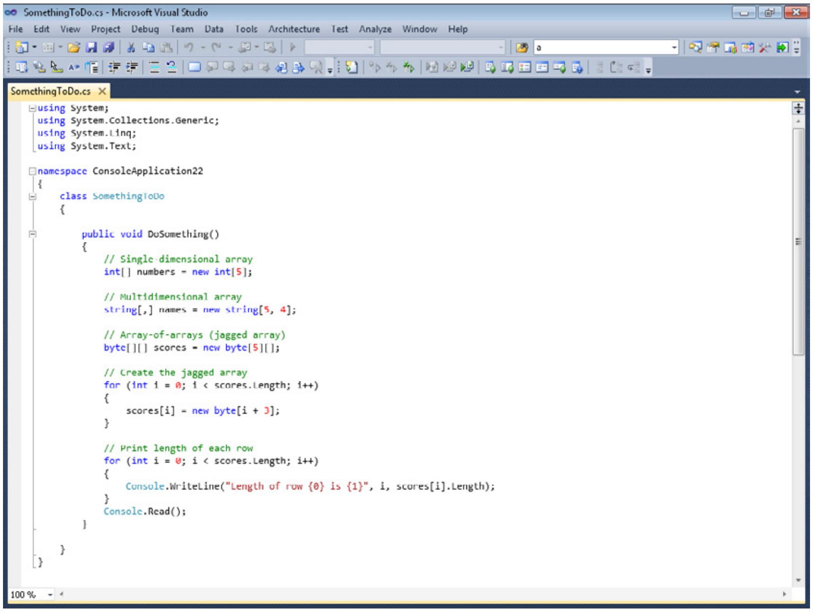
I talked about the different window layouts in vstipEnv0051, "Window Layouts: The Four Modes," on page 90, and vstipEnv0052, "Window Layouts: Design, Debug, and Full Screen," on page 91. Now I want to cover a lesser-known layout known as File View.

You get to File View by putting in the file name of any file associated with Visual Studio at the command prompt. In the following example, I've changed my directory to one of my solutions, and I'm putting in the name "SomethingToDo.cs":



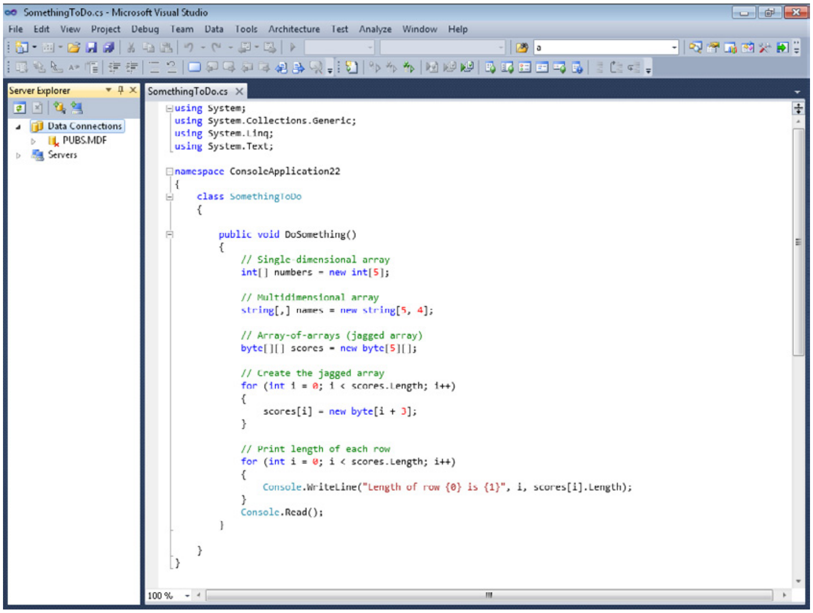
```
d:\Documents\Visual Studio 2010\Projects\ConsoleApplication22>SomethingToDo.cs
d:\Documents\Visual Studio 2010\Projects\ConsoleApplication22>
```

When I tried to do this without running the command prompt as Administrator, it wouldn't load the file. So you might have to run the command prompt as Administrator. If Visual Studio is already open, it shows the file and if not, it loads and shows the file:


SomethingToDo.cs - Microsoft Visual Studio
File Edit View Project Debug Team Data Tools Architecture Test Analyze Window Help
SomethingToDo.cs
100 %

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication22
{
    class SomethingToDo
    {

        public void DoSomething()
        {
            // Single-dimensional array
            int[] numbers = new int[5];

            // Multidimensional array
            string[,] names = new string[5, 4];

            // Array-of-arrays (jagged array)
            byte[][] scores = new byte[5][];

            // Create the jagged array
            for (int i = 0; i < scores.Length; i++)
            {
                scores[i] = new byte[i + 3];
            }

            // Print length of each row
            for (int i = 0; i < scores.Length; i++)
            {
                Console.WriteLine("Length of row {0} is {1}", i, scores[i].Length);
            }
            Console.Read();
        }

    }
}
```

Notice that it is a very sparse layout. The good news is that you can customize it any way you want, to have quick access to the tools you need most. I'll add Server Explorer for now:


SomethingToDo.cs - Microsoft Visual Studio
File Edit View Project Debug Team Data Tools Architecture Test Analyze Window Help
Server Explorer
Data Connections
PUBS.MDF
Servers
SomethingToDo.cs
100 %

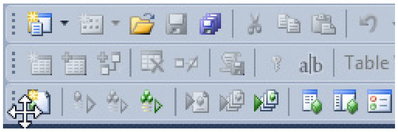
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication22
{
    class SomethingToDo
    {

        public void DoSomething()
        {
            // Single-dimensional array
            int[] numbers = new int[5];

            // Multidimensional array
            string[,] names = new string[5, 4];

            // Array-of-arrays (jagged array)
            byte[][] scores = new byte[5][];

            // Create the jagged array
            for (int i = 0; i < scores.Length; i++)
            {
                scores[i] = new byte[i + 3];
            }

            // Print length of each row
            for (int i = 0; i < scores.Length; i++)
            {
                Console.WriteLine("Length of row {0} is {1}", i, scores[i].Length);
            }
            Console.Read();
        }

    }
}
```

If you make changes, they are saved to your .vssettings file when you close Visual Studio. Remember that each mode is a distinct area, so the changes you make here are not seen in any of the other window modes.

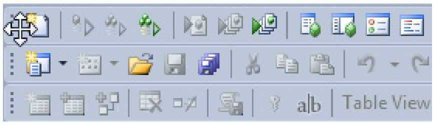
AX.36 Rearrange Your Toolbars

VERSIONS	2005, 2008, 2010
CODE	vstipEnv0027

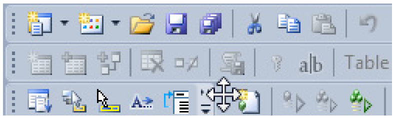
You can easily rearrange your toolbars by simply putting your mouse over the grip control (the four vertical dots to the left of every toolbar), as shown in the following illustration:



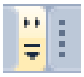
When you get the four-directional mouse pointer, just click and drag the toolbar to the new position:



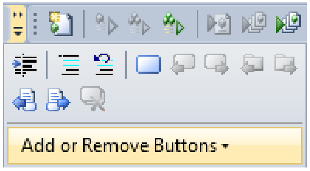
Be careful; it's real easy to get a shortened toolbar on your hands when moving toolbars beside other toolbars, as shown in the following illustration:



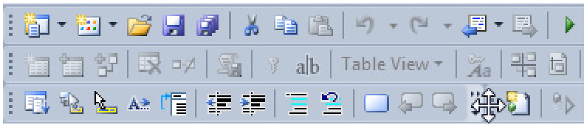
Note You know that you aren't seeing all the buttons on a toolbar if there are two arrows side-by-side on the far right of that toolbar, as shown in the following illustration:



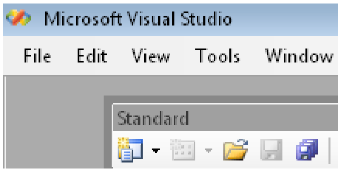
Of course, this shortened toolbar might be exactly what you want because the rest of the buttons are accessed by clicking on the drop-down arrow:



If this is not what you want, just click and drag the grip control of the toolbar to the right of the shortened toolbar and uncover as many buttons as you like:



In Visual Studio 2008 and 2005, you can also make the toolbar a floating one by dragging it off the toolbar area, as shown in the following illustration:

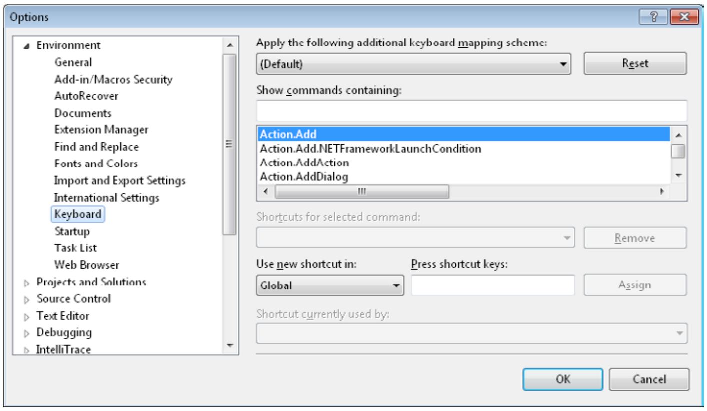


To put it back, just double-click anywhere on the title bar of the toolbar.

AX.37 Create a Shortcut Key for a Macro

WINDOWS	Alt, T, O
MENU	Tools | Options | Environment | Keyboard
COMMAND	Tools.CustomizeKeyboard
VERSIONS	2005, 2008, 2010
CODE	vstipTool0066

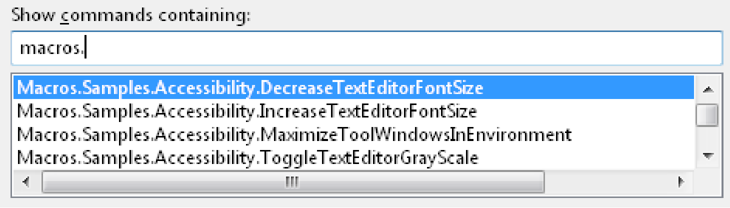
We covered creating shortcuts in vstipTool0063 (“Keyboard Shortcuts: Creating New Shortcuts,” page 127). Assuming you have a macro you would like to attach a shortcut key to, you can easily make your macros accessible. Let’s look at an example to show you how. First, go to Tools | Options | Environment | Keyboard:



Apply the following additional keyboard mapping scheme: (Default)

Use new shortcut in: Global

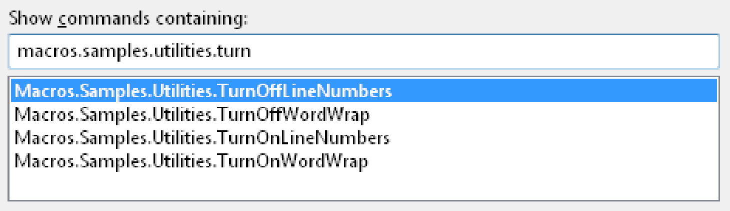
To access your macros, type **macros**. in the Show Commands Containing area:


Show commands containing: macros.

Macros.Samples.Accessibility.DecreaseTextEditorFontSize
Macros.Samples.Accessibility.IncreaseTextEditorFontSize
Macros.Samples.Accessibility.MaximizeToolWindowsInEnvironment
Macros.Samples.Accessibility.ToggleTextEditorGrayScale

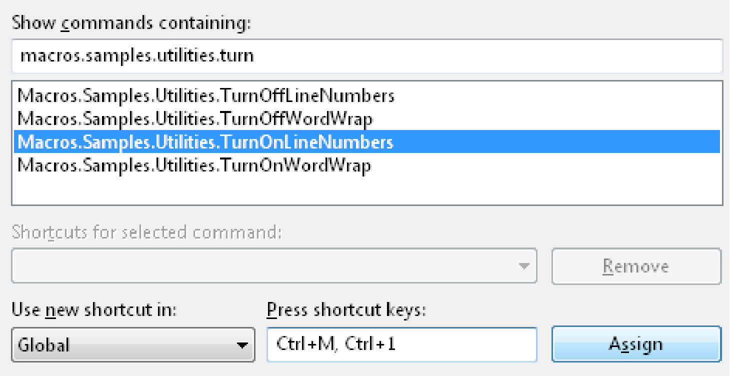
In this case, let's bind the Macros.Samples.Utilities.TurnOffLineNumbers and Macros.Samples.Utilities.TurnOnLineNumbers macros.

Warning Make sure the keyboard shortcuts you use here aren't already assigned to something else. They most likely aren't, but you should double-check by reviewing vstipTool0063 , "Keyboard Shortcuts Creating New Shortcuts," page 127.


Show commands containing: macros.samples.utilities.turn

Macros.Samples.Utilities.TurnOffLineNumbers
Macros.Samples.Utilities.TurnOffWordWrap
Macros.Samples.Utilities.TurnOnLineNumbers
Macros.Samples.Utilities.TurnOnWordWrap

Let's bind Macros.Samples.Utilities.TurnOnLineNumbers to Ctrl+M, CTRL+1, as shown in the following illustration:


Show commands containing: macros.samples.utilities.turn

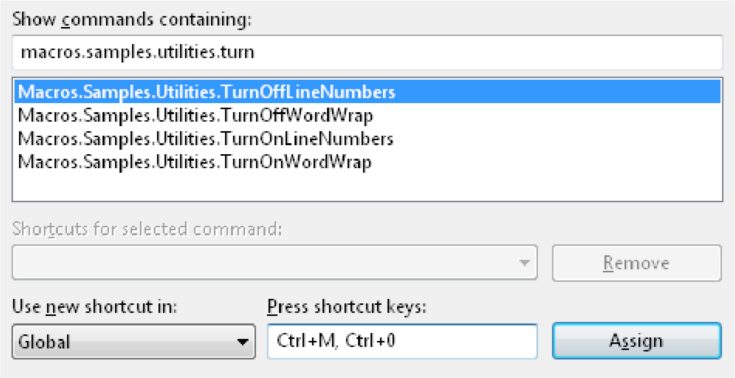
Macros.Samples.Utilities.TurnOffLineNumbers
Macros.Samples.Utilities.TurnOffWordWrap
Macros.Samples.Utilities.TurnOnLineNumbers
Macros.Samples.Utilities.TurnOnWordWrap

Shortcuts for selected command:

Use new shortcut in: Global

Press shortcut keys: Ctrl+M, Ctrl+1

Let's bind Macros.Samples.Utilities.TurnOffLineNumbers to Ctrl+M, Ctrl+0 (zero), as shown in the following illustration:



macros.samples.utilities.turn

Macros.Samples.Utilities.TurnOffLineNumbers
Macros.Samples.Utilities.TurnOffWordWrap
Macros.Samples.Utilities.TurnOnLineNumbers
Macros.Samples.Utilities.TurnOnWordWrap

Global

Ctrl+M, Ctrl+0

Click OK, and let’s test out our shortcuts. Go to any source code, and press Ctrl+M, Ctrl+0 to turn line numbers off and Ctrl+M, Ctrl+1 to turn them on.

AX.38 How to Run External Executables from the Command Window

COMMAND	Tools.Shell
VERSIONS	2005, 2008, 2010
CODE	vstipTool0089

You can run external programs from the Command Window (Ctrl+Alt+A) by using the **Tools. Shell** command. This can be useful if you run certain executables often (like Xcopy) and want to turn the action into a command alias. See vstipTool0068 (“Understanding Command Aliases,” page 113).

The general syntax for the command is as follows:

```
Tools.Shell [/command] [/output] [/dir:folder] path [args]
```

Arguments

Following are some key arguments you can use for this command:

- **/commandwindow**, **/command**, **/c**, or **/cmd**

  Optional. Specifies that the output for the executable is displayed in the Command Window.

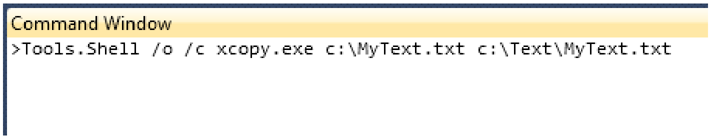
- **/dir:folder** or **/d: folder**

  Optional. Specifies the working directory to be set when the program is run.

- **/outputwindow**, **/output**, **/out**, or **/o**

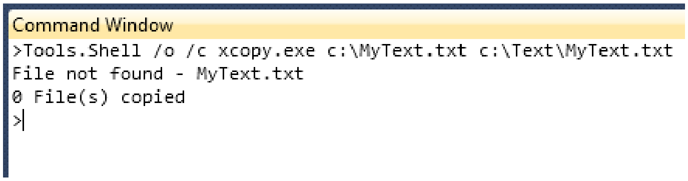
  Optional. Specifies that the output for the executable is displayed in the Output window.

For example, to run the Xcopy command, it would appear as shown in the following illustration:



```
>Tools.Shell /o /c xcopy.exe c:\MyText.txt c:\Text\MyText.txt
```

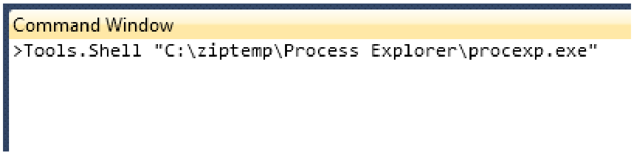
Just for fun, because I don't actually have the files in the sample command, the output shown in the following illustration is the result of running the Xcopy command:



```
>Tools.Shell /o /c xcopy.exe c:\MyText.txt c:\Text\MyText.txt
File not found - MyText.txt
0 File(s) copied
>
```

As you can see, the output from Xcopy was redirected to the Command Window and tells me that it can't find a file.

If the item you want to run isn't in the path environment variable, you must include the full path (surrounded by quotes if any spaces are in the path), as shown in the following illustration:



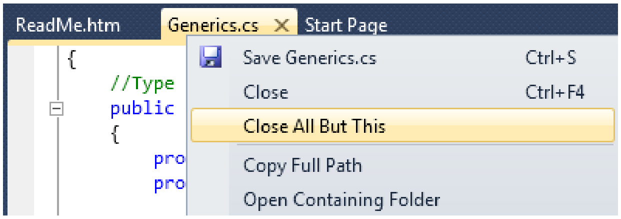
```
>Tools.Shell "C:\ziptemp\Process Explorer\procexp.exe"
```

Additional Tips from Chapter 4

AX.39 Close All But This on the File Tab Channel

DEFAULT	Alt+- (minus sign), A [VS2010 Only]
VISUAL BASIC 6	[no shortcut]
VISUAL C# 2005	Alt+- (minus sign), A [VS2010 Only]
VISUAL C++ 2	Alt+- (minus sign), A [VS2010 Only]
VISUAL C++ 6	Alt+- (minus sign), A [VS2010 Only]
VISUAL STUDIO 6	Alt+- (minus sign), A [VS2010 Only]
WINDOWS	[no shortcut]
COMMAND	File.CloseAllButThis; Window.ShowDockMenu
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0011

If you ever get the urge to close every file except the one you are currently working on, you can just right-click the current file tab and choose Close All But This:

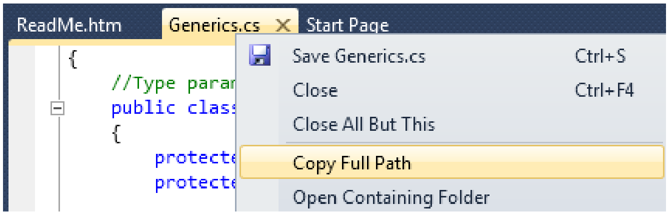


```
{
    //Type
    public
    {
        pro
        pro
```

AX.40 Copy a File's Full Path from the File Tab

DEFAULT	Alt+- (minus sign), F [VS2010 Only]
VISUAL BASIC 6	[no shortcut]
VISUAL C# 2005	Alt+- (minus sign), F [VS2010 Only]
VISUAL C++ 2	Alt+- (minus sign), F [VS2010 Only]
VISUAL C++ 6	Alt+- (minus sign), F [VS2010 Only]
VISUAL STUDIO 6	Alt+- (minus sign), F [VS2010 Only]
WINDOWS	[no shortcut]
COMMAND	File.CopyFullPath; Window.ShowDockMenu
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0013

You can quickly copy the full path of any file. Just right-click the tab for the file, and choose Copy Full Path, as shown in the following illustration:



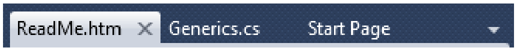
You now have the full path in your clipboard so that you can paste it (Ctrl+V) anywhere you need it.

AX.41 Understanding the File Tab Channel Drop-Down Button

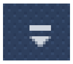
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0012

Here is a convenient way to know whether you are seeing all your files on the File Tab Channel.

Let's say you have a couple of files open: Notice that the button to the far right looks like an arrow pointing down:



But if you open up a few more files so that they all can't be seen on the file channel, the button changes to let you know that all the files are not visible on the File Tab Channel (indicated by the line above the arrow in the following illustration):

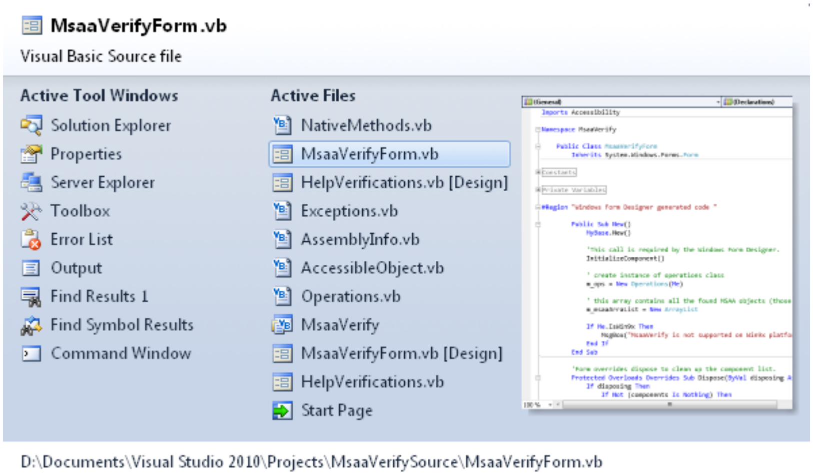


If you click the button, it gives you a list of all the files so that you can pick the one you want from the list.

AX.42 How to Disable the IDE Navigator

DEFAULT	Ctrl+F6 (next); Ctrl+Shift+F6 (previous)
VISUAL BASIC 6	Ctrl+F6 (next); Ctrl+Shift+F6 (previous)
VISUAL C# 2005	Ctrl+F6 (next); Ctrl+Shift+F6 (previous)
VISUAL C++ 2	Ctrl+F6; Ctrl+Tab (next); Ctrl+Shift+F6; Ctrl+Shift+Tab (previous)
VISUAL C++ 6	Ctrl+F6 (next); Ctrl+Shift+F6 (previous)
VISUAL STUDIO 6	Ctrl+F6; Ctrl+Tab (next); Ctrl+Shift+F6; Ctrl+Shift+Tab (previous)
WINDOWS	[no shortcut]
COMMAND	Window.NextDocumentWindow; Window.PreviousDocumentWindow
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0039

By default, if you press Ctrl+Tab, you get the IDE Navigator, as shown in the following illustration:



MsaaVerifyForm.vb
Visual Basic Source file

Active Tool Windows
- Solution Explorer
- Properties
- Server Explorer
- Toolbox
- Error List
- Output
- Find Results 1
- Find Symbol Results
- Command Window

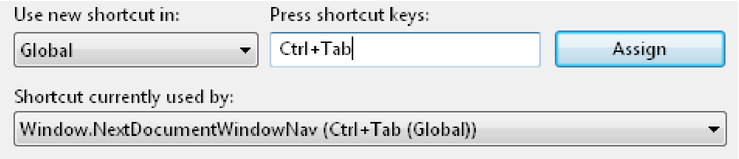
Active Files
- NativeMethods.vb
- MsaaVerifyForm.vb
- HelpVerifications.vb [Design]
- Exceptions.vb
- AssemblyInfo.vb
- AccessibleObject.vb
- Operations.vb
- MsaaVerify
- MsaaVerifyForm.vb [Design]
- HelpVerifications.vb
- Start Page

D:\Documents\Visual Studio 2010\Projects\MsaaVerifySource\MsaaVerifyForm.vb

Some people don't like this feature and instead would like to just iterate through open document tabs.

This behavior is bound to Ctrl+ F6 and to Ctrl+Shift+F6 in the General settings, but some people don't like this key combination.

You can easily rebind the commands to Ctrl+Tab and Ctrl+Shift+Tab. If you go to Tools | Options | Environment | Keyboard, you can see that Ctrl+Tab is bound to Window.NextDocumentWindowNav:

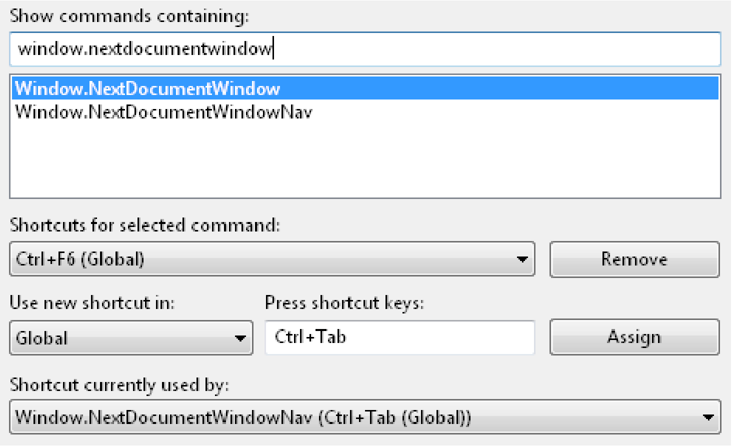


Use new shortcut in: Global

Press shortcut keys: Ctrl+Tab

Shortcut currently used by: Window.NextDocumentWindowNav (Ctrl+Tab (Global))

If you assign Ctrl+Tab to Window.NextDocumentWindow instead, you can see the result in the following illustration:



Show commands containing: window.nextdocumentwindow

Window.NextDocumentWindow
Window.NextDocumentWindowNav

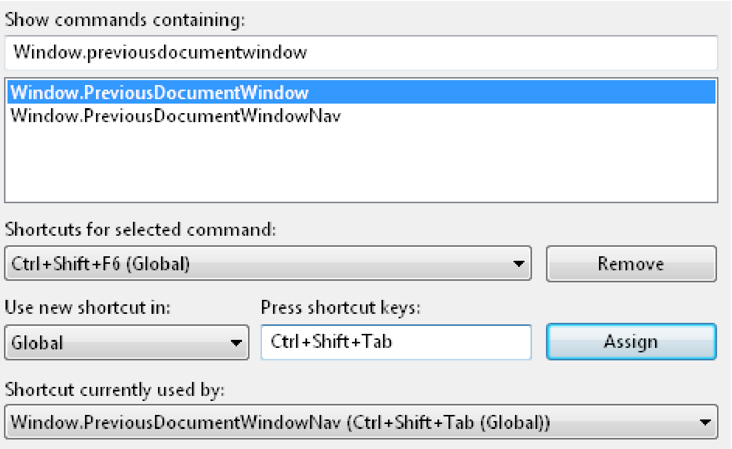
Shortcuts for selected command: Ctrl+F6 (Global)

Use new shortcut in: Global

Press shortcut keys: Ctrl+Tab

Shortcut currently used by: Window.NextDocumentWindowNav (Ctrl+Tab (Global))

The IDE Navigator does not show up anymore, and instead Ctrl+Tab iterates through the open document tabs. If you like this, you might want to bind Ctrl+Shift+Tab to Window.PreviousDocumentWindow as well:



Show commands containing: Window.previousdocumentwindow

Window.PreviousDocumentWindow
Window.PreviousDocumentWindowNav

Shortcuts for selected command: Ctrl+Shift+F6 (Global)

Use new shortcut in: Global

Press shortcut keys: Ctrl+Shift+Tab

Shortcut currently used by: Window.PreviousDocumentWindowNav (Ctrl+Shift+Tab (Global))

If you don't like the new setup, you can always reverse the process and rebind Ctrl+Tab and Ctrl+Shift+Tab to Window.NextDocumentWindowNav and Window.PreviousDocumentWindowNav, respectively.

For more information about binding keyboard shortcuts, refer to vstipTool0063 ("Keyboard Shortcuts Creating New Shortcuts," page 127).

AX.43 Thumbnail Previews in the IDE Navigator

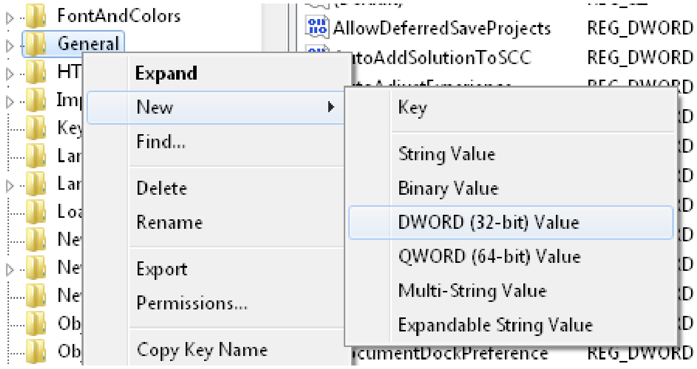
VERSIONS	2010
CODE	vstipTool0113

In vstipTool0023 ("Using the IDE Navigator," page 160), I show you how to use the IDE Navigator. This next tip comes from Paul Harrington on the Visual Studio Team. In Visual Studio 2008, when you used the IDE Navigator, you could also see thumbnail previews of the documents in the list. However, this feature was removed from Visual Studio 2010. The good news is that the feature is still there.

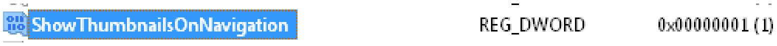
Warning This tip requires you to add a value to the registry, so you do this at your own risk. And this solution has been known to not always work 100 percent of the time.

Open up the registry (regedit.exe), and go to HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\10.0\General.

Now right-click the General key, and add a new DWORD value:

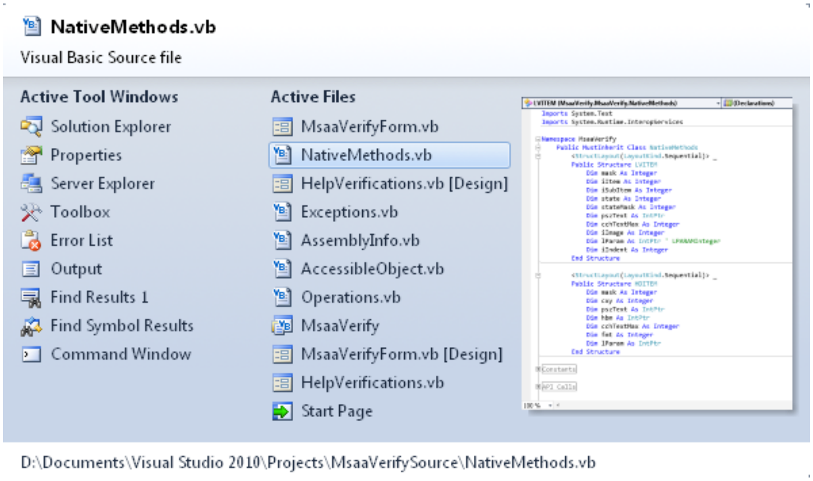


Call it ShowThumbnailsOnNavigation, and set the value to 1:



ShowThumbnailsOnNavigation REG_DWORD 0x00000001 (1)

Now you should have the thumbnail preview available:



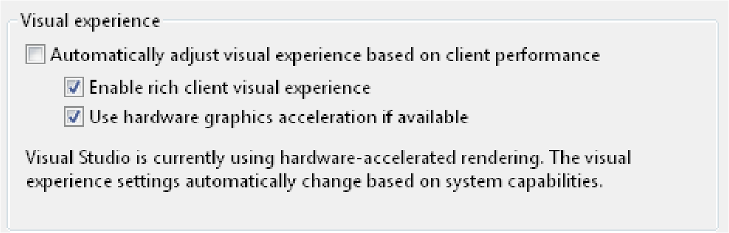
NativeMethods.vb
Visual Basic Source file

Active Tool Windows	Active Files
Solution Explorer	MsaaVerifyForm.vb
Properties	NativeMethods.vb
Server Explorer	HelpVerifications.vb [Design]
Toolbox	Exceptions.vb
Error List	AssemblyInfo.vb
Output	AccessibleObject.vb
Find Results 1	Operations.vb
Find Symbol Results	MsaaVerify
Command Window	MsaaVerifyForm.vb [Design]
	HelpVerifications.vb
	Start Page

D:\Documents\Visual Studio 2010\Projects\MsaaVerifySource\NativeMethods.vb

Troubleshooting

This is a little graphics intensive, so if you don't see the thumbnail, it could be because you haven't enabled the rich client visual experience under Tools | Options | Environment | General:



Visual experience

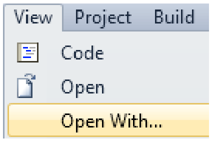
☐ Automatically adjust visual experience based on client performance
  ☑ Enable rich client visual experience
  ☑ Use hardware graphics acceleration if available

Visual Studio is currently using hardware-accelerated rendering. The visual experience settings automatically change based on system capabilities.

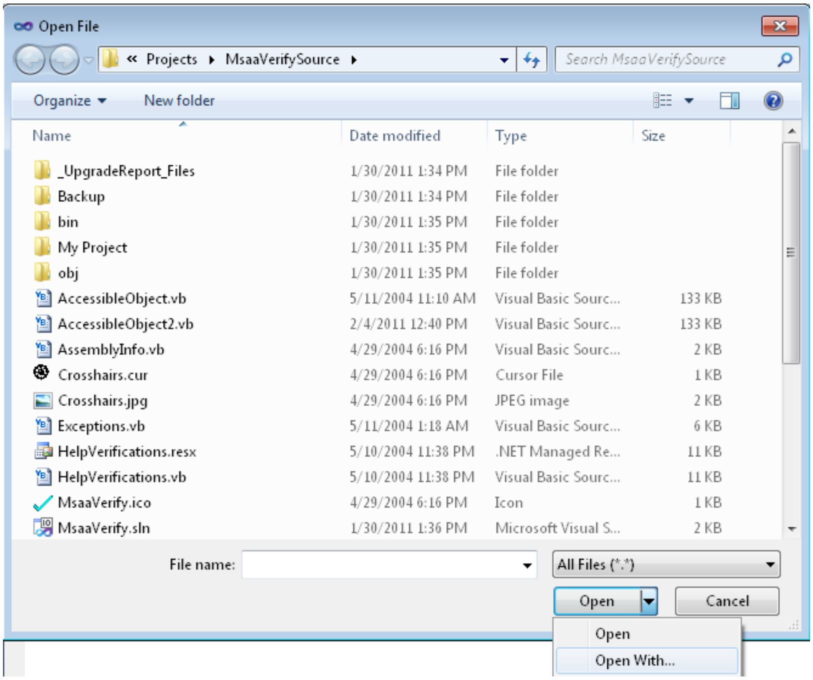
AX.44 Changing Editors Using Open With

WINDOWS	Alt,V, N
MENU	View | Open With
COMMAND	View.OpenWith
VERSIONS	2008, 2010
CODE	vstipEnv0037

You can use Open With to change the editor used to view a file. You can use this feature in several ways. For example, if you have an existing file open, you can go to View | Open With:

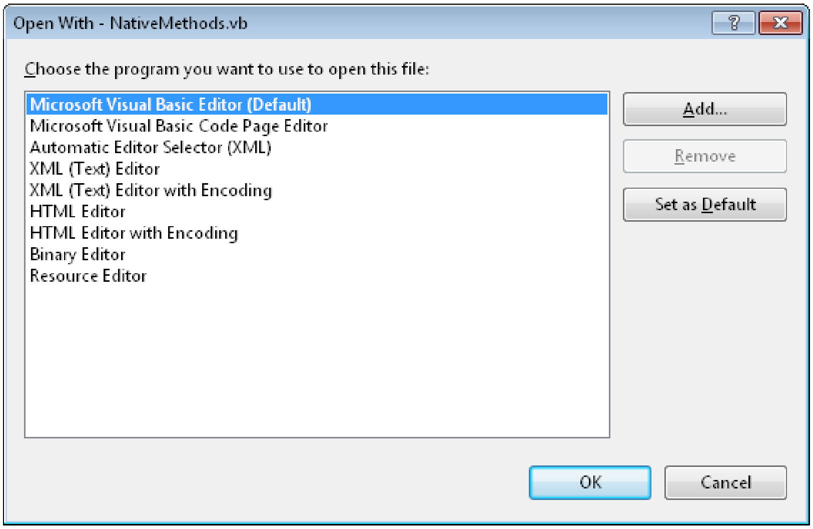


For files not opened yet, you can use this option from inside the Open File dialog box:



Name	Date modified	Type	Size
_UpgradeReport_Files	1/30/2011 1:34 PM	File folder	
Backup	1/30/2011 1:34 PM	File folder	
bin	1/30/2011 1:35 PM	File folder	
My Project	1/30/2011 1:35 PM	File folder	
obj	1/30/2011 1:35 PM	File folder	
AccessibleObject.vb	5/11/2004 11:10 AM	Visual Basic Sourc...	133 KB
AccessibleObject2.vb	2/4/2011 12:40 PM	Visual Basic Sourc...	133 KB
AssemblyInfo.vb	4/29/2004 6:16 PM	Visual Basic Sourc...	2 KB
Crosshairs.cur	4/29/2004 6:16 PM	Cursor File	1 KB
Crosshairs.jpg	4/29/2004 6:16 PM	JPEG image	2 KB
Exceptions.vb	5/11/2004 1:18 AM	Visual Basic Sourc...	6 KB
HelpVerifications.resx	5/10/2004 11:38 PM	.NET Managed Re...	11 KB
HelpVerifications.vb	5/10/2004 11:38 PM	Visual Basic Sourc...	11 KB
MsaaVerify.ico	4/29/2004 6:16 PM	Icon	1 KB
MsaaVerify.sln	1/30/2011 1:36 PM	Microsoft Visual S...	2 KB

Regardless of the method used, you get the Open With dialog box:

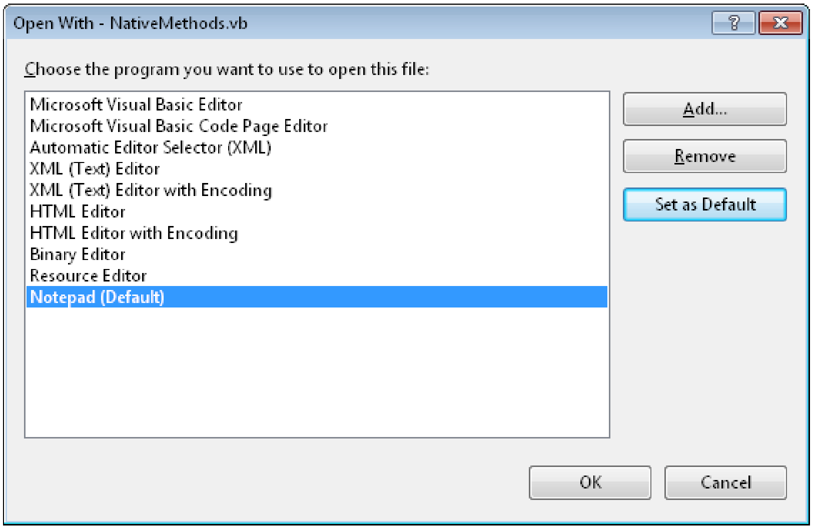


Open With - NativeMethods.vb

Choose the program you want to use to open this file:

- Microsoft Visual Basic Editor (Default)
- Microsoft Visual Basic Code Page Editor
- Automatic Editor Selector (XML)
- XML (Text) Editor
- XML (Text) Editor with Encoding
- HTML Editor
- HTML Editor with Encoding
- Binary Editor
- Resource Editor

Notice, in this example, that one editor is the default editor for the file type you want to open. This can easily be changed by selecting a new editor and clicking Set As Default:



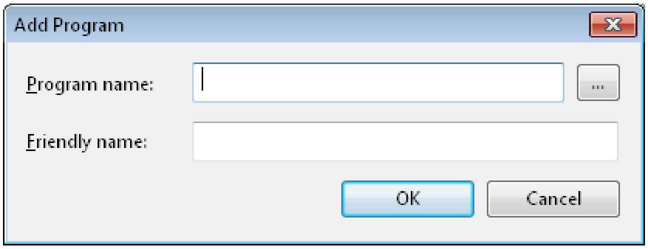
Open With - NativeMethods.vb

Choose the program you want to use to open this file:

- Microsoft Visual Basic Editor
- Microsoft Visual Basic Code Page Editor
- Automatic Editor Selector (XML)
- XML (Text) Editor
- XML (Text) Editor with Encoding
- HTML Editor
- HTML Editor with Encoding
- Binary Editor
- Resource Editor
- Notepad (Default)

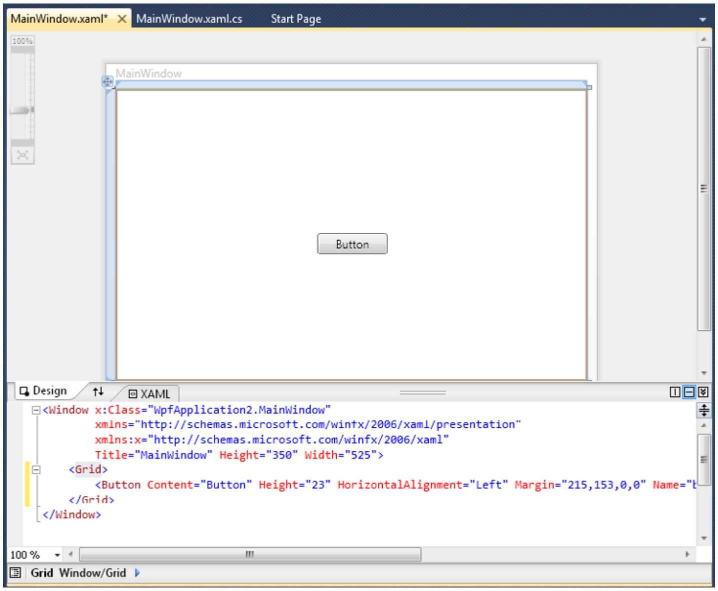
Warning Use this feature at your own risk, because it's important to be sure of the editor you are using before you make it the default.

Notice the Add button, shown in the following illustration, which enables you to indicate new programs that you want to use for opening files:



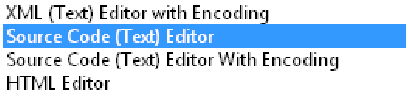
Example

One common use of Open With is to enable source code editing for WPF files. Normally, you get a designer/code view:



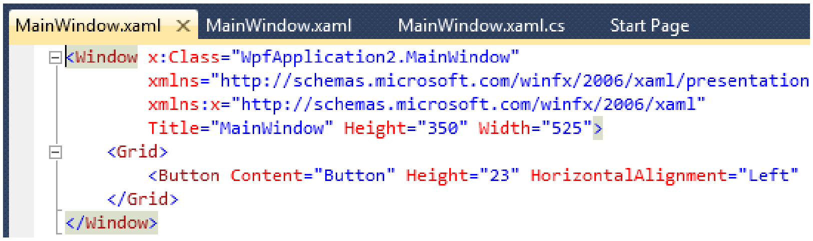
```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Button" Height="23" HorizontalAlignment="Left" Margin="215,153,0,0" Name="
    </Grid>
</Window>
```

To enable source code editing more easily, first use Open With and select Source Code (Text) Editor:



XML (Text) Editor with Encoding
Source Code (Text) Editor
Source Code (Text) Editor With Encoding
HTML Editor

Now you should see a “code only” view of the XAML:



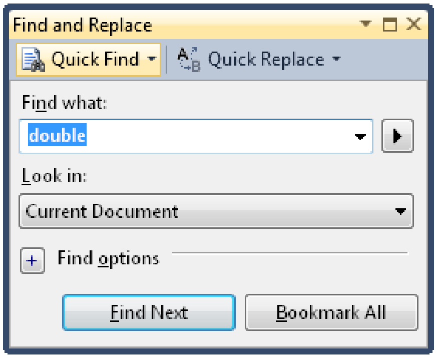
```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Button" Height="23" HorizontalAlignment="Left"
    </Grid>
</Window>
```

Additional Tips from Chapter 5

AX.45 Using a Simple Quick Find

DEFAULT	Ctrl+F
VISUAL BASIC 6	Ctrl+F
VISUAL C# 2005	Ctrl+F
VISUAL C++ 2	Alt+F3
VISUAL C++ 6	Ctrl+F
VISUAL STUDIO 6	Ctrl+F
WINDOWS	Alt,E, F, F
MENU	Edit | Find and Replace | Quick Find
COMMAND	Edit.Find
VERSIONS	2005, 2008, 2010
CODE	vstipFind0006

You can do a simple find anytime you want by pressing Ctrl+F to bring up the Quick Find window:



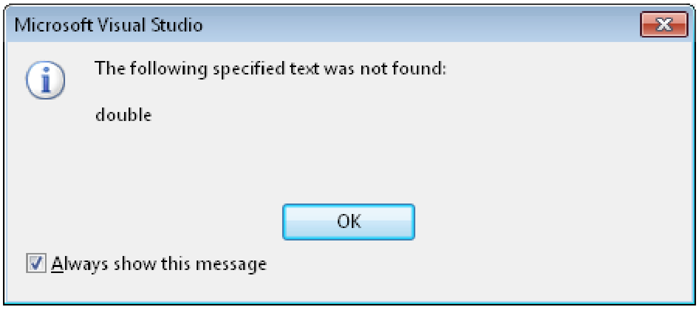
Find what: double

Look in: Current Document

The Find What area is automatically prepopulated with the word the cursor was currently on, or you can type in a new one.

To start, just press Enter or click Find Next, and the find operation finds the next instance of the search term you are looking for. By default, it looks from the current cursor location downward.

The search continues until you reach the end of the document, and then it returns to the beginning and shows the following dialog box:



The following specified text was not found:

double

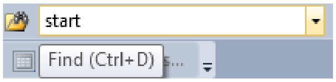
☑ Always show this message

As you can see, you can turn this off by clearing the Always Show This Message check box so that it doesn't annoy you.

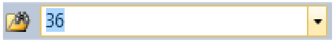
AX.46 Using the Find Combo Box

DEFAULT	Ctrl+D
VISUAL BASIC 6	[no shortcut]
VISUAL C# 2005	Ctrl+/
VISUAL C++ 2	Ctrl+D; Ctrl+F; Ctrl+A
VISUAL C++ 6	Ctrl+D
VISUAL STUDIO 6	Ctrl+Shift+F
WINDOWS	Ctrl+Shift+F
COMMAND	Edit.GoToFindCombo
VERSIONS	2005, 2008, 2010
CODE	vstipFind0009

While it tends to get ignored sometimes, the Find Combo box is actually quite useful. In the following examples, I'll show how you can quickly use this area while you are writing your code:



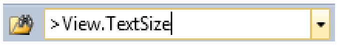
It can be used as a quick way to go to a line without any dialog boxes popping up. Just press Ctrl+D to get to the combo box, and then type in a line number, and finally, press Ctrl+G to go to the line number you typed:



36

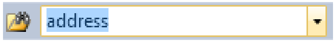
Also, it can be used to execute commands by the pressing Ctrl+Forward slash (/) followed by your command:

Note Ctrl+/ is bound to the **Tools.GoToCommandLine** command for all languages except C#.



>View.TextSize

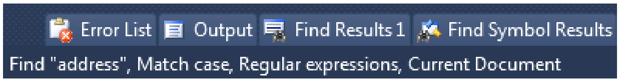
Its primary use, however, is to perform a simple find operation. Press Ctrl+D, type in whatever you are looking for, and press Enter:



address

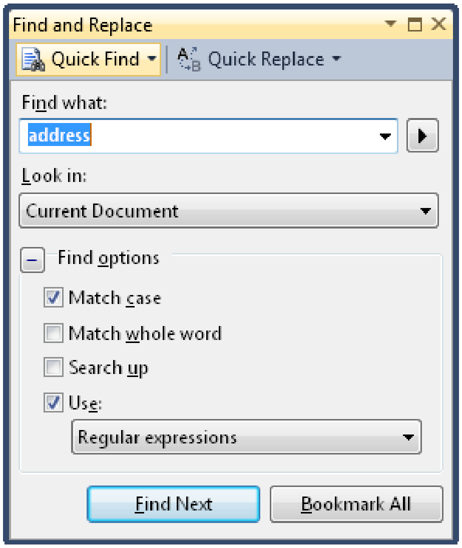
It finds the next instance of the search term. If you have a lot of text to go through, you can hold down the Enter key to quickly go through the document. One other thing to note is that, unlike a Quick Find, this find does *not* give you a dialog box telling you when you have looped around to where you started.

When you use this feature, *pay attention to the status bar* at the bottom. It shows you all the options that have been set for the current search:



Find "address", Match case, Regular expressions, Current Document

The options used here are set in the Quick Find tool window (Ctrl+F):



Find what: address

Look in: Current Document

Find options

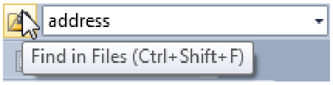
☑ Match case

☐ Match whole word

☐ Search up

☑ Use: Regular expressions

Last, but certainly not least, is the Find In Files button, which brings up the Find In Files tool window (see vstipFind0013, "Find in Files: Find Options," page 186):

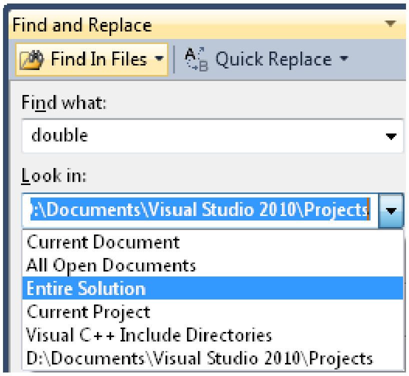


address

AX.47 Customize the Files to Search with Find In Files

DEFAULT	Ctrl+Shift+F
VISUAL BASIC 6	Ctrl+Shift+F
VISUAL C# 2005	Ctrl+Shift+F
VISUAL C++ 2	Ctrl+Shift+F
VISUAL C++ 6	Ctrl+Shift+F
VISUAL STUDIO 6	[no shortcut]
WINDOWS	Alt, E, F, I
MENU	Edit | Find and Replace | Find In Files
COMMAND	Edit.FindinFiles
VERSIONS	2005, 2008, 2010
CODE	vstipFind0005

In the Find In Files dialog box (Ctrl+Shift+F), choose any option *except* Current Document and All Open Documents:

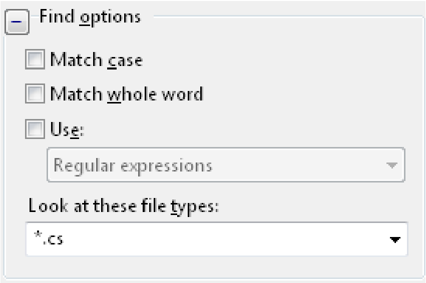


Find what: double

Look in: :\Documents\Visual Studio 2010\Projects

- Current Document
- All Open Documents
- Entire Solution
- Current Project
- Visual C++ Include Directories
- D:\Documents\Visual Studio 2010\Projects

The Look At These File Types combo box is enabled under Find Options:



☐ Match case

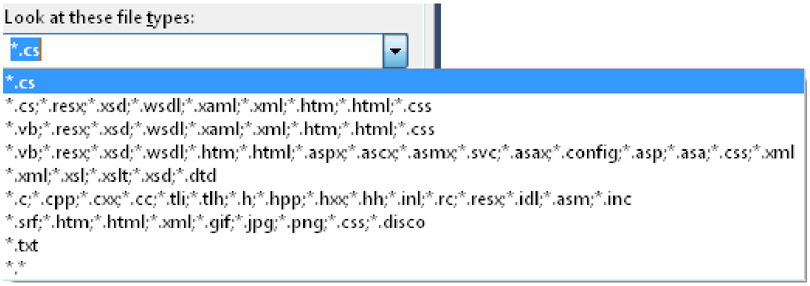
☐ Match whole word

☐ Use:

Regular expressions

Look at these file types: *.cs

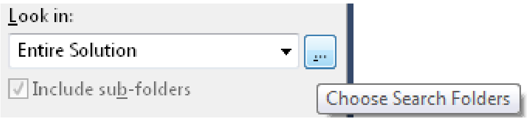
Now you can choose from the predetermined list of file types, or you can include your own (semicolon delimited) list:



Look at these file types: *.cs

```
*.cs
*.cs;*.resx;*.xsd;*.wsdl;*.xaml;*.xml;*.htm;*.html;*.css
*.vb;*.resx;*.xsd;*.wsdl;*.xaml;*.xml;*.htm;*.html;*.css
*.vb;*.resx;*.xsd;*.wsdl;*.htm;*.html;*.aspx;*.ascx;*.asmx;*.svc;*.asax;*.config;*.asp;*.asa;*.css;*.xml
*.xml;*.xsl;*.xslt;*.xsd;*.dtd
*.c;*.cpp;*.cxx;*.cc;*.tli;*.tlh;*.h;*.hpp;*.hxx;*.hh;*.inl;*.rc;*.resx;*.idl;*.asm;*.inc
*.srf;*.htm;*.html;*.xml;*.gif;*.jpg;*.png;*.css;*.disco
*.txt
*.*
```

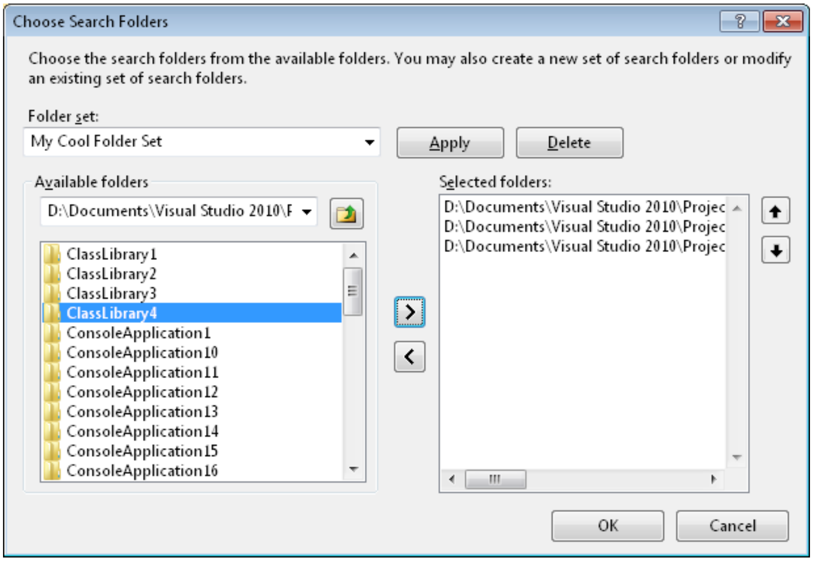
You can also create a customized set of folders to search in by going back to the Look In area and clicking the ellipsis:



Look in: Entire Solution

☑ Include sub-folders

Now you have the Choose Search Folders dialog box, where you can completely customize the folders to look in, and you can even create a custom name for the folder set by typing a new name in the Folder Set area:



Choose Search Folders

Choose the search folders from the available folders. You may also create a new set of search folders or modify an existing set of search folders.

Folder set: My Cool Folder Set

Available folders: D:\Documents\Visual Studio 2010\F

ClassLibrary1
ClassLibrary2
ClassLibrary3
ClassLibrary4
ConsoleApplication1
ConsoleApplication10
ConsoleApplication11
ConsoleApplication12
ConsoleApplication13
ConsoleApplication14
ConsoleApplication15
ConsoleApplication16

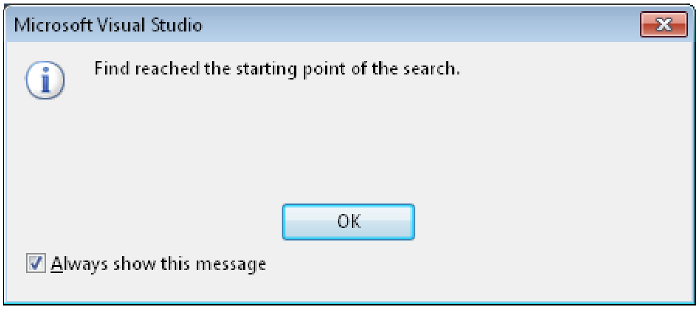
Selected folders:

D:\Documents\Visual Studio 2010\Projec
D:\Documents\Visual Studio 2010\Projec
D:\Documents\Visual Studio 2010\Projec

AX.48 How to Show and Hide Find Messages

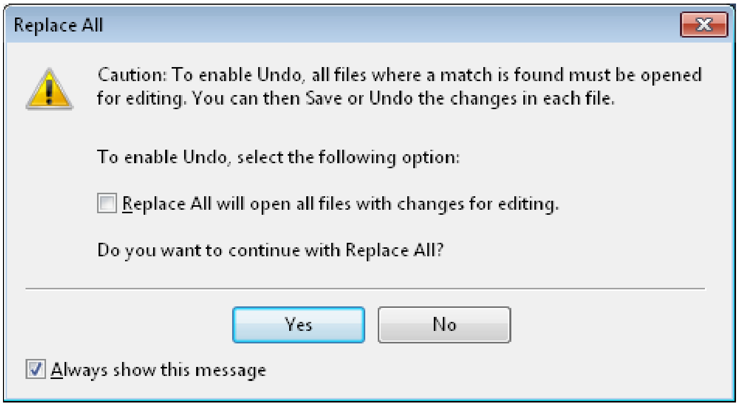
WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | Find and Replace
VERSIONS	2005, 2008, 2010
CODE	vstipFind0017

When working with the various find-tool windows, you sometimes get messages based on what you are doing. These range from informational messages to warning messages, as shown in the following illustrations:



Find reached the starting point of the search.

☑ Always show this message



Caution: To enable Undo, all files where a match is found must be opened for editing. You can then Save or Undo the changes in each file.

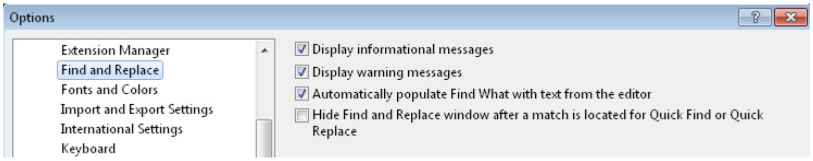
To enable Undo, select the following option:

☐ Replace All will open all files with changes for editing.

Do you want to continue with Replace All?

☑ Always show this message

If you clear the Always Show This Message check box, the messages go away. But what if you want the messages back or don't want to wait for a message to pop up until you can turn the messages off? No problem. Just go to Tools | Options | Environment | Find And Replace:



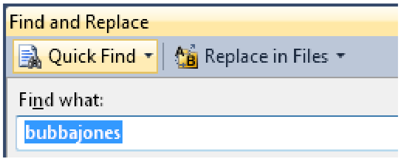
☑ Display informational messages
☑ Display warning messages
☑ Automatically populate Find What with text from the editor
☐ Hide Find and Replace window after a match is located for Quick Find or Quick Replace

The Display Informational Messages and Display Warning Messages check boxes toggle, showing these messages when you use a find operation. Informational messages really aren't very important, but you should consider carefully whether turning off warning messages is a good idea.

AX.49 How to Not Automatically Search for the Currently Selected Word

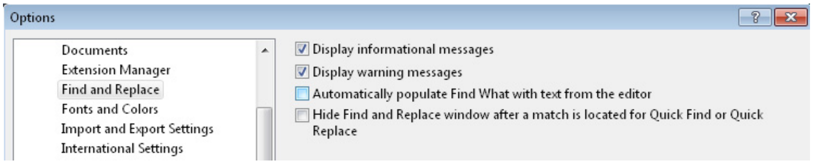
WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | Find and Replace
VERSIONS	2005, 2008, 2010
CODE	vstipFind0018

Does it annoy you when your find operations automatically populate with the word you happen to have the cursor in?



Find what:
bubbajones

You can simply go to Tools | Options | Environment | Find And Replace and clear the Automatically Populate Find What With Text From The Editor check box:



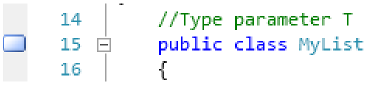
☑ Display informational messages
☑ Display warning messages
☐ Automatically populate Find What with text from the editor
☐ Hide Find and Replace window after a match is located for Quick Find or Quick Replace

AX.50 Setting Bookmarks

DEFAULT	Ctrl+K, Ctrl+K
VISUAL BASIC 6	Ctrl+K, Ctrl+K; Ctrl+K, T
VISUAL C# 2005	Ctrl+K, Ctrl+K; Ctrl+B, Ctrl+T; Ctrl+B, T
VISUAL C++ 2	Ctrl+F2
VISUAL C++ 6	Ctrl+K, Ctrl+K; Ctrl+F2
VISUAL STUDIO 6	Ctrl+K, Ctrl+K
WINDOWS	Alt,E, K, T
MENU	Edit | Bookmarks | Toggle Bookmark
COMMAND	Edit.ToggleBookmark
VERSIONS	2005, 2008, 2010
CODE	vstipTool0047

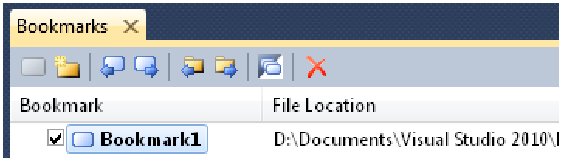
Bookmarks are a pretty cool feature that a lot of people don't seem to know about. Essentially, bookmarks provide a way to mark locations in your code. Unlike comment tokens ("TODOs"), bookmarks are not stored with the source code and thus are seen only by you.

You have numerous ways to set a Bookmark. The simplest way is to use Ctrl+K, Ctrl+K to create a single bookmark:



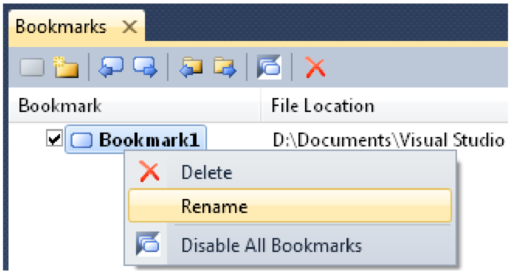
```
    //Type parameter T
    public class MyList
    {
```

When you set a bookmark, it creates a glyph in the margin (see image above) and creates an entry in the Bookmarks window:



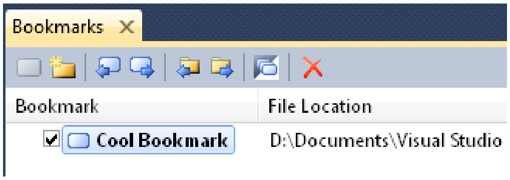
Bookmark	File Location
☑ Bookmark1	D:\Documents\Visual Studio 2010\

The good news is that you don't have to keep the default name that is given for the bookmark. Just right-click the entry in the window, and choose Rename:



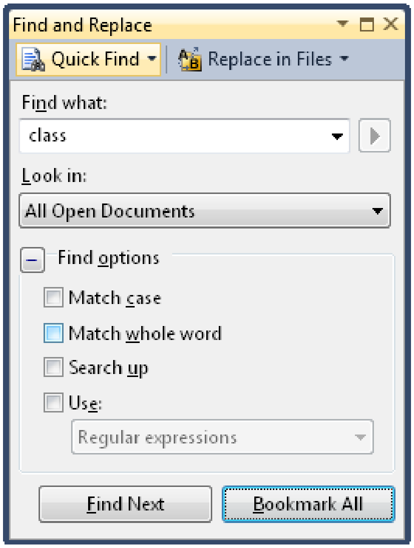
Bookmark	File Location
☑ Bookmark1	D:\Documents\Visual Studio

Then put in whatever you want for the name:



Bookmark	File Location
☑ Cool Bookmark	D:\Documents\Visual Studio

You can continue to use either the keyboard command or the menu option to create bookmarks. Another great way to create bookmarks is to use the Bookmark All (bottom right) button in the Quick Find dialog box (Ctrl+F):



Find what: class

Look in: All Open Documents

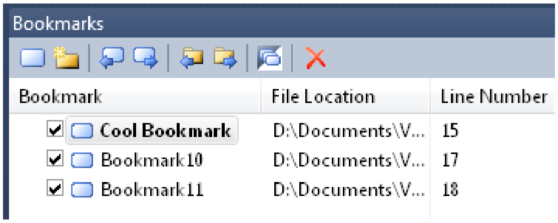
Find options
☐ Match case
☐ Match whole word
☐ Search up
☐ Use: Regular expressions

The Bookmark All button becomes available only if you choose Current Document or All Open Documents from the Look In drop-down box.

AX.51 Organizing Bookmarks

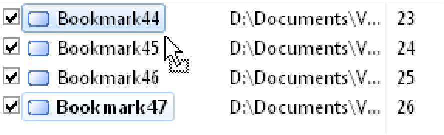
VERSIONS	2005, 2008, 2010
CODE	vstipTool0048

The Bookmarks window gives you some basic information about the bookmarks, and you have the ability to rename them as well:



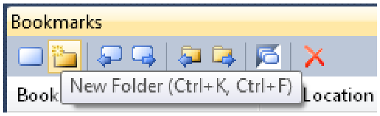
Bookmark	File Location	Line Number
☑ **Cool Bookmark**	D:\Documents\V...	15
☑ Bookmark10	D:\Documents\V...	17
☑ Bookmark11	D:\Documents\V...	18

However, when you have a lot of bookmarks, it is probably a good idea to organize your bookmarks. You can drag the bookmarks around to reorganize them in the list:

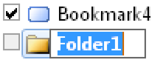


☑ Bookmark44	D:\Documents\V...	23
☑ Bookmark45	D:\Documents\V...	24
☑ Bookmark46	D:\Documents\V...	25
☑ **Bookmark47**	D:\Documents\V...	26

Another thing you can do is create folders in the Bookmarks window (Ctrl+K, Ctrl+F):

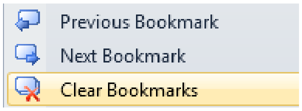


Just give the folder(s) a name, and you have a great place to organize bookmarks:



Now you can just drag your bookmarks into the folder(s) you have created:

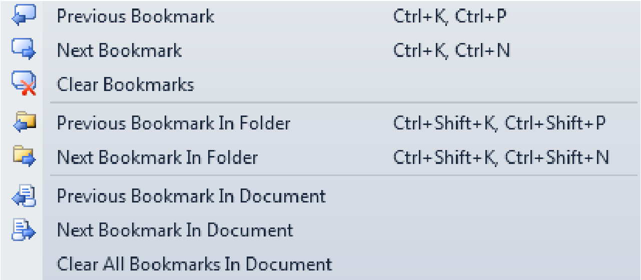
Naturally, if you have no need for a bookmark, you can simply press Del to delete the currently selected one, or you can go to Edit | Bookmarks | Clear Bookmarks to remove all your bookmarks:



AX.52 Navigating Bookmarks

DEFAULT	Ctrl+K, Ctrl+P (previous bookmark); Ctrl+K, Ctrl+N (next bookmark); Ctrl+Shift+K, Ctrl+Shift+P (previous bookmark in folder); Ctrl+Shift+K, Ctrl+Shift+N (next bookmark in folder)
VISUAL BASIC 6	Ctrl+K, Ctrl+P (previous bookmark); Ctrl+K, P (previous bookmark); Ctrl+K, Ctrl+N (next bookmark); Ctrl+K, N (next bookmark); Ctrl+Shift+K, Ctrl+Shift+P (previous bookmark in folder); Ctrl+Shift+K, Ctrl+Shift+N (next bookmark in folder)
VISUAL C# 2005	Ctrl+K, Ctrl+P (previous bookmark); Ctrl+B, Ctrl+P (previous bookmark); Ctrl+B, P (previous bookmark); Ctrl+K, Ctrl+N (next bookmark); Ctrl+B, Ctrl+N (next bookmark); Ctrl+B, N (next bookmark); [no shortcut] (previous bookmark in folder); [no shortcut] (next bookmark in folder)
VISUAL C++ 2	Shift+F2 (previous bookmark); F2 (next bookmark); Ctrl+Shift+K, Ctrl+Shift+P (previous bookmark in folder); Ctrl+Shift+K, Ctrl+Shift+N (next bookmark in folder)
VISUAL C++ 6	Ctrl+K, Ctrl+P (previous bookmark); Shift+F2 (previous bookmark); Ctrl+K, Ctrl+N (next bookmark); F2 (next bookmark); Ctrl+Shift+K, Ctrl+Shift+P (previous bookmark in folder); Ctrl+Shift+K, Ctrl+Shift+N (next bookmark in folder)
VISUAL STUDIO 6	Ctrl+K, Ctrl+P (previous bookmark); Ctrl+K, Ctrl+N (next bookmark); Ctrl+Shift+K, Ctrl+Shift+P (previous bookmark in folder); Ctrl+Shift+K, Ctrl+Shift+N (next bookmark in folder)
WINDOWS	Alt,E, K, [P (previous),B (next)]
MENU	Edit | Bookmarks | [Previous, Next] Bookmark [In Folder, Document]
COMMAND	Edit.[Previous, Next] Bookmark; Edit.[Previous, Next] Bookmark [In Folder, Document]
VERSIONS	2005, 2008, 2010
CODE	vstipTool0049

After you have created and organized your bookmarks, you want to navigate them. You can go to Edit | Bookmarks to see several available options:


Previous Bookmark Ctrl+K, Ctrl+P
Next Bookmark Ctrl+K, Ctrl+N
Clear Bookmarks
Previous Bookmark In Folder Ctrl+Shift+K, Ctrl+Shift+P
Next Bookmark In Folder Ctrl+Shift+K, Ctrl+Shift+N
Previous Bookmark In Document
Next Bookmark In Document
Clear All Bookmarks In Document

[Next,Previous] Bookmark In Document

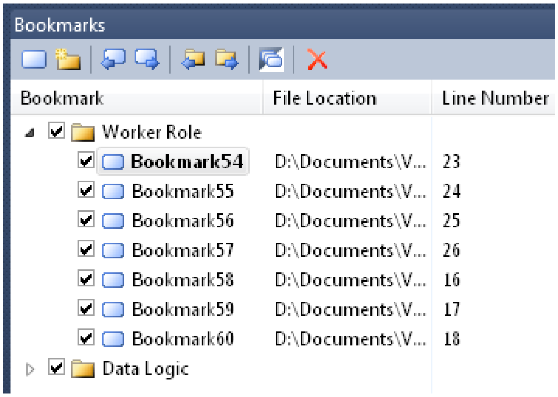
This option allows you to restrict browsing to only those bookmarks in the current open document, independently of how they are organized in the Bookmarks window.

So, even though you have lots of bookmarks in different folders within the Bookmarks window, this option moves only between the bookmarks in the current document, ignoring any organization.

[Next,Previous] Bookmark In Folder

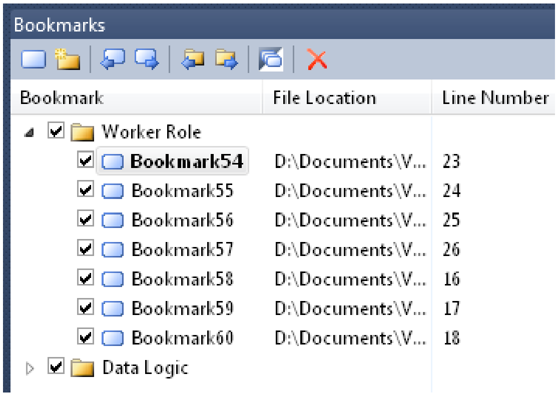
This option allows you to restrict browsing to only those bookmarks in the current folder in the Bookmarks window, independent of how they are arranged in the source code.

Pretty much the opposite of the previous feature, this feature ignores how the bookmarks are organized in the source code and moves only within the bookmarks in the current folder within the Bookmarks window:



Bookmark	File Location	Line Number
☑ Worker Role		
☑ Bookmark54	D:\Documents\V...	23
☑ Bookmark55	D:\Documents\V...	24
☑ Bookmark56	D:\Documents\V...	25
☑ Bookmark57	D:\Documents\V...	26
☑ Bookmark58	D:\Documents\V...	16
☑ Bookmark59	D:\Documents\V...	17
☑ Bookmark60	D:\Documents\V...	18
☑ Data Logic		

When you reach the last Bookmark in the folder, it loops back around to the first bookmark in the current folder:

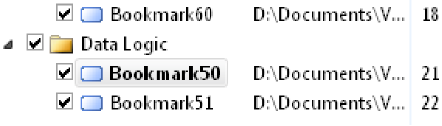


Bookmark	File Location	Line Number
☑ Worker Role		
☑ **Bookmark54**	D:\Documents\V...	23
☑ Bookmark55	D:\Documents\V...	24
☑ Bookmark56	D:\Documents\V...	25
☑ Bookmark57	D:\Documents\V...	26
☑ Bookmark58	D:\Documents\V...	16
☑ Bookmark59	D:\Documents\V...	17
☑ Bookmark60	D:\Documents\V...	18
☑ Data Logic		

[Next,Previous] Bookmark

This option allows you to navigate between bookmarks in the Bookmarks window. This feature is very similar to the [Previous, Next] Bookmark In Folder feature and moves sequentially through bookmarks.

The difference comes when you reach the last bookmark in a folder. Instead of looping back around to the first bookmark in the folder, this option continues to the next folder and moves sequentially through those bookmarks as well (and so on):



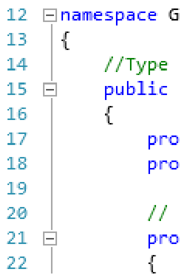
☑ Bookmark60	D:\Documents\V...	18
☑ Data Logic		
☑ **Bookmark50**	D:\Documents\V...	21
☑ Bookmark51	D:\Documents\V...	22

Additional Tips from Chapter 6

AX.53 Turn On Line Numbers

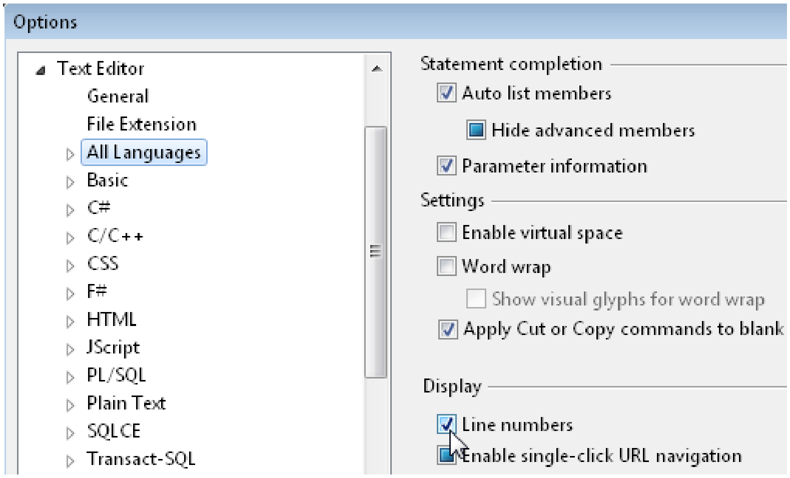
WINDOWS	Alt,T,O
MENU	Tools | Options | Text Editor | All Languages | General | Display
COMMAND	Macros.Samples.Utilities.TurnOnLineNumbers; Macros.Samples.Utilities.TurnOffLineNumbers
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0025

As you can see in the following illustration, it's great to have line numbers in your code:



```
12 namespace G
13 {
14     //Type
15     public
16     {
17         pro
18         pro
19
20         //
21         pro
22         {
```

Line numbers are not on by default. To turn on line numbers, just go to Tools | Options | Text Editor | General | Display and select the Line Numbers check box:



Statement completion

☑ Auto list members

■ Hide advanced members

☑ Parameter information

Settings

☐ Enable virtual space

☐ Word wrap

☐ Show visual glyphs for word wrap

☑ Apply Cut or Copy commands to blank

Display

☑ Line numbers

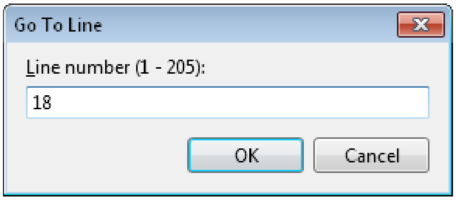
■ Enable single-click URL navigation

AX.54 Go to a Line Number

DEFAULT	Ctrl+G
VISUAL BASIC 6	[no shortcut]
VISUAL C# 2005	Ctrl+G
VISUAL C++ 2	Ctrl+G
VISUAL C++ 6	Ctrl+G
VISUAL STUDIO 6	Ctrl+G
WINDOWS	Alt,E,G
MENU	Edit | Go To
COMMAND	Edit.GoTo
VERSIONS	2005, 2008, 2010, 2010 SP1
CODE	vstipEdit0026

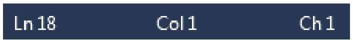
You have three main ways you can go to any line number in your code.

First, you can go to any line number by simply pressing Ctrl+G to see the following dialog box. Just type in your desired line number, and click OK. The cursor moves to the line number you typed:



Line number (1 - 205): 18

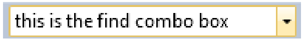
Second, you can double-click in the status bar area that shows your current location (lower-right part of your screen) to get the "Go To Line" dialog box:



Ln 18 Col 1 Ch 1

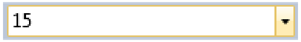
Third, you can use the Find Combo box to quickly go to a line number. This technique does not work with a default installation of Visual Studio 2010 but was fixed with Service Pack 1. It requires two steps:

1. Press Ctrl+D to put your cursor into the Find Combo box:



this is the find combo box

2. Type in any line number, and press Ctrl+G to go to that line number:



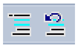
15

Note In Visual Studio 2010, you might have to press the escape key (Esc) to get out of the Find Combo box and back to your code.

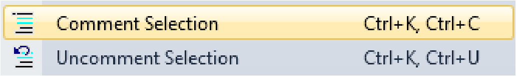
AX.55 Comment and Uncomment Code

DEFAULT	Ctrl+K, Ctrl+C (comment); Ctrl+K, Ctrl+U (uncomment)
VISUAL BASIC 6	Ctrl+K, Ctrl+C (comment); Ctrl+K, Ctrl+U (uncomment)
VISUAL C# 2005	Ctrl+K, Ctrl+C (comment); Ctrl+E, Ctrl+C (comment); Ctrl+E, C (comment); Ctrl+E, Ctrl+U (uncomment); Ctrl+E, U (uncomment); Ctrl+K, Ctrl+U (uncomment)
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+K, Ctrl+C (comment); Ctrl+K, Ctrl+U (uncomment)
VISUAL STUDIO 6	Ctrl+K, Ctrl+C (comment); Ctrl+K, Ctrl+U (uncomment)
WINDOWS	Alt,E, V, M; Alt,E, V, E;
MENU	Edit | Advanced | Comment Selection; Edit | Advanced | Uncomment Selection
COMMAND	Edit.CommentSelection; Edit.UncommentSelection
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0047

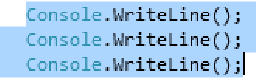
Sometimes it's the simple things we forget about. So I present to you the classic Comment and Uncomment selection. Naturally, you have the Comment and Uncomment buttons, shown in the following illustration:



And, of course, you have the menu items:

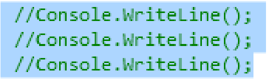

Comment Selection Ctrl+K, Ctrl+C
Uncomment Selection Ctrl+K, Ctrl+U

But it's the keyboard shortcuts that are really important. These, predictably, comment or uncomment lines of code for you. So let's say you have some code you want commented out. Just select it, as shown in the following illustration:



```
  Console.WriteLine();
  Console.WriteLine();
  Console.WriteLine();
```

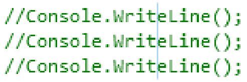
Then press Ctrl+K, Ctrl+C:



```
//Console.WriteLine();
//Console.WriteLine();
//Console.WriteLine();
```

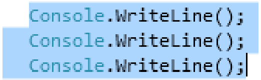
OK, great, but what if you don’t want to use the mouse? No problem. Just hold Alt+Shift+[Up or Down Arrow] to do a vertical selection. You don’t have to select the entire line to comment or uncomment it.

Note In Visual Studio 2005 and Visual Studio 2008, you have to go right or left one character before you can go up or down for vertical selection.



```
//Console.WriteLine();
//Console.WriteLine();
//Console.WriteLine();
```

Then press Ctrl+K, Ctrl+U (in this example):



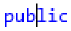
```
  Console.WriteLine();
  Console.WriteLine();
  Console.WriteLine();
```

And there you go. You now have Comment and Uncomment actions anytime you want them.

AX.56 Select the Current Word

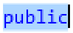
DEFAULT	Ctrl+W
VISUAL BASIC 6	Ctrl+Shift+W
VISUAL C# 2005	Ctrl+Shift+W
VISUAL C++ 2	Ctrl+W
VISUAL C++ 6	Ctrl+W
VISUAL STUDIO 6	Ctrl+W
WINDOWS	[no shortcut]
COMMAND	Edit.SelectCurrentWord
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0039

You can easily select the current word in Visual Studio by putting your cursor in the word:



```
public
```

And then just press Ctrl+W:

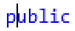


```
public
```

AX.57 Delete Through the Beginning or End of a Word

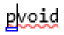
DEFAULT	Ctrl+Del (delete to end); Ctrl+Backspace (delete to start)
VISUAL BASIC 6	Ctrl+Del (delete to end); Ctrl+Backspace (delete to start
VISUAL C# 2005	Ctrl+Del (delete to end); Ctrl+Backspace (delete to start
VISUAL C++ 2	Ctrl+Del (delete to end); Ctrl+Backspace (delete to start
VISUAL C++ 6	Ctrl+Del (delete to end); Ctrl+Backspace (delete to start
VISUAL STUDIO 6	Ctrl+Del (delete to end); Ctrl+Backspace (delete to start
WINDOWS	[no shortcut]
COMMAND	Edit.WordDeleteToEnd; Edit.WordDeleteToStart
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0040

You can delete from the current cursor position through the beginning or end of a word. Let me Illustrate with a simple example. Let's say you want to change the word "public":



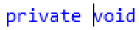
```
public
```

Given the current cursor position, you could delete to the end of the word by pressing Ctrl+Del:



```
pvoid
```

Then you could type the new word:



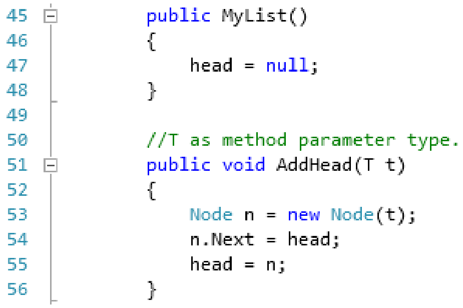
```
private void
```

This is a somewhat contrived example, but you get the idea. Additionally, Ctrl+Backspace deletes from the current cursor location through the beginning of a word.

AX.58 Click and Drag Text to a New Location

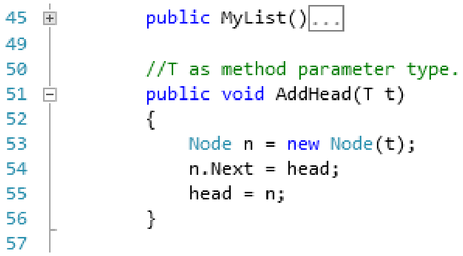
COMMAND	OtherContextMenus.DragandDrop.MoveHere
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0041

Often you will find yourself with the need to move text around in the Editor. Text can easily be moved around by simply selecting it and dragging it to a new location. Start with something you want to move, like a method:



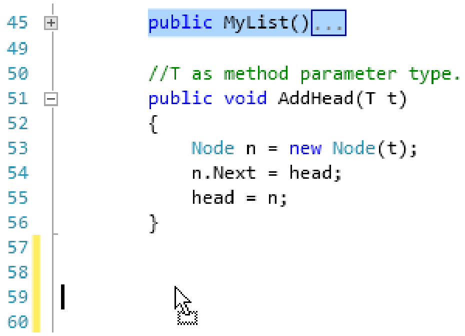
```
        public MyList()
        {
            head = null;
        }

        //T as method parameter type.
        public void AddHead(T t)
        {
            Node n = new Node(t);
            n.Next = head;
            head = n;
        }
```

In this case, you might want to collapse the code to make it easier to select and move, as shown in the following illustration:



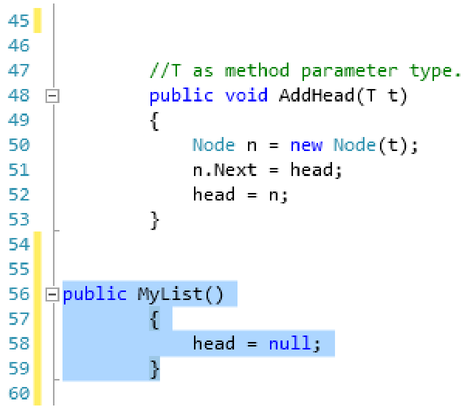
```
        public MyList()...

        //T as method parameter type.
        public void AddHead(T t)
        {
            Node n = new Node(t);
            n.Next = head;
            head = n;
        }

```

Now just select the text, and then click and drag (left mouse button) to move the cursor to the destination:



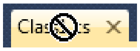
```
        public MyList()...

        //T as method parameter type.
        public void AddHead(T t)
        {
            Node n = new Node(t);
            n.Next = head;
            head = n;
        }
```

Release the mouse button to finish the move to the new location:

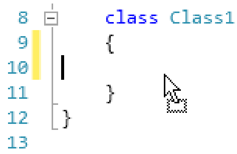


```
        //T as method parameter type.
        public void AddHead(T t)
        {
            Node n = new Node(t);
            n.Next = head;
            head = n;
        }


public MyList()
        {
            head = null;
        }

```

This technique can also be used to move text from one file to another. Just select the text, and then click and drag your mouse pointer over the tab for the new file:



Even though you get the “can’t drop” indicator, it switches to the new file. Just keep holding the mouse button down, and move the cursor to the new location:



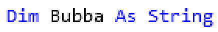
```
    class Class1
    {

    }
}

```

Then release and you are all set.

AX.59 Make Selection Uppercase or Lowercase

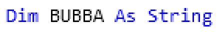
DEFAULT	Ctrl+Shift+U (uppercase); Ctrl+U (lowercase)
VISUAL BASIC 6	Ctrl+Shift+U (uppercase); Ctrl+U (lowercase)
VISUAL C# 2005	Ctrl+Shift+U (uppercase); Ctrl+U (lowercase)
VISUAL C++ 2	Ctrl+Shift+U (uppercase); Ctrl+U (lowercase)
VISUAL C++ 6	Ctrl+Shift+U (uppercase); Ctrl+U (lowercase)
VISUAL STUDIO 6	Ctrl+Shift+U (uppercase); Ctrl+U (lowercase)
WINDOWS	Alt, E, V, U (uppercase); Alt, E, V, L (lowercase)
MENU	Edit | Advanced | Make Uppercase; Edit | Advanced | Make Lowercase
COMMAND	Edit.MakeUppercase; Edit.MakeLowercase
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0044

You can easily change the case of any text:



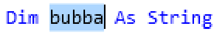
```
Dim Bubba As String
```

Just select the as much of the word you want to change case for, and press Ctrl+Shift+U to make the selected characters all uppercase:



```
Dim BUBBA As String
```

Or if you prefer, you can always make all the selected characters lowercase by pressing Ctrl+U:



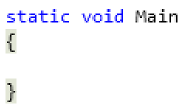
```
Dim bubba As String
```

Again, you don't have to select the whole word; you can select only the characters you want to change.

AX.60 Brace Matching Rectangle

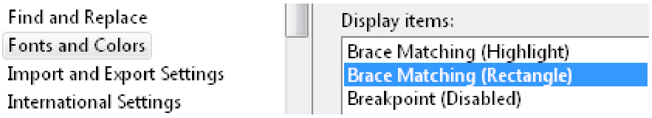
WINDOWS	Alt,T, O
MENU	Tools | Options | Fonts and Colors | Display Items
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0050

The default colors can be a pain sometimes, especially when it comes to certain colors. One of these, for me, is the Brace Matching Rectangle. The default colors are light grey:

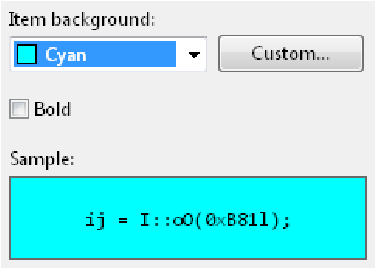


```
static void Main
{

}
```

To change the color, first go to Tools | Options | Fonts and Colors | Display Items and choose Brace Matching (Rectangle), as shown in the following illustration:



Now choose your new color (Cyan in this example):



Item background: Cyan

☐ Bold

Sample:

```
ij = I::oO(0xB81l);
```

A Note About VB

Visual Basic treats this a little differently. When you first close braces, it gives you the Brace Matching Rectangle color.

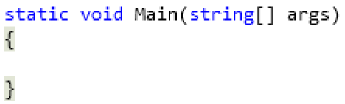
After the braces are in place, if you click next to them, it uses the highlighted reference colors.

AX.61 Automatic Delimiter Highlighting

WINDOWS	Alt,T, O
MENU	Tools | Options | Fonts and Colors | Display Items
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0071

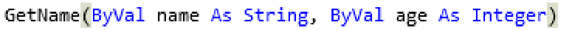
You have probably seen automatic delimiter highlighting in action before. It shows up when you close parentheses, curly brackets, and other similar delimiters.

C#



```
static void Main(string[] args)
{

}
```

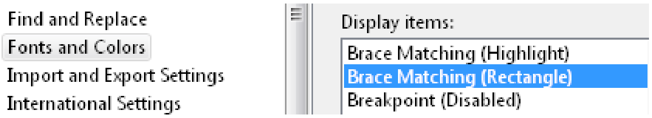
VB



```
GetName(ByVal name As String, ByVal age As Integer)
```

Changing Colors

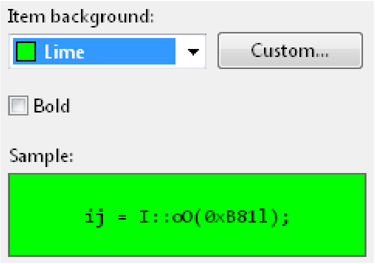
You can modify the colors to suit your need by going to Tools | Options | Environment | Fonts and Colors and selecting Brace Matching (Rectangle):



Display items:

- Brace Matching (Highlight)
- Brace Matching (Rectangle)
- Breakpoint (Disabled)

Let's change the background to lime green and click OK:



Item background: 🟩 Lime

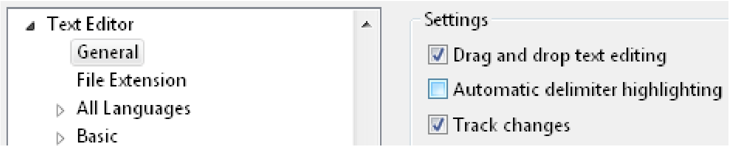
☐ Bold

Sample:

```
ij = I::oO(0xB81l);
```

Turning It Off

If you don't like this feature, you can go to Tools | Options | Text Editor | General and clear the Automatic Delimiter Highlighting check box, as shown in the following illustration:



Settings

☑ Drag and drop text editing

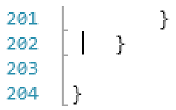
☐ Automatic delimiter highlighting

☑ Track changes

AX.62 Move or Select to the Top or Bottom of the Current View in the Editor

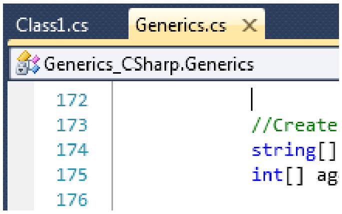
DEFAULT	Ctrl+PgUp (move top); Ctrl+PgDn (move bottom); Ctrl+Shift+PgUp (select top); Ctrl+Shift+PgDn (select bottom)
VISUAL BASIC 6	[no shortcut]
VISUAL C# 2005	Ctrl+PgUp (move top); Ctrl+PgDn (move bottom); Ctrl+Shift+PgUp (select top); Ctrl+Shift+PgDn (select bottom)
VISUAL C++ 2	Ctrl+PgUp (move top); Ctrl+PgDn (move bottom); Ctrl+Shift+PgUp (select top); Ctrl+Shift+PgDn (select bottom)
VISUAL C++ 6	Ctrl+PgUp (move top); Ctrl+PgDn (move bottom); Ctrl+Shift+PgUp (select top); Ctrl+Shift+PgDn (select bottom)
VISUAL STUDIO 6	Ctrl+PgUp (move top); Ctrl+PgDn (move bottom); Ctrl+Shift+PgUp (select top); Ctrl+Shift+PgDn (select bottom)
WINDOWS	[no shortcut]
COMMAND	Edit.ViewTop; Window.ViewBottom;Edit.ViewTopExtend;Edit.ViewBottomExtend
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0056

This tip comes in handy when you want to travel from one end of your screen to the other. For example, when you are working with documents, you might find yourself at the bottom of the screen and want to get to the top:



```
            }
        }

}
```

Just press Ctrl+PgUp, and you are taken to the top of the screen as close to the current column position as possible:



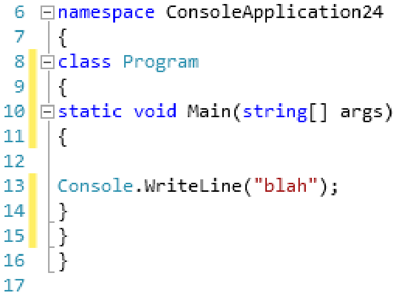
```

            //Create
            string[]
            int[] ag

```

Using Ctrl+PgDn takes you to the bottom of the screen. You can also use Ctrl+Shift+Pg[Up or Dn] to select everything from the current cursor position to the top or bottom of the screen.

AX.63 Format the Current Document or Selection

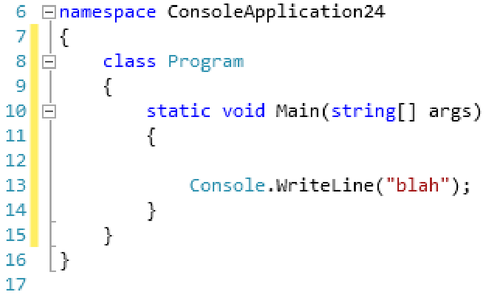
DEFAULT	Ctrl+K, Ctrl+D (document); Ctrl+K, Ctrl+F (selection)
VISUAL BASIC 6	Ctrl+K, Ctrl+D (document); Ctrl+K, Ctrl+F (selection)
VISUAL C# 2005	Ctrl+K, Ctrl+D (document); Ctrl+E, Ctrl+D (document); Ctrl+E, D (document); Ctrl+K, Ctrl+F (selection); Ctrl+E, Ctrl+F (selection); Ctrl+E, F (selection)
VISUAL C++ 2	[no shortcut] (document); Ctrl+Shift+F (selection); Ctrl+Alt+I (selection)
VISUAL C++ 6	Ctrl+K, Ctrl+D (document); Ctrl+K, Ctrl+F (selection); Alt+F8 (selection)
VISUAL STUDIO 6	Ctrl+K, Ctrl+D (document); Ctrl+K, Ctrl+F (selection); Alt+F8 (selection)
WINDOWS	Alt,E, V, A (document); Alt,E, V, F (selection)
MENU	Edit | Advanced | Format Document; Edit | Advanced | Format Selection
COMMAND	Edit.FormatDocument; Edit.FormatSelection
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0057

Let's say you have some code that isn't formatted properly.



```
namespace ConsoleApplication24
{
class Program
{
static void Main(string[] args)
{

Console.WriteLine("blah");
}
}
}

```

Now you want it to look good. Just select the code, and then go to Edit | Advanced | Format Selection to get the result shown in the following illustration:



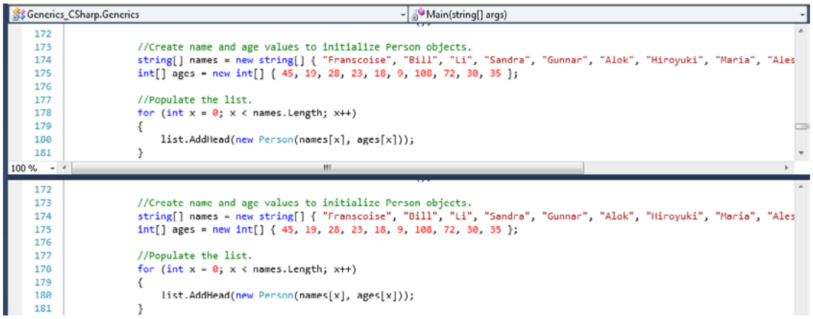
```
namespace ConsoleApplication24
{
    class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine("blah");
        }
    }
}

```

If you want to fix the formatting for the entire document you are currently working in, just go to Edit | Advanced | Format Document. This operation formats everything in the current open document for you, without your having to select specific areas.

AX.64 Use F6 to Jump Between Split Windows

DEFAULT	F6; Shift+F6
VISUAL BASIC 6	F6; Shift+F6
VISUAL C# 2005	[no shortcut]
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	F6; Shift+F6
VISUAL STUDIO 6	F6; Shift+F6
WINDOWS	[no shortcut]
COMMAND	Window.NextSplitPane; Window.PreviousSplitPane
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0005

If you have split windows (Window | Split), you can easily move the cursor between the panes without using your mouse: Just press F6:



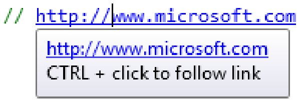
```

            //Create name and age values to initialize Person objects.
            string[] names = new string[] { "Franscoise", "Bill", "Li", "Sandra", "Gunnar", "Alok", "Hiroyuki", "Maria", "Ales
            int[] ages = new int[] { 45, 19, 28, 23, 18, 9, 108, 72, 30, 35 };

            //Populate the list.
            for (int x = 0; x < names.Length; x++)
            {
                list.AddHead(new Person(names[x], ages[x]));
            }
```

```

            //Create name and age values to initialize Person objects.
            string[] names = new string[] { "Franscoise", "Bill", "Li", "Sandra", "Gunnar", "Alok", "Hiroyuki", "Maria", "Ales
            int[] ages = new int[] { 45, 19, 28, 23, 18, 9, 108, 72, 30, 35 };

            //Populate the list.
            for (int x = 0; x < names.Length; x++)
            {
                list.AddHead(new Person(names[x], ages[x]));
            }
```

AX.65 Turn Off Single-Click URL Navigation in the Editor

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | All Languages | General | Display
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0060

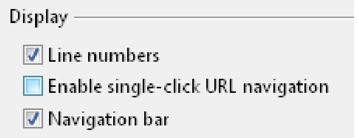
In most languages, by default, single-click URL navigation is turned on, which allows you to use Ctrl+Click to follow a link from the Editor:



```
// http://www.microsoft.com
```

http://www.microsoft.com
CTRL + click to follow link

This feature can be turned off by going to Tools | Options | Text Editor | All Languages [or your language] | General | Display and clearing the Enable Single-Click URL Navigation check box:



Display

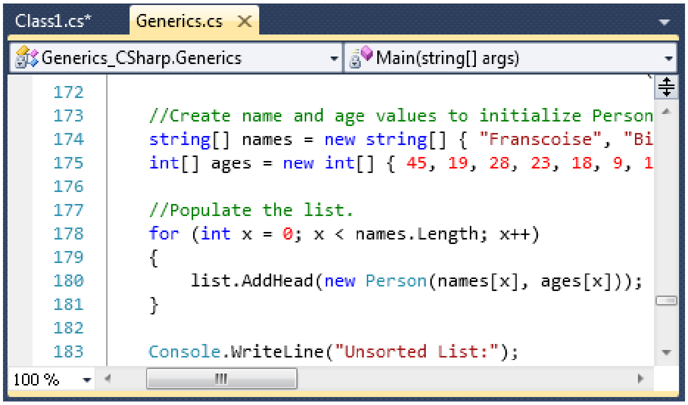
☑ Line numbers
☐ Enable single-click URL navigation
☑ Navigation bar

AX.66 Hide the Vertical and/or Horizontal Scroll Bars

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | General | Display
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0058

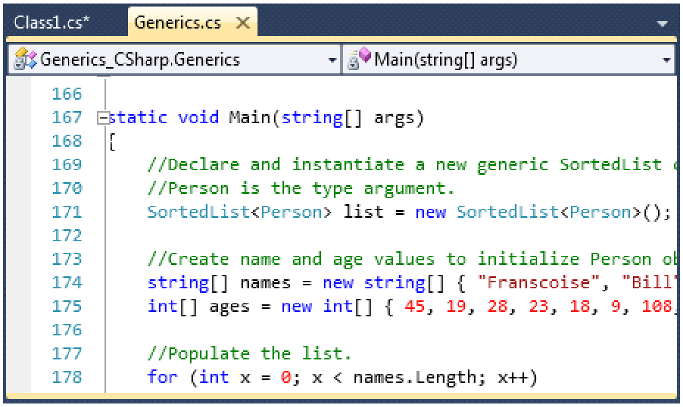
This isn't a commonly used option but can be useful in certain situations. If you want more real estate on your screen, you can go to Tools | Options | Text Editor | General | Display and clear the Vertical Scroll Bar and/or Horizontal Scroll Bar check boxes to make the scroll bars go away.

Before:



```

//Create name and age values to initialize Person
string[] names = new string[] { "Franscoise", "Bi
int[] ages = new int[] { 45, 19, 28, 23, 18, 9, 1

//Populate the list.
for (int x = 0; x < names.Length; x++)
{
    list.AddHead(new Person(names[x], ages[x]));
}

Console.WriteLine("Unsorted List:");
```

After:



```

static void Main(string[] args)
{
    //Declare and instantiate a new generic SortedList
    //Person is the type argument.
    SortedList<Person> list = new SortedList<Person>();

    //Create name and age values to initialize Person ob
    string[] names = new string[] { "Franscoise", "Bill"
    int[] ages = new int[] { 45, 19, 28, 23, 18, 9, 108

    //Populate the list.
    for (int x = 0; x < names.Length; x++)
```

Of course, you no longer have your scroll bars, which can be somewhat annoying.

AX.67 How to Convert Tabs to Spaces and Vice Versa

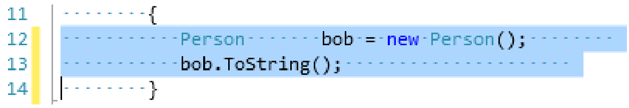
DEFAULT	[no shortcut]
VISUAL BASIC 6	[no shortcut]
VISUAL C# 2005	[no shortcut]
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	[no shortcut]
VISUAL STUDIO 6	Ctrl+Q (tabify); Ctrl+Shift+Q (untabify)
WINDOWS	Alt,E, V, T (tabify); Alt,E, V, B (untabify)
MENU	Edit | Advanced | Tabify Selected Lines; Edit | Advanced | Untabify Selected Lines
COMMAND	Edit.TabifySelectedLines; Edit.UntabifySelectedLines; Edit.ConvertSpacesToTabs; Edit.ConvertTabsToSpaces
VERSIONS	2008, 2010
CODE	vstipEdit0028

Some people prefer spaces; others prefer tabs. You can have it any way you want it with this next item. You can convert spaces to tabs and convert tabs to spaces on your selected lines. You can do this in a couple of ways, and each has different results.

Tabify/Untabify

If all you want to do is convert leading spaces to tabs (or vice versa), you would use the Tabify/Untabify commands. First, pick a line with some leading spaces, as shown in the following illustration.

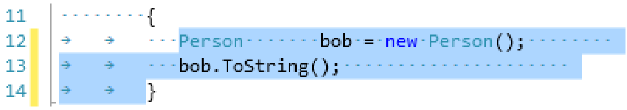
Note You don't have to select the entire line for this to work; as long as any part of the line is selected, it performs the action.



```
        {
            Person      bob = new Person();        
            bob.ToString();                    
        }
```

Now go to Edit | Advanced | Tabify Selected Lines.

You should get the leading spaces converted to tabs, as shown in the following illustration:



```
        {
		   Person      bob = new Person();        
		   bob.ToString();                    
		}
```

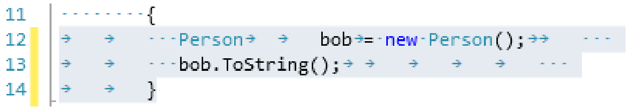
To change leading tabs to spaces, you would use the Untabify Selected Lines command.

ConvertSpacesToTabs/ConvertTabsToSpaces

OK, so what if you want to convert *all* spaces to tabs? Well, you have to use commands that have no shortcut or menu items. The commands you are interested in are Edit.ConvertTabsToSpaces and Edit.ConvertSpacesToTabs.

The following illustration shows what ConvertSpacesToTabs does to our example.

Note For these commands, you have to select everywhere you want to convert, because the command does not automatically convert the entire line.



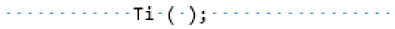
```
        {
		   Person	bob	= new Person();		   
		   bob.ToString();					   
		}
```

As you can see, almost all spaces are converted to tabs. Because spaces are converted to tabs in increments of 4 (default), if you have, say, 6 spaces, it results in a tab and 2 spaces left over. That is why you see some leftover spaces in the example. If you select these same lines and run ConvertTabsToSpaces, it inserts spaces instead of tabs.

AX.68 Delete Horizontal White Space

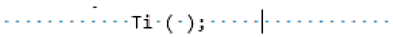
DEFAULT	Ctrl+K, Ctrl+\
VISUAL BASIC 6	Ctrl+K, Ctrl+\
VISUAL C# 2005	Ctrl+K, Ctrl+\; Ctrl+E, Ctrl+\; Ctrl+E, \
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+K, Ctrl+\
VISUAL STUDIO 6	Ctrl+K, Ctrl+\
WINDOWS	Alt,E, V, H
MENU	Edit | Advanced | Delete Horizontal White Space
COMMAND	Edit.DeleteHorizontalWhiteSpace
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0037

Want to get rid of some extra spaces? First find a line that has some extra white space that you want to get rid of:



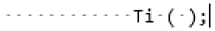
```
············Ti·(·);··················
```

Put your cursor in the extra white space:



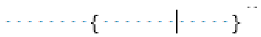
```
············Ti·(·);·····|············
```

Go to Edit | Advanced | Delete Horizontal White Space or press Ctrl+K, Ctrl+\, and the extra space is gone:



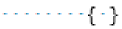
```
············Ti·(·);|
```

Interestingly, this also works between items as well. Just find some space and put your cursor in it:



```
········{·······|····}
```

Then get rid of the extra space:



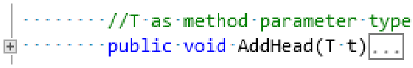
```
········{·}
```

Note One space always remains when you use this feature between items.

AX.69 Expanding Your Code with Outlining

DEFAULT	Ctrl+M, Ctrl+M
VISUAL BASIC 6	Ctrl+M, Ctrl+M
VISUAL C# 2005	Ctrl+M, Ctrl+M; Ctrl+M, M
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+M, Ctrl+M
VISUAL STUDIO 6	[no shortcut]
WINDOWS	Alt,E, O, T
MENU	Edit | Outlining | Toggle Outlining Expansion
COMMAND	Edit.ToggleOutliningExpansion
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0030

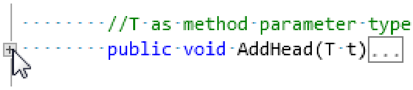
By default, outlining is enabled in Visual Studio. Suppose you encounter a collapsed area of code, as shown in the following illustration, and you want to see all the code that is collapsed in that area:



```
        //T as method parameter type.
        public void AddHead(T t)...
```

You have three ways to expand it:

- Click the plus sign to expand the area.



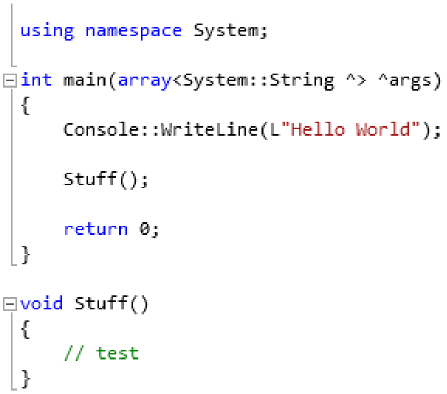
```
        //T as method parameter type.
        public void AddHead(T t)...
```

- Click anywhere in the area to be expanded, and press Ctrl+M, Ctrl+M.
- Click anywhere in the area to be expanded, and go to Edit | Outlining | Toggle Outlining Expansion on the menu bar.

AX.70 Collapsing or Expanding All Your Code with Outlining

DEFAULT	Ctrl+M, Ctrl+L
VISUAL BASIC 6	Ctrl+M, Ctrl+L
VISUAL C# 2005	Ctrl+M, Ctrl+L; Ctrl+M, L
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+M, Ctrl+L
VISUAL STUDIO 6	[no shortcut]
WINDOWS	Alt,E, O, L
MENU	Edit | Outlining | Toggle All Outlining
COMMAND	Edit.ToggleAllOutlining
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0031

You can easily collapse or expand all your code with outlining. For example, suppose you have code that is expanded, as shown in the following illustration:

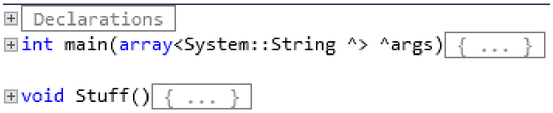


```
using namespace System;

int main(array<System::String ^> ^args)
{
    Console::WriteLine(L"Hello World");

    Stuff();

    return 0;
}

void Stuff()
{
    // test
}
```

You want it all collapsed, so you have two options:

- Press Ctrl+M, Ctrl+L.
- Go to Edit | Outlining | Toggle All Outlining on the menu bar.

The result is collapsed code:



```
⊞ Declarations
⊞int main(array<System::String ^> ^args){ ... }

⊞void Stuff(){ ... }
```

Just repeat one of the steps to reverse the process, and all your code is expanded again. This is particularly useful if you are using a feature (certain Find operations) that can't look inside collapsed code.

AX.71 Turn Off or Turn On Outlining

DEFAULT	Ctrl+M, Ctrl+P (stop outlining)
VISUAL BASIC 6	Ctrl+M, Ctrl+P (stop outlining)
VISUAL C# 2005	Ctrl+M, Ctrl+P (stop outlining); Ctrl+M, P (stop outlining)
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+M, Ctrl+P (stop outlining)
VISUAL STUDIO 6	[no shortcut]
WINDOWS	Alt,E, O, P (stop outlining); Alt, E, O, U (start outlining)
MENU	Edit | Outlining | Stop Outlining; Edit | Outlining | Start Automatic Outlining (Not Available in C++ 2005 and 2008)
COMMAND	Edit.StopOutlining; Edit.OutliningStartAutomaticOutlining
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0033

If you don't like the outlining feature in Visual Studio, you can turn it off one of two ways:

- Press Ctrl+M, Ctrl+P on your keyboard.
- Go to Edit | Outlining | Stop Outlining on your menu bar.

To turn outlining back on, go to Edit | Outlining | Start Automatic Outlining on your menu bar. Unfortunately, no keyboard shortcut is available for turning outlining back on.

Note Start Automatic Outlining is not available in C++ 2005/2008. To get it back in C++ 2005/2008, just close and reopen the file you are working on.

AX.72 Understanding Virtual Space

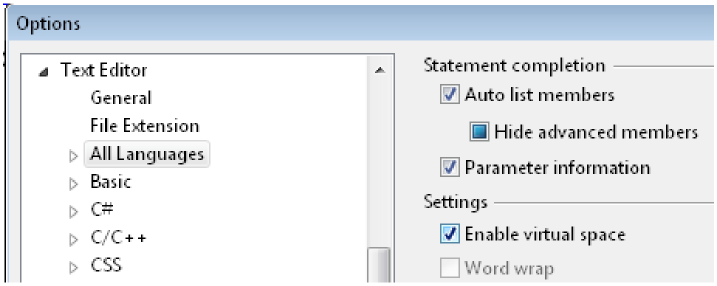
WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | All Languages | General | Settings
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0023

Virtual space is a little difficult to understand if you aren't familiar with older editors. We used to have (and some people still have) editors that treat everywhere on a line as editable space.

Let me explain: Without virtual space, the line ends where the code ends.

If I move my cursor to the end of any line and press my Right Arrow key, it goes to the next line. This is the way editors have been for a while now, and this isn't really new information.

However, this wasn't always the case. There was a time when you could type anywhere you wanted, anytime you wanted, without restriction. Some text editors still allow this today. Virtual space allows you to go back to the old style of editing, which is preferred by some. Go to Tools | Options | Text Editor | All Languages | General | Settings, and select the Enable Virtual Space check box to turn this feature on:



Statement completion

☑ Auto list members

■ Hide advanced members

☑ Parameter information

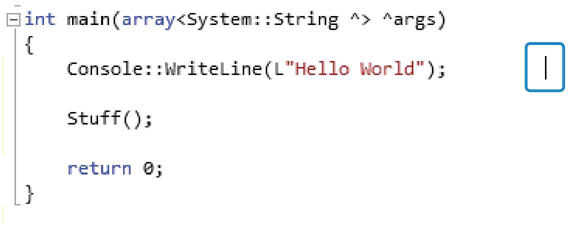
Settings

☑ Enable virtual space

☐ Word wrap

Note Enable Virtual Space and Word Wrap are mutually exclusive options, so you have to choose one or the other.

After you select the Enable Virtual Space option, you can type anywhere on a line, regardless of whether or not the code ends:

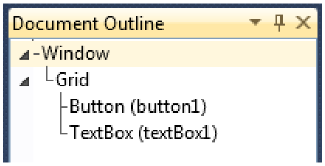


```
int main(array<System::String ^> ^args)
{
    Console::WriteLine(L"Hello World");

    Stuff();

    return 0;
}
```

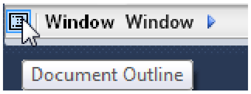
AX.73 Document Outline: WPF and Silverlight Projects

DEFAULT	Ctrl+Alt+T
VISUAL BASIC 6	Ctrl+Alt+T
VISUAL C# 2005	Ctrl+Alt+T; Ctrl+W, Ctrl+U; Ctrl+W, U
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+Alt+D
VISUAL STUDIO 6	Ctrl+Alt+T
WINDOWS	Alt,V, E, D
MENU	View | Other Windows | Document Outline
COMMAND	View.DocumentOutline
VERSIONS	2010
LANGUAGES	C#, VB
CODE	vstipTool0117

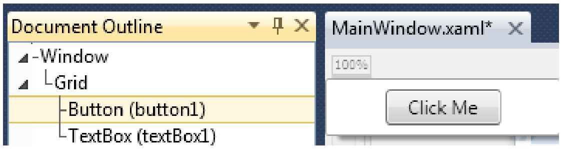
When working with WPF and XAML, it can sometimes get tricky finding items. This is where the Document Outline feature comes in handy. In this example, I've created a WPF Application and put a few controls on it. The following illustration shows what I get when I pull up the Document Outline (Ctrl+Alt+T):



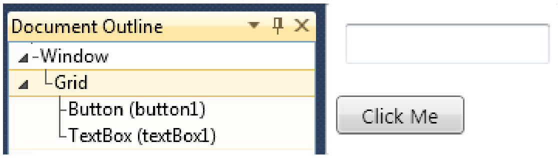
You can also get to the Document Outline by clicking the Document Outline button located in the lower-left corner of the screen by default:



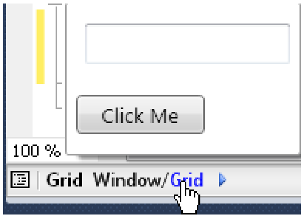
Notice how it shows each control and the parent/child relationships. If the experience stopped there, it would be OK, but it actually gets even better when you put your mouse pointer over any item in the list, as shown in the following illustration:



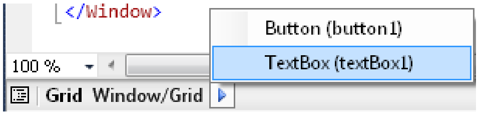
Putting the mouse pointer over a parent shows a preview of the parent and all the children:



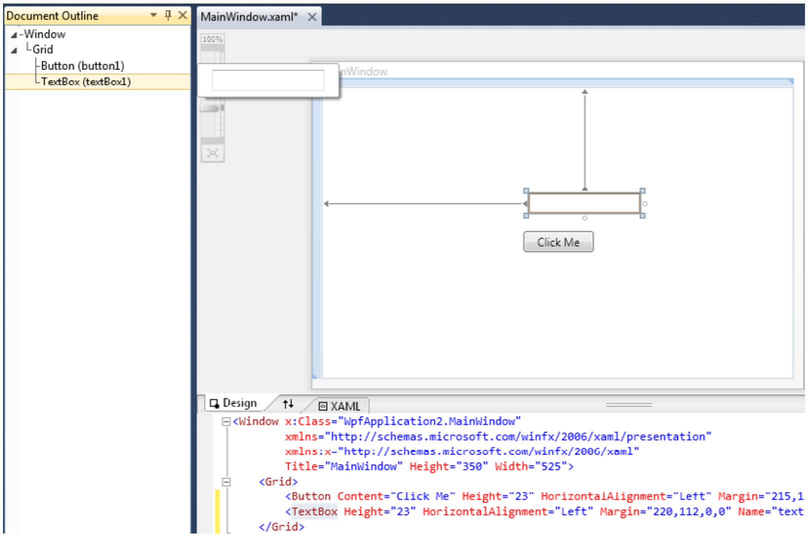
You can also get a preview from the outline presented at the lower part of the screen:



And you can dig into the details if needed:



Also, when you click on any item in the Document Outline, it is selected in both XAML and Design view:



```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Click Me" Height="23" HorizontalAlignment="Left" Margin="215,1
        <TextBox Height="23" HorizontalAlignment="Left" Margin="220,112,0,0" Name="text
    </Grid>
```

To sum it up, the Document Outline can be used when working with WPF to do the following:

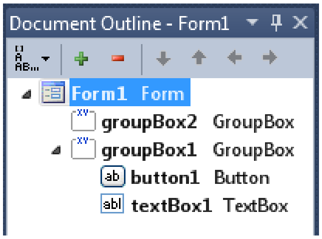
- View the logical structure of elements in your XAML.
- View a thumbnail preview of an element in a pop-up window.
- Navigate to specific elements, in Design view and in XAML view.
- Put user input focus on deeply nested elements that might be hard to select on the design surface itself.
- Locate controls that might be visually hidden by other controls.

AX.74 Document Outline: Windows Form Projects

DEFAULT	Ctrl+Alt+T
VISUAL BASIC 6	Ctrl+Alt+T
VISUAL C# 2005	Ctrl+Alt+T; Ctrl+W, Ctrl+U; Ctrl+W, U
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+Alt+D
VISUAL STUDIO 6	Ctrl+Alt+T
WINDOWS	Alt,V, E, D
MENU	View | Other Windows | Document Outline
COMMAND	View.DocumentOutline
VERSIONS	2005, 2008, 2010
CODE	vstipTool0118

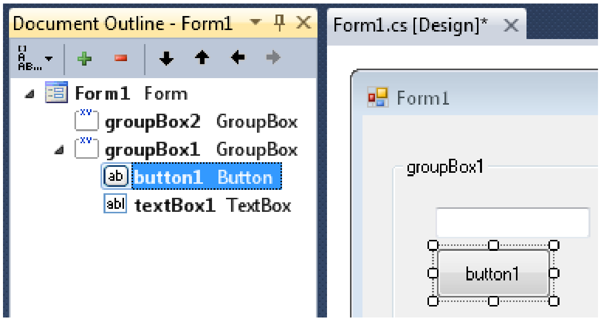
The Document Outline is used to get a bird's eye view of items in your project. Let's look at using it with Windows Form projects.

For this example, I've created a new Windows Forms Application and put a few sample controls on it. The following illustration shows what the Document Outline (Ctrl+Alt+T) looks like:



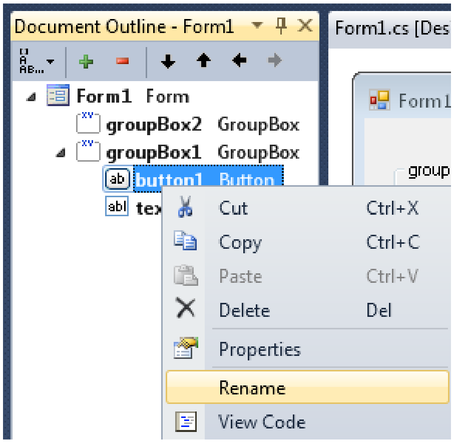
Selection

When you click any item in the list, that item is selected on the form in Design view:



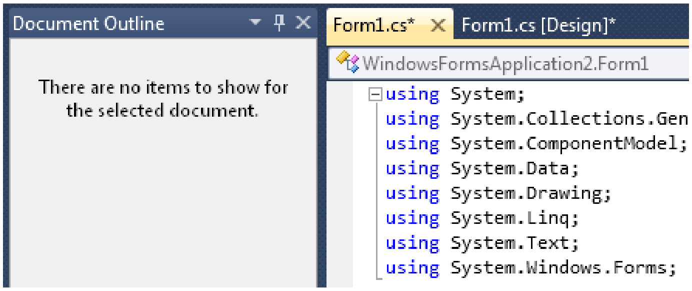
Context Commands

Additionally, you can access a variety of commands by right-clicking any item, including the ability to rename the control from the Document Outline:



View Code

You can also press F7 with any item selected to view the code for it, but remember that the Document Outline does not work in code view:

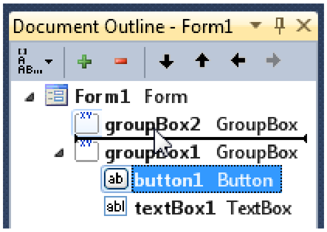


There are no items to show for the selected document.

```
using System;
using System.Collections.Gen
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

Relocate Items

The ability to move items from one container to another is supported as well:

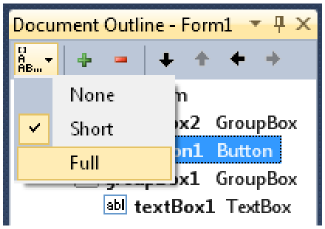


Toolbar Controls

The toolbar supports a variety of functions as described in the following sections.

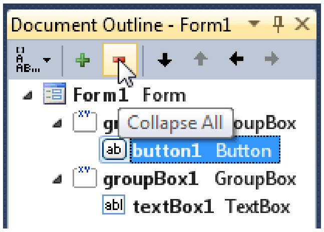
Name display

You can select different name display styles:



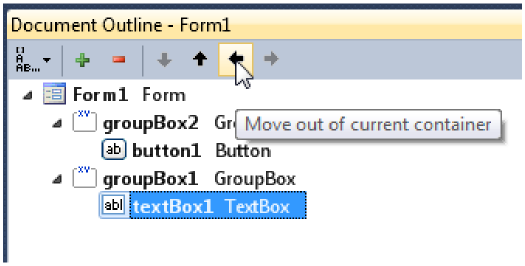
Expand/collapse

You can expand or collapse the entire outline:



Moving around

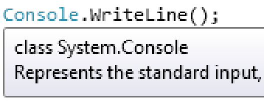
The toolbar even supports moving around within and between containers:



AX.75 Change the Tooltip Font Size

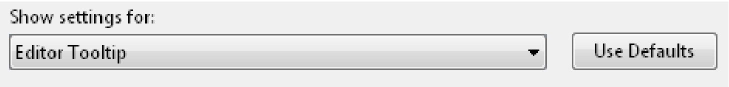
WINDOWS	Alt,T, O
MENU	Tools | Options | Fonts and Colors | Show settings For
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0046

Ever want to change the size of your tooltip font? Here is what it looks like by default:



```
Console.WriteLine();
class System.Console
Represents the standard input,
```

To change it, just go to Tools | Options | Fonts and Colors | Show Settings For. From there, select Editor Tooltip from the drop-down list, as shown in the following illustration:



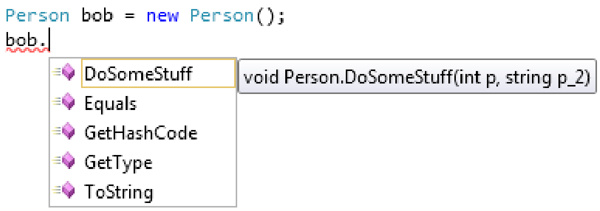
Show settings for:
Editor Tooltip

Now that you have the Editor Tooltip settings, change the font to a bigger size and then click OK. Now you should see a bigger font size on your tooltips.

AX.76 Change the Statement Completion Font Size

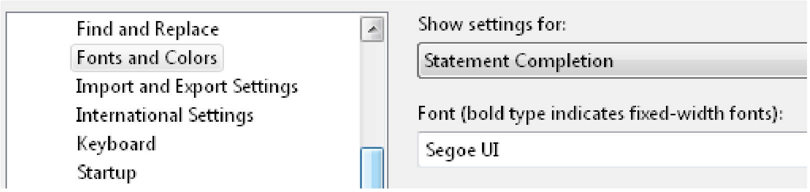
WINDOWS	Alt,T, O
MENU	Tools | Options | Fonts and Colors
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0055

Are you like me and think that the statement completion font is annoyingly small and hard to read?



```
Person bob = new Person();
bob.
```

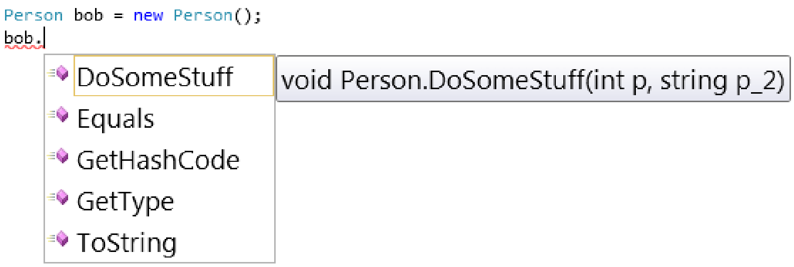
Well, you can easily increase the font size by going to Tools | Options | Fonts And Colors | Show Settings For | Statement Completion, as shown in the following illustration:



Show settings for: Statement Completion

Font (bold type indicates fixed-width fonts): Segoe UI

From this dialog box, just make the font any size you like. Click OK to exit the dialog box, and now you should have statement completion in a size you can read:

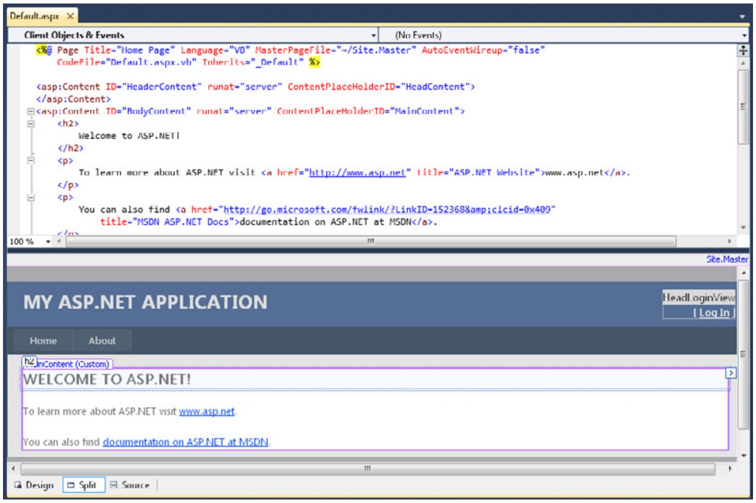


```
Person bob = new Person();
bob.
```

AX.77 Vertical Split View for Web Projects

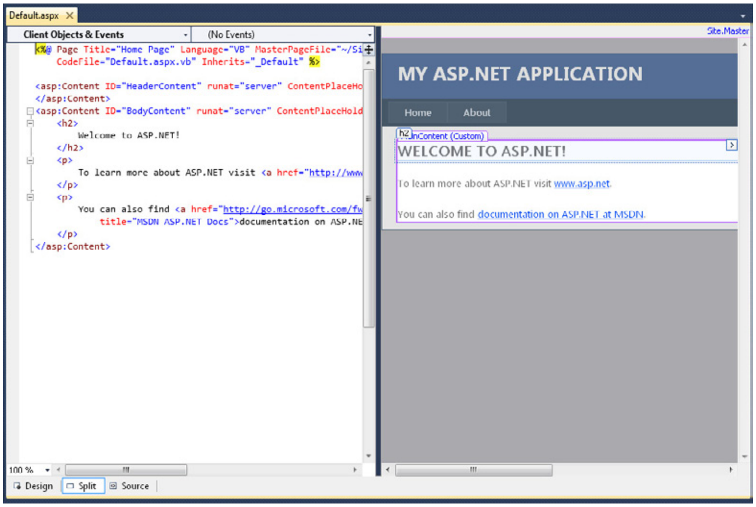
WINDOWS	Alt,T, O
MENU	Tools | Options
COMMAND	Tools.Options
VERSIONS	2008, 2010
CODE	vstipEdit0081

The default way Split view handles panes is to tile them horizontally:



```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master" AutoEventWireup="false"
    CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www.asp.net" title="ASP.NET Website">www.asp.net</a>.
    </p>
    <p>
        You can also find <a href="http://go.microsoft.com/fwlink/?LinkID=152368&amp;clcid=0x409"
            title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
```

However, you can change this by going to Tools | Options | HTML Designer | General and selecting Split Views Vertically. Now the Split view tiles the panes vertically:



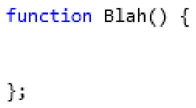
```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Si
    CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHo
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHold
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www
    </p>
    <p>
        You can also find <a href="http://go.microsoft.com/fw
            title="MSDN ASP.NET Docs">documentation on ASP.NE
    </p>
</asp:Content>
```

Note When you perform this action, you might have to close and reopen some files to see it take effect.

AX.78 Open JScript Braces on a New Line

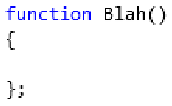
WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | JScript | Formatting
COMMAND	Tools.Options
VERSIONS	2008, 2010
CODE	vstipEdit0087

By default, Visual Studio formats JScript functions and control blocks with the open brace on the same line as the declaration:



```
function Blah() {


};
```

Without getting into the big debate about whether this is good or bad, let's assume you prefer to have your braces inline vertically:



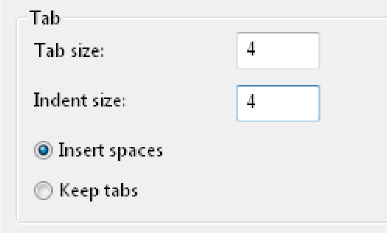
```
function Blah()
{

};
```

Go to Tools | Options | Text Editor | JScript | Formatting, and choose whether you want the open brace on a new line for functions, control blocks, or both.

AX.79 Insert Spaces vs. Keep Tabs

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | [Language] | Tab
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0072

Some people like tabs in their code, and others are partial to spaces. You can specify what you want by going to Tools | Options | Text Editor | [Language] | Tab:



Tab

Tab size: 4

Indent size: 4

● Insert spaces

○ Keep tabs

Tab Size

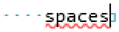
Use this setting to specify the distance in spaces between tab stops. The default is four spaces. Every time you hit the Tab key, it advances the number of spaces specified.

Indent Size

Use this setting to specify the size in spaces of an automatic indentation. The default is four spaces. Tab characters, space characters, or both are inserted according to the specified size. When the editor automatically indents your code, it uses this setting to determine how much space to use.

Insert Spaces

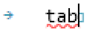
Indent operations insert only space characters, not Tab characters. For example, if the indent size is set to 5, five space characters are inserted whenever you press the Tab key or click the Increase Indent button on the formatting toolbar:



```
····spaces
```

Keep Tabs

Indent operations insert Tab characters. Each Tab character comprises the number of spaces specified in the Tab Size setting. If the indent size is not an even multiple of the Tab size, space characters are added to make up the difference:



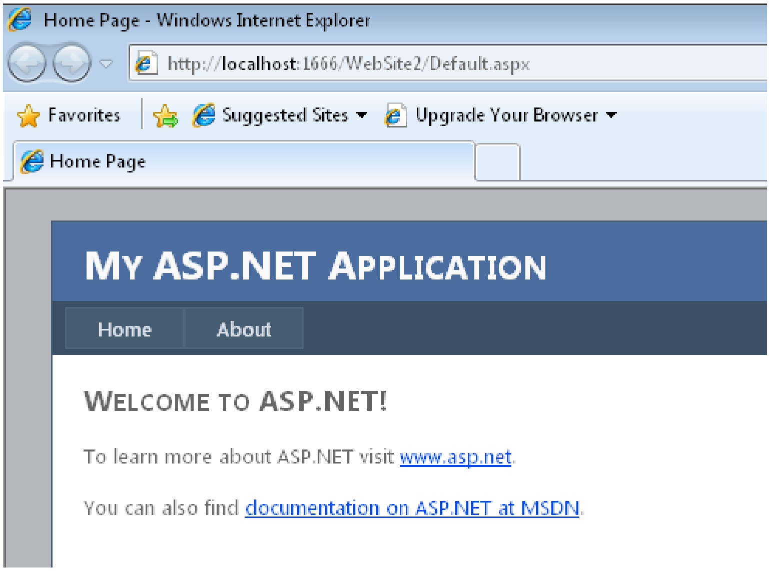
```
→   tab
```

AX.80

View in Browser

DEFAULT	Ctrl+Shift+W
VISUAL BASIC 6	Ctrl+Shift+W
VISUAL C# 2005	[no shortcut]
VISUAL C++ 2	Ctrl+Shift+W
VISUAL C++ 6	Ctrl+Shift+W
VISUAL STUDIO 6	Ctrl+Shift+W
WINDOWS	Alt,F, B
MENU	File | View in Browser
COMMAND	File.ViewinBrowser
VERSIONS	2005, 2008, 2010
CODE	vstipTool0119

You can quickly view your current page in your browser by pressing Ctrl+Shift+W. This automatically opens up your default browser (see vstipEnv0057, “Using Additional Browsers for Web Development,” page 96):


Home Page - Windows Internet Explorer
http://localhost:1666/WebSite2/Default.aspx
Favorites
Suggested Sites
Upgrade Your Browser
Home Page
Home
About

My ASP.NET Application

Welcome to ASP.NET!

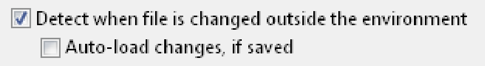
To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

AX.81 Detect When a File Is Changed Outside the Environment

WINDOWS	Alt,T, O
MENU	Tools | Options Environment | Documents
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0073

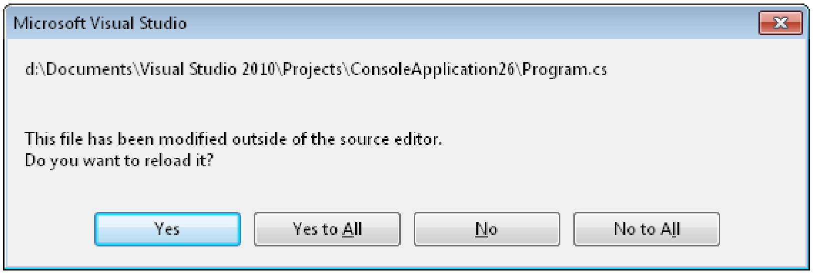
If you go to Tools | Options | Environment | Documents, you can see two interesting options:



☑ Detect when file is changed outside the environment
  ☐ Auto-load changes, if saved

Detect When File Is Changed Outside The Environment

When this option is selected, a message immediately notifies you of changes to an open file that have been made by an editor outside the IDE. The message enables you to reload the file from storage:



d:\Documents\Visual Studio 2010\Projects\ConsoleApplication26\Program.cs

This file has been modified outside of the source editor.
Do you want to reload it?

Auto-Load Changes, If Saved

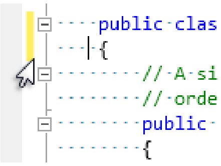
When you have the Detect When File Is Changed Outside The Environment options selected and an open file in the IDE changes outside the IDE, a warning message is generated by default. However, if the Auto-Load Changes, If Saved option is selected, no warning appears and the document is reloaded in the IDE to pick up the external changes.

As you can see, the default is to detect the changes but not auto-load them. This is generally a good strategy because any changes you load from outside the editor should be reviewed, to avoid erasing work you have already in the editor.

AX.82 Turn Off the Selection Margin

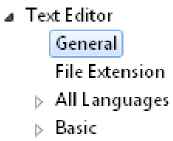
WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0034

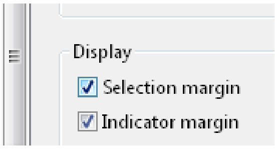
For those who aren't familiar with it, the selection margin is the area between line numbers and the outline indicators. It is used to show code changes and to enable you to select an entire line of code with one click. The following illustration shows the area in action:



```
    public clas
    {
        // A si
        // orde
        public
        {
```

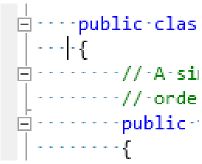
If you don't have use for the change tracking and line selection, you can easily turn this feature off by going to Tools | Options | Text Editor | General | Display and clearing the Selection Margin check box:





☑ Selection margin
☑ Indicator margin

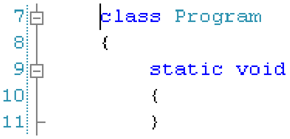
Now the selection margin is gone:



```
    public clas
    {
        // A si
        // orde
        public
        {
```

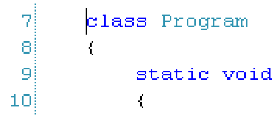
An interesting side effect in Visual Studio 2005 and Visual Studio 2008 is that hiding the selection margin also hides the outline area.

Before:



```
class Program
{
    static void
    {
    }
```

After:



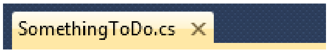
```
class Program
{
    static void
    {
```

AX.83 Reuse the Same Editor Window When Opening Files

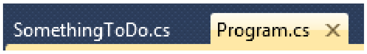
WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | Documents
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0027

Normally, when you open up a document (in this example “Program.cs”), it creates a new tab.

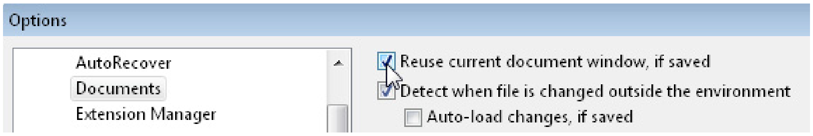
Before:



After:



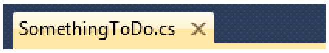
You can reuse the same document window if you want by going to Tools | Options | Environment | Documents and selecting the Reuse Current Document Window, If Saved check box:



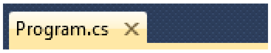
☑ Reuse current document window, if saved
☑ Detect when file is changed outside the environment
☐ Auto-load changes, if saved

Now when you open a document, you see the following results.

Before:



After:



The caveat here is that you must have a saved document for this to work. If the document was not saved, a new document window would be created even with this option turned on.

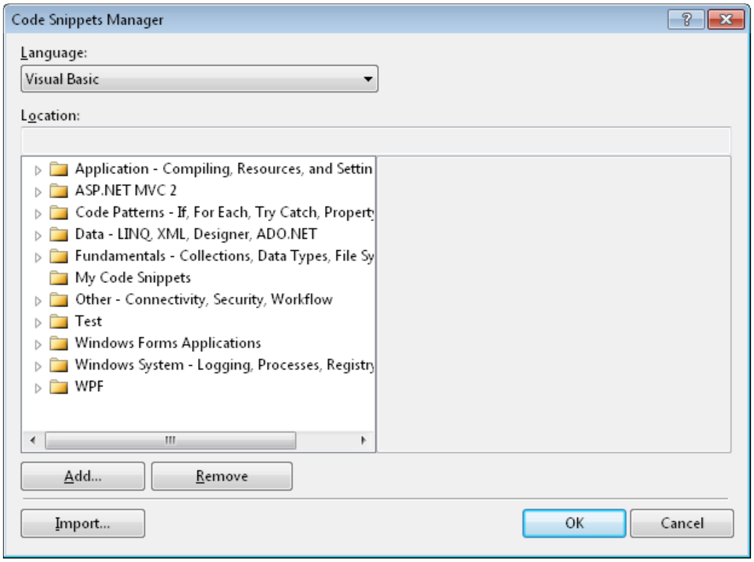
AX.84 Sharing Snippets with Your Team

DEFAULT	Ctrl+K, Ctrl+B
VISUAL BASIC 6	Ctrl+K, Ctrl+B
VISUAL C# 2005	Ctrl+K, Ctrl+B
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+K, Ctrl+B
VISUAL STUDIO 6	Ctrl+K, Ctrl+B
WINDOWS	Alt,T, T
MENU	Tools | Code Snippets Manager
COMMAND	Tools.CodeSnippetsManager
VERSIONS	2005, 2008, 2010
CODE	vstipTool0075

Sometimes you might want to share special snippets with your team for others to use. It’s easy to do, and it’s a great way to ensure common code constructs remain the same.

First create a network directory that all the team members can read from, and put all your special .snippet files there. If you don’t have experience working with snippet files, take a look at vstipTool0016, “Create New Code Snippets from Existing Ones,” page 271.

Next have each team member open the Code Snippets Manager (Ctrl+K, Ctrl+B):

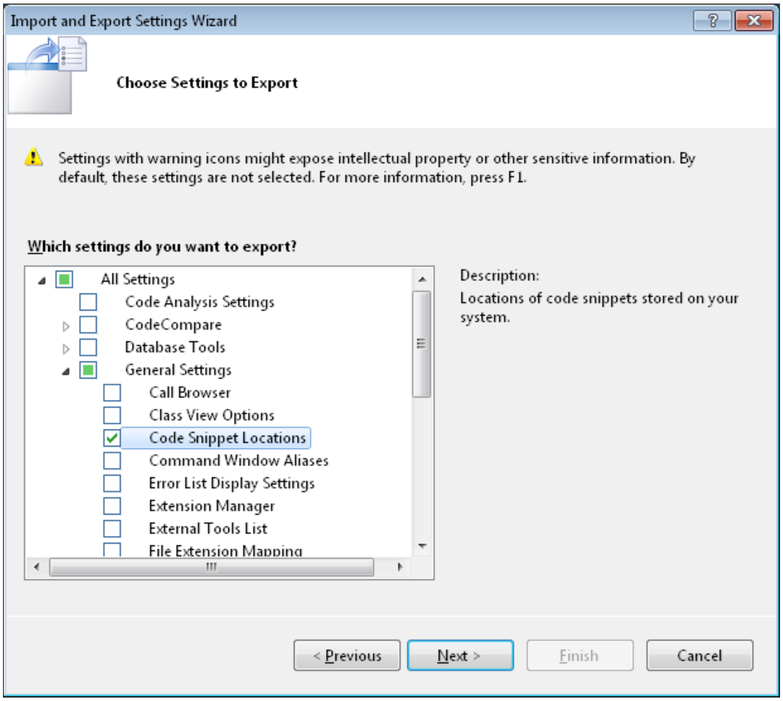


Code Snippets Manager
Location:
Application - Compiling, Resources, and Settin
ASP.NET MVC 2
Code Patterns - If, For Each, Try Catch, Propert
Data - LINQ, XML, Designer, ADO.NET
Fundamentals - Collections, Data Types, File Sy
My Code Snippets
Other - Connectivity, Security, Workflow
Test
Windows Forms Applications
Windows System - Logging, Processes, Registry
WPF
Add...
Remove
Import...
OK
Cancel

Language: Visual Basic

Ask your team members to click Add and select the network directory you created earlier, and everyone can use the shared snippets.

If you don't want to have everyone doing this manually, you can do these steps yourself and export just your code snippet locations, as shown in the following illustration:



Settings with warning icons might expose intellectual property or other sensitive information. By default, these settings are not selected. For more information, press F1.

☑ Code Snippet Locations

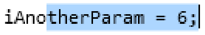
Description:
Locations of code snippets stored on your system.

Then have everyone import the .vssettings file you created to get the shared location. For more information about exporting, see vstipEnv0021, “Exporting Your Environment Settings,” on page 6.

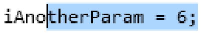
AX.85 Swap the Current Anchor Position

DEFAULT	Ctrl+K, Ctrl+A
VISUAL BASIC 6	Ctrl+K, Ctrl+A
VISUAL C# 2005	Ctrl+K, Ctrl+A; Ctrl+E, Ctrl+A; Ctrl+E, A
VISUAL C++ 2	Ctrl+Shift+X
VISUAL C++ 6	Ctrl+K, Ctrl+A
VISUAL STUDIO 6	Ctrl+K, Ctrl+A
WINDOWS	[no shortcut]
COMMAND	Edit.SwapAnchor
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0068

When you select text, the “anchor” is the cursor location at the end of the selection. By default, the anchor is to the right of the selection. However, you can use Ctrl+K, Ctrl+A to swap the anchor from the right side to the left side, as shown in the following illustrations:

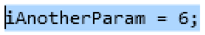


```
iAnotherParam = 6;
```



```
iAnotherParam = 6;
```

OK, so why would you want to do this? Well, those who use Emacs like this feature quite a bit, and it has been around a long, long time. One great benefit comes when you need to expand a selection from left to right. By swapping the anchor, you can hold down your Shift key and use your Left Arrow key to expand the selection:

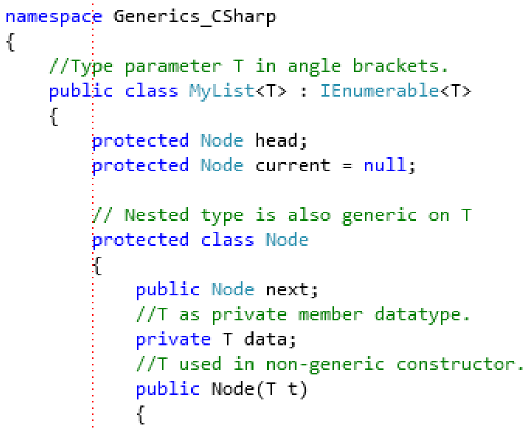


```
iAnotherParam = 6;
```

AX.86 Guidelines: A Hidden Feature for the Visual Studio Editor

VERSIONS	2005, 2008, 2010
CODE	vstipEdit0015

Guidelines are used when you want visible column indicators to help you keep things lined up in the editor:



```
namespace Generics_CSharp
{
    //Type parameter T in angle brackets.
    public class MyList<T> : IEnumerable<T>
    {
        protected Node head;
        protected Node current = null;

        // Nested type is also generic on T
        protected class Node
        {
            public Node next;
            //T as private member datatype.
            private T data;
            //T used in non-generic constructor.
            public Node(T t)
            {
```

Visual Studio 2010

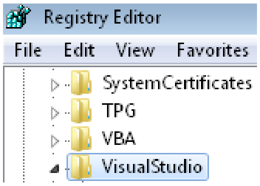
This feature has been removed in Visual Studio 2010. With that said, if you are using Visual Studio 2010, an extension, created by Paul Harrington, is available at *http://visualstudiogallery.msdn.microsoft.com/en-us/0fbf2878-e678-4577-9fdb-9030389b338c*.

Visual Studio 2005/2008

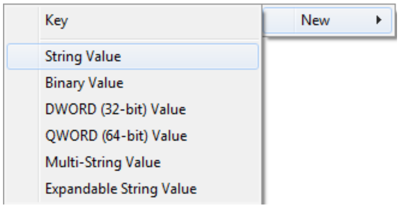
To add the guidelines to the user interface, you need to follow these steps:

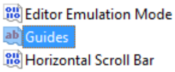
1. Shut down Visual Studio.
2. Go to [HKEY_CURRENT_USER]\Software\Microsoft\VisualStudio\<version>\Text Editor in the registry (regedit.exe).

Warning Editing the registry can cause serious problems if you don't know what you are doing, so edit it at your own risk.



3. Create a string value called Guides:

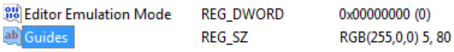




4. **Set Guides to the following:**

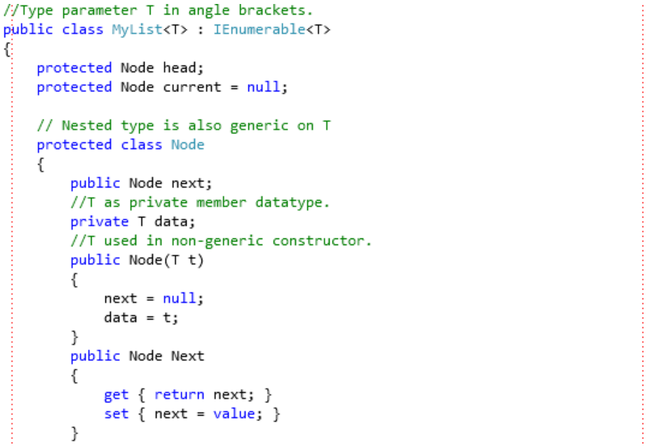
   RGB(x,y,z) n1,…,n13, where x,y,z are the RGB color values representing the color you want for the guides; and n is the column number position at which you want the guides to appear.

   You can have at most 13 guidelines. For example, RGB(255,0,0) 5, 80 places a red guideline at column numbers 5 and 80:



Editor Emulation Mode	REG_DWORD	0x00000000 (0)
Guides	REG_SZ	RGB(255,0,0) 5, 80

5. **Open Visual Studio, and open a file in the Editor:**



```
//Type parameter T in angle brackets.
public class MyList<T> : IEnumerable<T>
{
    protected Node head;
    protected Node current = null;

    // Nested type is also generic on T
    protected class Node
    {
        public Node next;
        //T as private member datatype.
        private T data;
        //T used in non-generic constructor.
        public Node(T t)
        {
            next = null;
            data = t;
        }
        public Node Next
        {
            get { return next; }
            set { next = value; }
        }
```

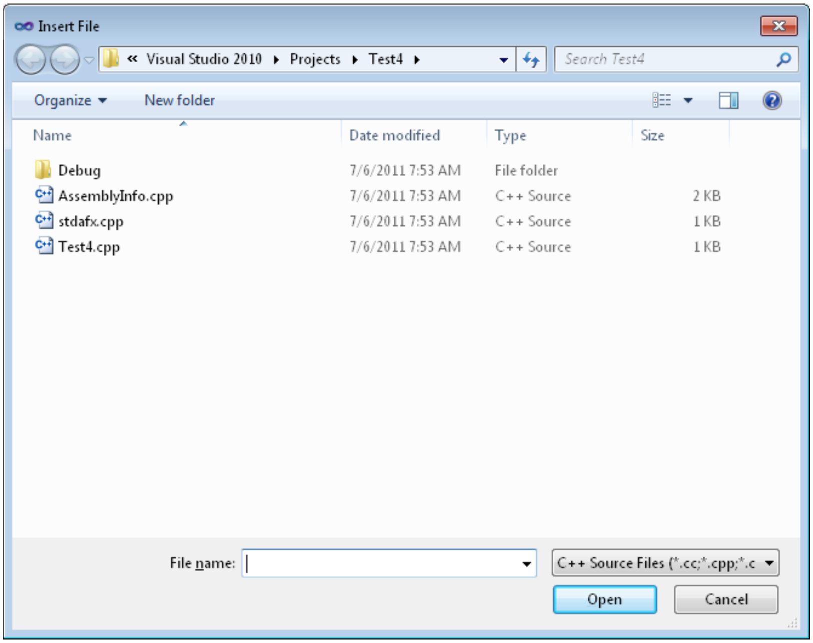
Removing guidelines

To delete the guidelines, just delete the Guides value you created above and then close and reopen Visual Studio.

AX.87 Insert File as Text

WINDOWS	Alt,E, X
MENU	Edit | Insert File As Text
COMMAND	Edit.InsertFileAsText
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0021

Another classic item that tends to get overlooked in the wake of snippets and T4 templates is the Insert File As Text feature. Let's say you have a chunk of code in a file, and you want it in another file too. Just go to Edit | Insert File As Text to see the following dialog box:



Name	Date modified	Type	Size
Debug	7/6/2011 7:53 AM	File folder	
AssemblyInfo.cpp	7/6/2011 7:53 AM	C++ Source	2 KB
stdafx.cpp	7/6/2011 7:53 AM	C++ Source	1 KB
Test4.cpp	7/6/2011 7:53 AM	C++ Source	1 KB

C++ Source Files (*.cc;*.cpp;*.c

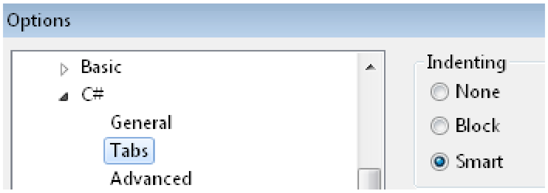
Choose your file, and it inserts the contents of that file as though you had just typed the text in yourself.

> **Note** When performing this action, you might need to change the file type to look for.

AX.88 Indenting: Smart vs. Block vs. None

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | [Language] | Tabs | Indenting
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0049

Indenting is something we all deal with in the Editor all the time. In this tip, I reveal what each of the available indenting choices does for you. If you go to Tools | Options | Text Editor | [Language] | Tabs | Indenting, you can see the options shown in the following illustration:

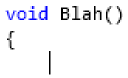

Options

Basic
C#
General
Tabs
Advanced

Indenting
○ None
○ Block
● Smart

Smart

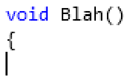
Let's begin with the option that most people have by default: Smart indenting. Create a new method, and press Enter after the curly brace:


```
void Blah()
{
    |
```

Notice that Smart indenting pays attention to where it is and automatically indents after the opening curly brace. This is the "smart" part of the indenting because it knew that you pressed Enter after an opening curly brace and assumed you wanted to indent.

Block

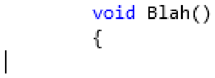
Now let's look at Block indenting:


```
void Blah()
{
|
```

In this case, the cursor maintains its current indent level and doesn't "smart" indent based on context. Block indenting is the old indenting style that is preferred by people who want control over when indenting happens.

None

Of course, if you really want total control, you can choose None in the Options dialog box and turn off any indenting at all:



```
        void Blah()
        {
|
```

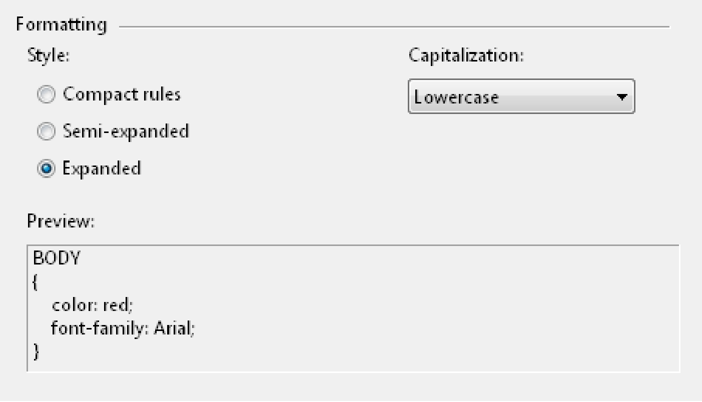
The cursor returns to column 1 every time you press Enter.

AX.89 Change CSS Formatting

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | CSS | Formatting
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0094

People are very picky about how their code is styled. Fortunately, Visual Studio offers you the ability to format your CSS code the way you like it. Just go to Tools | Options | Text Editor | CSS | Formatting:

Note In Visual Studio 2005, the menu path is Tools | Options | Text Editor | CSS | Format.



Formatting

Style:

○ Compact rules

○ Semi-expanded

● Expanded

Capitalization: Lowercase

Preview:

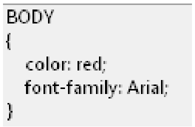
```
BODY
{
   color: red;
   font-family: Arial;
}
```

Style

You have three style options available:

Expanded (Default)

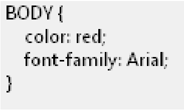
Provides the most readability by adding extra space in the styles. The selector and initial brace appear on their own lines, declarations are indented on subsequent lines, and the closing brace is aligned with the matching opening brace:



```
BODY
{
   color: red;
   font-family: Arial;
}
```

Semi-expanded

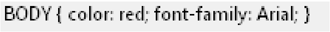
Provides a trade-off between readability and compactness by reducing space. The selector and initial brace ({) are positioned on the same line, declarations are indented on subsequent lines, and the closing brace (}) is aligned with the matching opening brace:



```
BODY {
   color: red;
   font-family: Arial;
}
```

Compact rules

Provides maximum amount of reduced space. The selector and declaration are positioned on the same line:

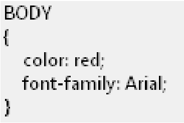


```
BODY { color: red; font-family: Arial; }
```

Capitalization

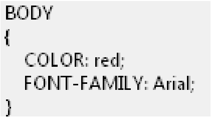
This option is pretty straightforward and provides casing instructions for the properties.

Lowercase (Default)



```
BODY
{
   color: red;
   font-family: Arial;
}
```

Uppercase



```
BODY
{
   COLOR: red;
   FONT-FAMILY: Arial;
}
```

As entered

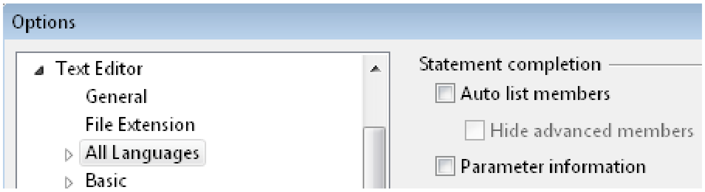
Leaves the casing alone and doesn't modify the user input.

AX.90 How to Turn Off Automatic IntelliSense

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | All Languages [or specific language] | General | Statement completion
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0054

So IntelliSense might not be for everyone. It can sometimes annoy people when it automatically pops up. You can disable automatic IntelliSense and still have it come up only when you want it to.

Go to Tools | Options | Text Editor | All Languages [or specific language] | General | Statement Completion, and clear the Auto List Members check box:



Statement completion
☐ Auto list members
☐ Hide advanced members
☐ Parameter information

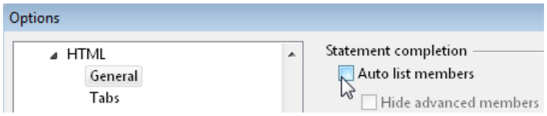
Now anytime you want IntelliSense, just press Ctrl+J to bring it up.

AX.91 Disable HTML, CSS, or JScript IntelliSense

WINDOWS	Alt,T, O
MENU	Tools | Options | Text Editor | HTML | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0085

Personally, I love IntelliSense everywhere I write code.

However, some people don't like it in some places, and one place I find where that is especially true is in the HTML editor. You can turn off IntelliSense by going to Tools | Options | Text Editor | HTML | General and clearing the Auto List Members check box:



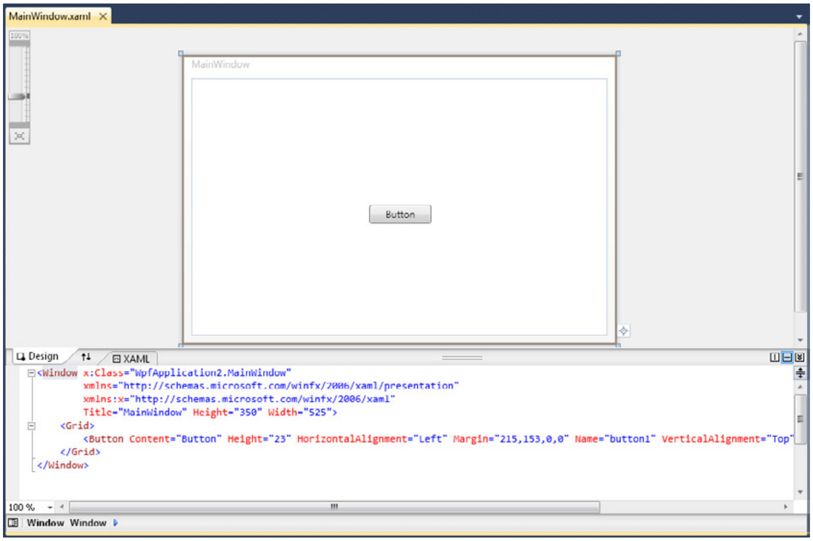
☐ Auto list members
☐ Hide advanced members

This action disables IntelliSense for the HTML text editor. Additionally, if you want to do the same for CSS or Jscript, just go to Tools | Options | Text Editor | [CSS or JScript] | General and perform the same step.

AX.92 Design and XAML on Different Document Tabs

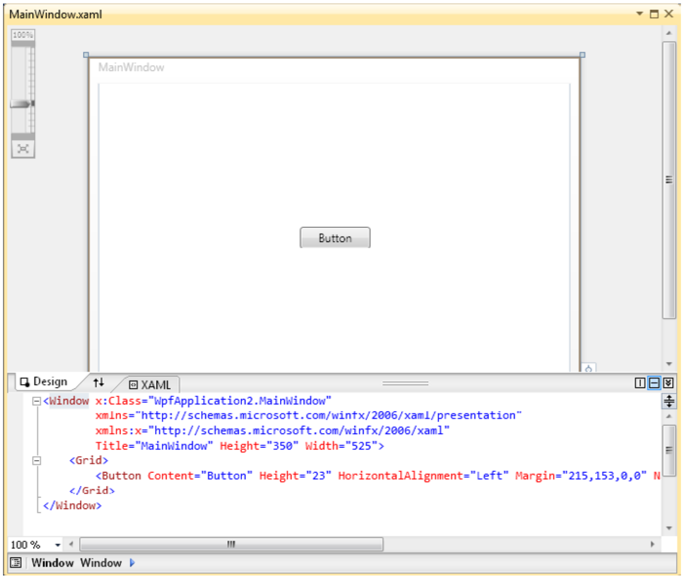
VERSIONS	2010
CODE	vstipTool0009

Now that you can free your document windows and put them on multiple monitors, many people are asking how to do this with code-behind files. Assume you have the following XAML document open:



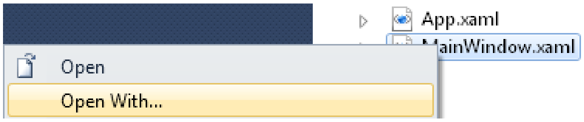
```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Button" Height="23" HorizontalAlignment="Left" Margin="215,153,0,0" Name="button1" VerticalAlignment="Top"
    </Grid>
</Window>
```

But you want the Design view in one window and the XAML view in another so that you can work on each document in a separate monitor. Well, if you click and drag the document tab, both the design and the XAML go together:

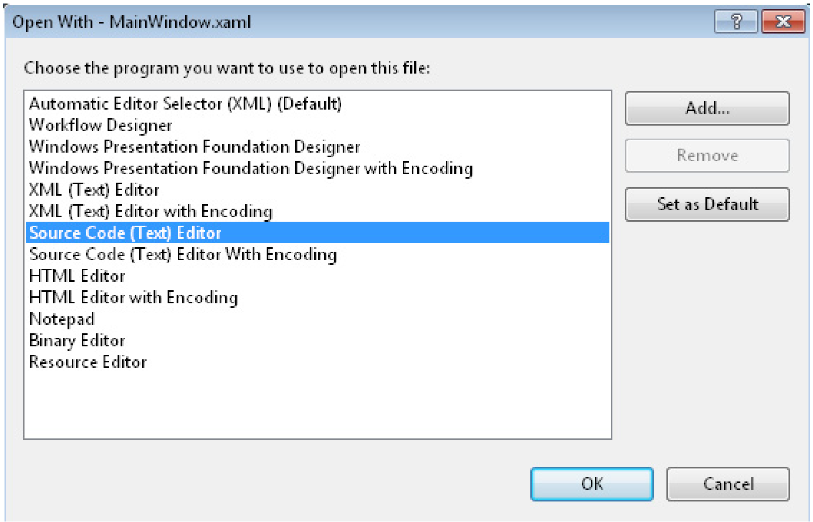


```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Button" Height="23" HorizontalAlignment="Left" Margin="215,153,0,0" N
    </Grid>
</Window>
```

So what do you do? As it turns out, the solution is pretty easy. With the design view in one document, go to Solution Explorer, right-click the XAML file, and choose Open With:



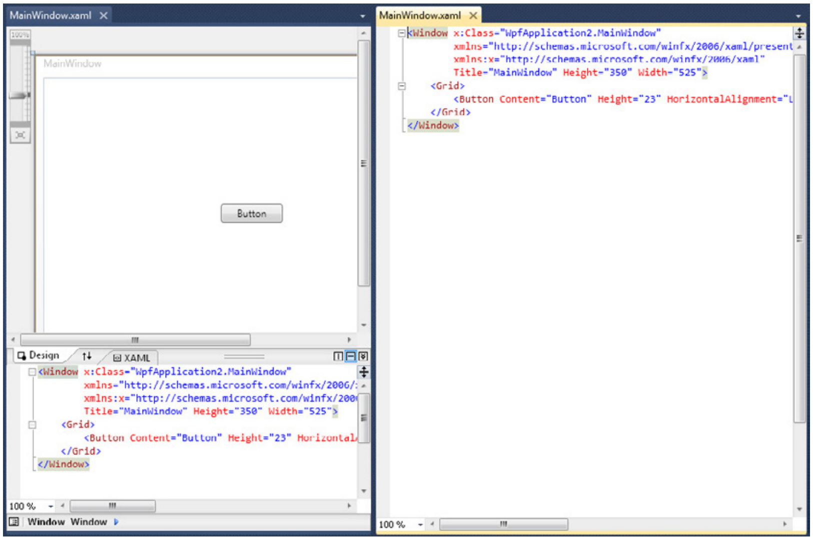
Now, in the Open With dialog box, select Source Code (Text) Editor and click OK:



Choose the program you want to use to open this file:

- Automatic Editor Selector (XML) (Default)
- Workflow Designer
- Windows Presentation Foundation Designer
- Windows Presentation Foundation Designer with Encoding
- XML (Text) Editor
- XML (Text) Editor with Encoding
- Source Code (Text) Editor
- Source Code (Text) Editor With Encoding
- HTML Editor
- HTML Editor with Encoding
- Notepad
- Binary Editor
- Resource Editor

You now have two XAML windows, one for your designer and one for the XAML:



```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/present
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Button" Height="23" HorizontalAlignment="L
    </Grid>
</Window>
```

```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/:
        xmlns:x="http://schemas.microsoft.com/winfx/200
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Button" Height="23" Horizontal
    </Grid>
</Window>
```

Now you can put them on two different monitors or whatever you feel like doing.

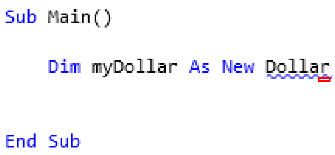
AX.93 Using Generate from Usage

WINDOWS	Alt,E, I, G, T
MENU	Edit | IntelliSense | Generate | New Type
COMMAND	Edit.GenerateNewType; EditorContextMenus.CodeWindow.Generate.GenerateNewType
VERSIONS	2010
CODE	vstipEdit0011

I'm a big fan of Test Driven Development (TDD), so I absolutely love this tip because it is a big step toward enabling TDD activities in Visual Studio. The idea behind it is simple; it allows you to use classes and members before you define them. This was created so that you can write your tests and use classes/members that haven't been created yet per the tenets of TDD ("red, green, refactor"). The C# and VB implementations are slightly different, so let's take a look at those differences.

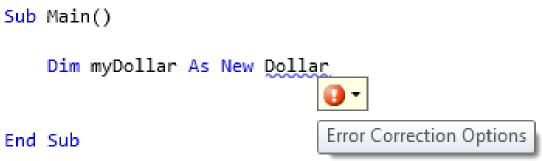
VB

Start by using a class that you haven't created yet:



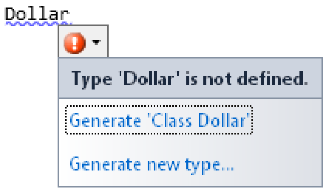
```
Sub Main()

    Dim myDollar As New Dollar

End Sub
```

Obviously, you will get an error:



```
Sub Main()

    Dim myDollar As New Dollar

End Sub
```

But wait. When you click the Error Correction Options, you get something new:



```
Dollar
```

Type 'Dollar' is not defined.

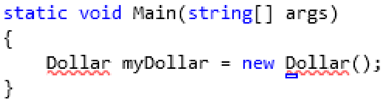
Click Generate 'Class Dollar', and you get a new file called Dollar.vb with a class stub inside:

```
Class Dollar

End Class
```

The error is gone, and you are ready to start using the class. You can repeat this process for new members that you create for the new class, as you use them.

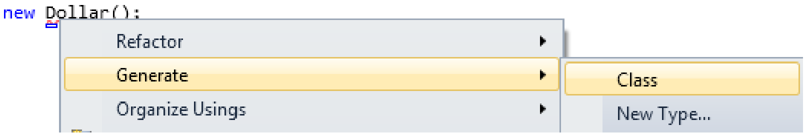
C#

Start by using a class that doesn't exist:

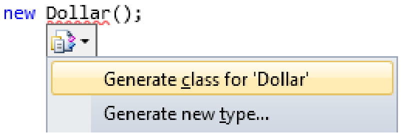


```
static void Main(string[] args)
{
    Dollar myDollar = new Dollar();
}
```

Here you have a couple of options. You can right-click and choose Generate | Class, or you can click the smart tag and choose Generate class for 'Dollar'. They both do the same thing, and you can see both options in the following illustrations:

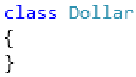


```
new Dollar();
```



```
new Dollar();
```

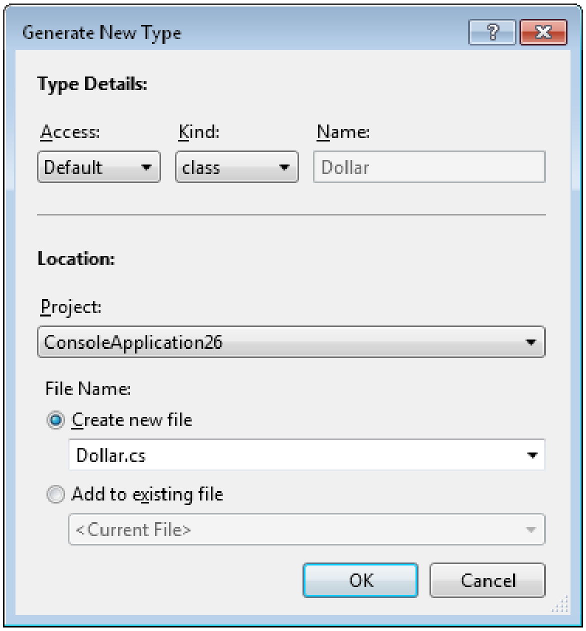
You get a new file called Dollar.cs with a class stub inside:



```
class Dollar
{
}
```

The error is gone, and you are ready to start using the class. You can repeat this process for new members that you create for the new class, as you use them.

When you get the hang of this and actually start using this feature for TDD activities, you would not use the examples I provided but would instead choose the Generate New Type option. The dialog box is the same for VB and C#.



Type Details:

Access: Default
Kind: class
Name: Dollar

Location:

Project: ConsoleApplication26

File Name:
● Create new file
Dollar.cs
○ Add to existing file
<Current File>

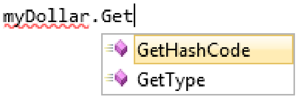
Notice that you have the ability to set Access, Kind, and—most importantly—Location. It's the Location option that TDD folks will use to put the classes into the proper project outside of their test projects.

AX.94 IntelliSense Suggestion Mode

DEFAULT	Ctrl+Alt+Spacebar
WINDOWS	Alt,E, I, T
MENU	Edit | IntelliSense | Toggle Completion Mode
COMMAND	Edit.ToggleCompletionMode
VERSIONS	2008, 2010
CODE	vstipEdit0012

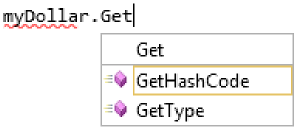
IntelliSense comes in two modes: Completion and Suggestion. You are already familiar with IntelliSense completion mode; it's the traditional mode that we have all used for years. But if you are into Test Driven Development (TDD), completion mode can be very annoying at times.

TDD developers often use classes and members before they exist. It's not fun when you go to type the name of something that doesn't exist and you get IntelliSense. Especially because you sometimes accidentally get an option you didn't want:



```
myDollar.Get
```

The solution is suggestion mode. Just press Ctrl+Alt+Spacebar to go into this mode:



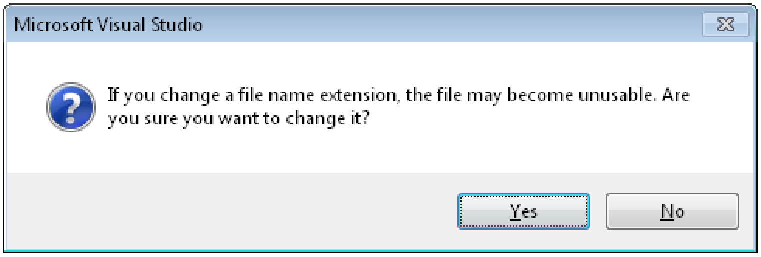
```
myDollar.Get
```

Now you get the best of both worlds: You can type in a name that doesn't exist and have quick access to the completion mode options as well. The risk of getting an option you don't want is reduced.

AX.95 Turn Off Automatic Symbol Renaming When You Rename a File in Solution Explorer

WINDOWS	Alt,T, O
MENU	Tools | Options | Projects And Solutions | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0011

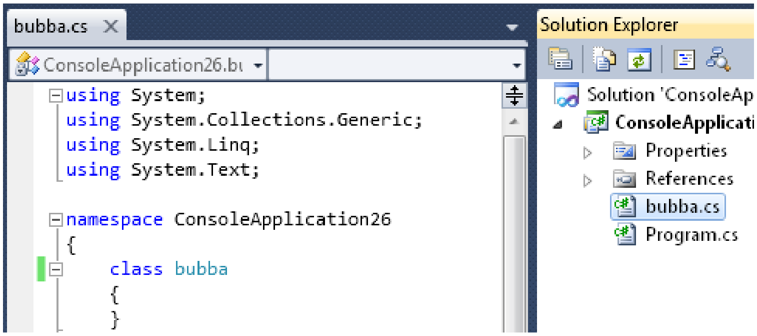
When you start to rename a class file in Solution Explorer, you receive the following prompt:



Microsoft Visual Studio

If you change a file name extension, the file may become unusable. Are you sure you want to change it?

When you click Yes, both the class name in all your code and the file are renamed:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication26
{
    class bubba
    {
    }
```

If you find yourself doing this a lot, you can get rid of the prompt to rename by going to Tools | Options | Projects And Solutions | General and clearing the Prompt For Symbolic Renaming When Renaming Files check box.

From now on, when you do your renaming, the prompt does not appear and everything is renamed automatically.

AX.96 Mark Methods and Types as Hidden from IntelliSense and the Object Browser

VERSIONS	2005, 2008, 2010
CODE	vstipTool0088

When you make assemblies for others to use, you might sometimes want to have hidden methods, properties, and other elements. To hide an item, you use the System.ComponentModel namespace.

C#

```
using System.ComponentModel;
```

VB

```
Imports System.ComponentModel
```

Hiding

To hide a property or method, you use the EditorBrowsable(EditorBrowsableState.Never) attribute.

C#

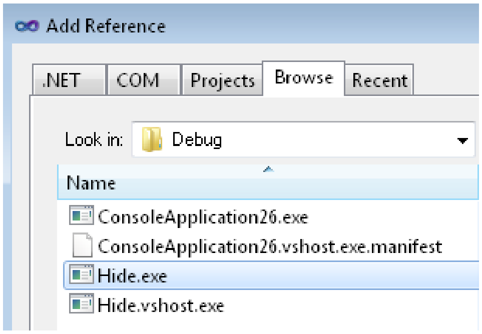
```
public class Person
{
    public int Age { get; set; }

    [EditorBrowsable(EditorBrowsableState.Never)]
    public string Name { get; set; }

    [EditorBrowsable(EditorBrowsableState.Never)]
    public void DoStuff()
    {
        //todo
    }
}
```

VB

```
Public Class Person

    Public Property Age As Integer

    <EditorBrowsable(EditorBrowsableState.Never)>
    Public Property Name As String

    <EditorBrowsable(EditorBrowsableState.Never)>
    Public Sub DoStuff()
        'todo stuff
    End Sub

End Class
```

Use

Testing this is a little tricky. You have to use the assembly externally because hiding is meant to hinder external users, *not* the creator of the assembly. Just making a reference to another project isn't enough. You actually have to go to the Browse tab and set a reference to the external assembly that is created:



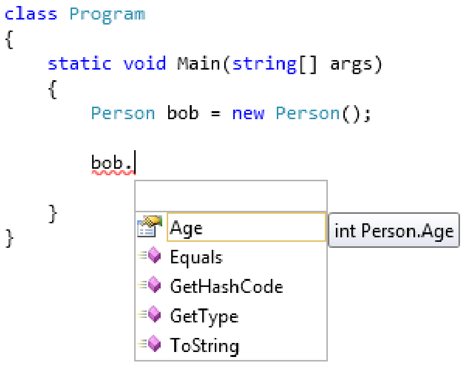
Look in: Debug

ConsoleApplication26.exe
ConsoleApplication26.vshost.exe.manifest
Hide.exe
Hide.vshost.exe

Result

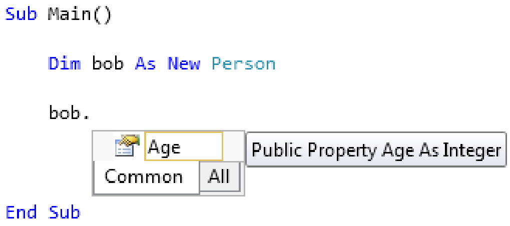
When you test it, the hidden items should not be visible.

C#



```
class Program
{
    static void Main(string[] args)
    {
        Person bob = new Person();

        bob.

    }
}
```

VB



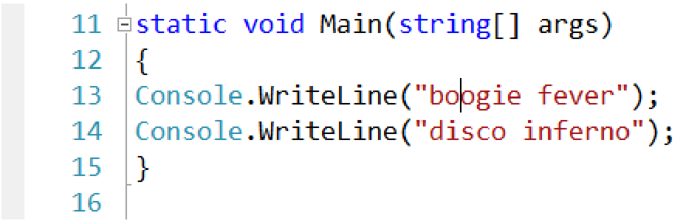
```
Sub Main()

    Dim bob As New Person

    bob.

End Sub
```

Additional Tips from Chapter 7

AX.97 Set or Remove a Breakpoint

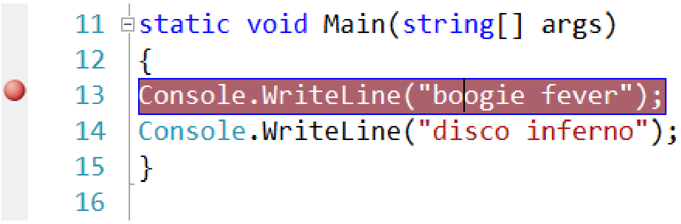
DEFAULT	F9
VISUAL BASIC 6	F9
VISUAL C# 2005	F9
VISUAL C++ 2	F9; Ctrl+Shift+F9
VISUAL C++ 6	F9
VISUAL STUDIO 6	F9
WINDOWS	Alt,D, G
MENU	Debug | Toggle Breakpoint
COMMAND	Debug.ToggleBreakpoint
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0016

Setting a simple breakpoint is easy. Just find some code:



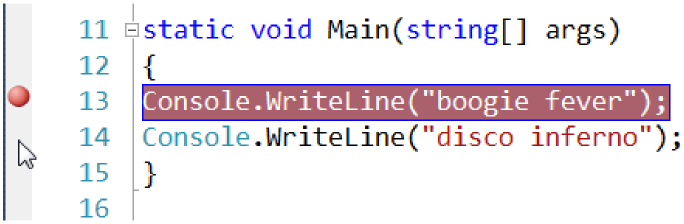
```
static void Main(string[] args)
{
Console.WriteLine("boogie fever");
Console.WriteLine("disco inferno");
}

```

Now set (or remove) a breakpoint either by pressing F9 or by selecting Debug | Toggle Breakpoint on your menu bar:



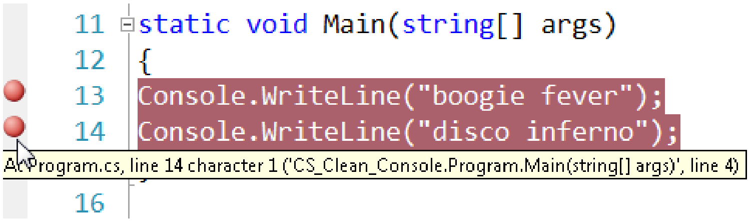
```
static void Main(string[] args)
{
Console.WriteLine("boogie fever");
Console.WriteLine("disco inferno");
}

```

If you prefer using the mouse, you can set a breakpoint by positioning your pointer in the margin next to the line on which you want to set the breakpoint:



```
static void Main(string[] args)
{
Console.WriteLine("boogie fever");
Console.WriteLine("disco inferno");
}
```

Then click your left mouse button:



```
static void Main(string[] args)
{
Console.WriteLine("boogie fever");
Console.WriteLine("disco inferno");
```

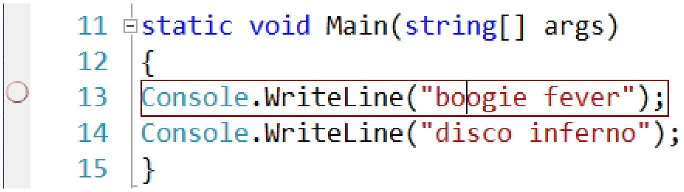
At Program.cs, line 14 character 1 ('CS_Clean_Console.Program.Main(string[] args)', line 4)

All the breakpoints you have set show up in the Breakpoints window as a list that you can work with later on.

AX.98 Enable or Disable a Breakpoint

DEFAULT	Ctrl+F9
VISUAL BASIC 6	[no shortcut]
COMMAND	Debug.EnableBreakpoint
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0017

Sometimes you want to keep a breakpoint around but need to temporarily disable it. This is easy to do and requires only a slight muscle memory modification from setting and removing a breakpoint. Just put your cursor on the line with the breakpoint you want to disable, then press Ctrl+F9 to disable the breakpoint (notice that the glyph in the margin changes):



```
static void Main(string[] args)
{
Console.WriteLine("boogie fever");
Console.WriteLine("disco inferno");
}
```

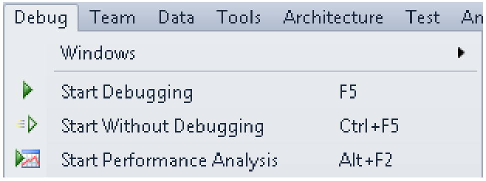
To re-enable it, press Ctrl+F9 again.

All your disabled breakpoints are easily visible in the Breakpoints window alongside your enabled breakpoints so that you can work with them later.

AX.99 Start Debugging vs. Start Without Debugging

DEFAULT	F5 (start); Ctrl+F5 (start w/o debug)
VISUAL BASIC 6	F5 (start); Ctrl+F5 (start w/o debug); Ctrl+Alt+Break (stop)
VISUAL C# 2005	F5 (start); Ctrl+F5 (start w/o debug); Shift+F5 (stop)
VISUAL C++ 2	F5 (start); Ctrl+F5 (start w/o debug); Alt+F5 (stop)
VISUAL C++ 6	F5 (start); Ctrl+F5 (start w/o debug); Shift+F5 (stop)
VISUAL STUDIO 6	F5 (start); Ctrl+F5 (start w/o debug); Shift+F5 (stop)
WINDOWS	Alt,D, S (start); Alt,D, H (start w/o debug)
MENU	Debug | Start Debugging; Debug | Start Without Debugging
COMMAND	Debug.Start; Debug.StartWithoutDebugging
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0037

There seems to be a great deal of confusion as to what actually happens when you use Start Debugging (F5) versus Start Without Debugging (Ctrl+F5):


Debug
Team
Data
Tools
Architecture
Test
An
Windows
Start Debugging F5
Start Without Debugging Ctrl+F5
Start Performance Analysis Alt+F2

Starting with Debugging

Let's start with the basics: When you press F5 (Start Debugging) in Visual Studio, it launches your application, attaches the debugger, and lets you do all the "normal" things you would expect.

According to the documentation (see "Debugger Roadmap" at *http://msdn.microsoft.com/en-us/library/k0k771bt(v=VS.100).aspx*), here is what the debugger does:

> *"The Visual Studio debugger is a powerful tool that allows you to observe the run-time behavior of your program and locate logic errors. The debugger works with all Visual Studio programming languages and their associated libraries. With the debugger, you can break, or suspend, execution of your program to examine your code, evaluate and edit variables in your program, view registers, see the instructions created from your source code, and view the memory space used by your application."*

The debugger: Release builds

One popular misconception is that the debugger doesn't come into play for release builds. This isn't true. Set a breakpoint in some code for a release build, and press F5 to see whether it stops there. Of course it does!

The debugger is attached. Now, what about things that aren't happening?

Note Release builds are optimized versions of your code, so make sure to test your breakpoint on a piece of code that is used.

For example, you can't use the System.Diagnostics.Debug class methods to send messages to the Output window because the compiler strips them out for release builds:

```
System.Diagnostics.Debug.WriteLine
    ("This is a debug message.");
```

Starting Without Debugging

This is exactly what it sounds like. When you choose the Start Without Debugging option, the application starts without the debugger attached. That's it! Nothing else happens. It just doesn't attach the debugger. Otherwise, everything else is the same. So, the practical implications of this are obvious: Without the debugger attached, when the application runs, it does not hit breakpoints, emit debug messages, and so on.

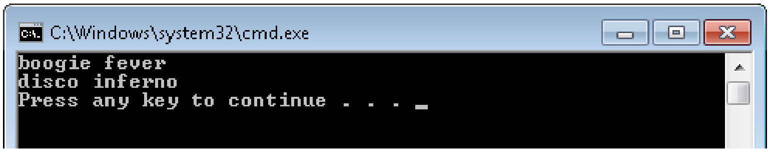
So now let's deal with the biggest myth about choosing the Start Without Debugging option.

Myth: Start Without Debugging creates a release build

Not true. It uses the build you are currently on. So if you are using a debug build and you press Ctrl+F5, Visual Studio runs that debug build. The easiest way to test this is to use conditional compilation to see whether you are using a debug build:

```
#if DEBUG
    Console.WriteLine("boogie fever");
    Console.WriteLine("disco inferno");

    System.Diagnostics.Debug.WriteLine
        ("This is a debug message.");
#endif
```

In this example, the Console statements run, but the System.Diagnostics.Debug statement does not run because the debugger is not attached. So you get the following output:



```
boogie fever
disco inferno
Press any key to continue . . . _
```

Finally

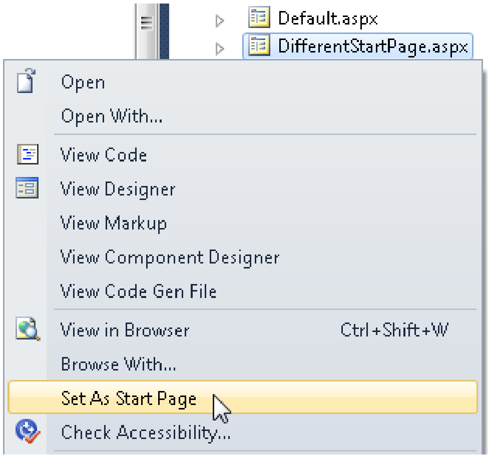
OK, so the obvious question is "Why have this option?" Well, the answer is simple: So that you can run the application without having the overhead of the debugger attached, to do a quick "Smoke Test" to see whether the code runs. As you learn more about this feature, you might find other reasons for wanting to run without the debugger attached.

Now you have a better idea of the difference between the Start Debugging and Start Without Debugging options.

AX.100 Set As Start Page

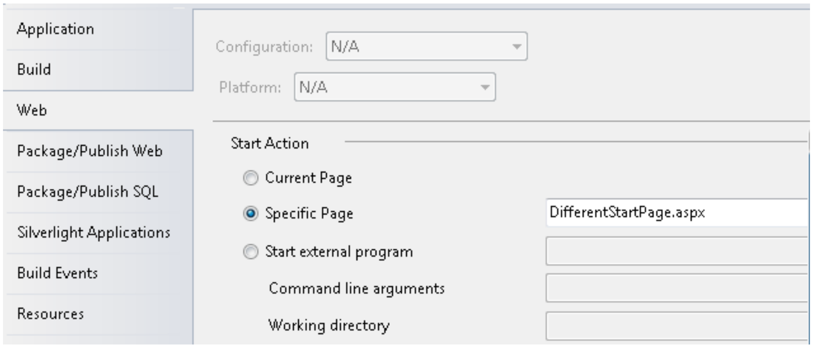
WINDOWS	ALT, P, P, P, Enter
MENU	Project | Set As Start Page
COMMAND	Project.SetAsStartPage
VERSIONS	2005, 2008, 2010
CODE	vstipProj0027

When you create a new web project, the default start page is set for you automatically and is the first page you see when you start debugging. However, you might find that you want a different page to start with instead. It's easy to change the start page. Simply right-click the new start page in Solution Explorer, and select Set As Start Page:



The changes take effect immediately, and the new page becomes the start page until you change it.

In case you are curious, this actually changes the Specific Page value in your Web Application properties dialog box (Web Tab, Start Action):



Configuration: N/A

Platform: N/A

Start Action

○ Current Page

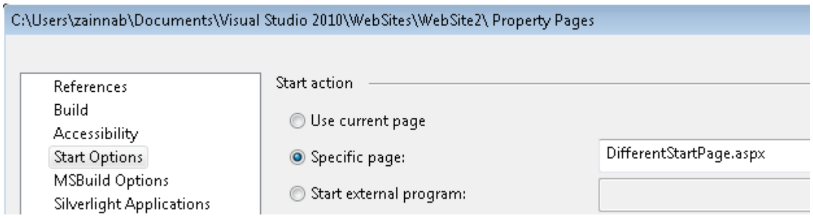
● Specific Page DifferentStartPage.aspx

○ Start external program

Command line arguments

Working directory

For websites, you can find this in the project properties dialog box under Start Options:


C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite2\ Property Pages

Start action

○ Use current page

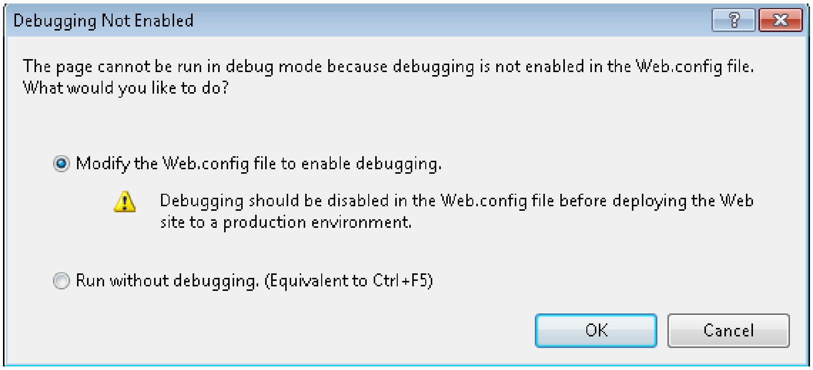
● Specific page: DifferentStartPage.aspx

○ Start external program:

AX.101 Enable Debugging in Web.Config

VERSIONS	2008, 2010
CODE	vstipProj0026

If you have ever created a web project in Visual Studio, you have undoubtedly encountered the following dialog box:



The page cannot be run in debug mode because debugging is not enabled in the Web.config file. What would you like to do?

● Modify the Web.config file to enable debugging.

Debugging should be disabled in the Web.config file before deploying the Web site to a production environment.

○ Run without debugging. (Equivalent to Ctrl+F5)

A quick read tells you that it defaults to the Modify The Web.config File To Enable Debugging option:

```
<system.web>
  <compilation debug="true"
               targetFramework="4.0"/>
```

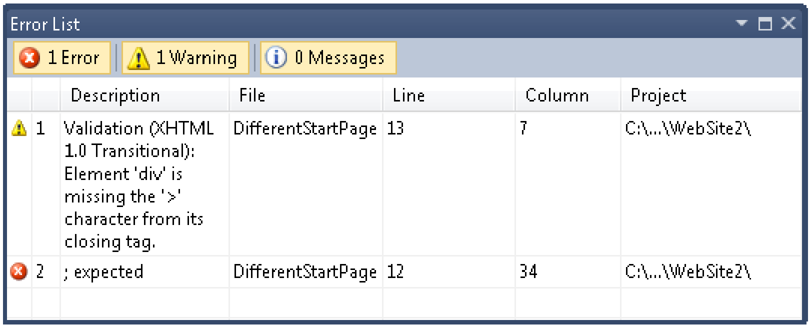
A couple of things need to be pointed out here:

- You can avoid this dialog box by editing your Web.config file manually and setting **debug** to true.
- No dialog box comes up to enable you to turn this off again before you deploy, so be aware that you should ensure that it is turned off before you go to production.

AX.102 View the Error List Window

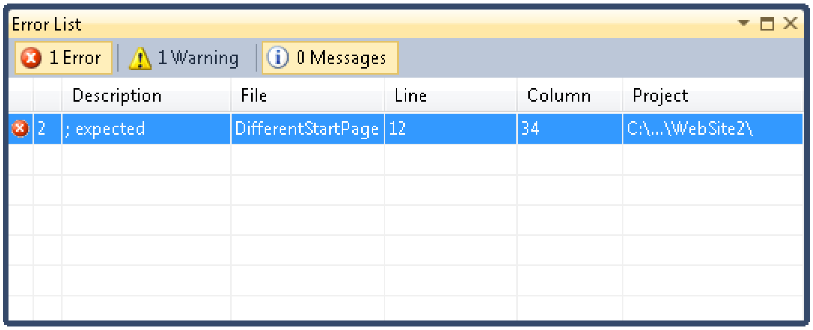
DEFAULT	Ctrl+\, E; Ctrl+\, Ctrl+E
VISUAL BASIC 6	Ctrl+\, E; Ctrl+\, Ctrl+E; Ctrl+W, Ctrl+E
VISUAL C# 2005	Ctrl+\, E; Ctrl+\, Ctrl+E; Ctrl+W, Ctrl+E; Ctrl+W, E
VISUAL C++ 2	Ctrl+\, E; Ctrl+\, Ctrl+E
VISUAL C++ 6	Ctrl+\, E; Ctrl+\, Ctrl+E
VISUAL STUDIO 6	Ctrl+\, E; Ctrl+\, Ctrl+E
WINDOWS	Alt,V, I
MENU	View | Error List
COMMAND	View.ErrorList
VERSIONS	2005, 2008, 2010
CODE	vstipTool0018

You can view the Error List window by pressing Ctrl+\, E inside Visual Studio:



		Description	File	Line	Column	Project
⚠	1	Validation (XHTML 1.0 Transitional): Element 'div' is missing the '>' character from its closing tag.	DifferentStartPage	13	7	C:\...\WebSite2\
✖	2	; expected	DifferentStartPage	12	34	C:\...\WebSite2\

Also, you can filter by type, using the toggle buttons at the top (Errors, Warnings, Messages) to show or hide particular messages:

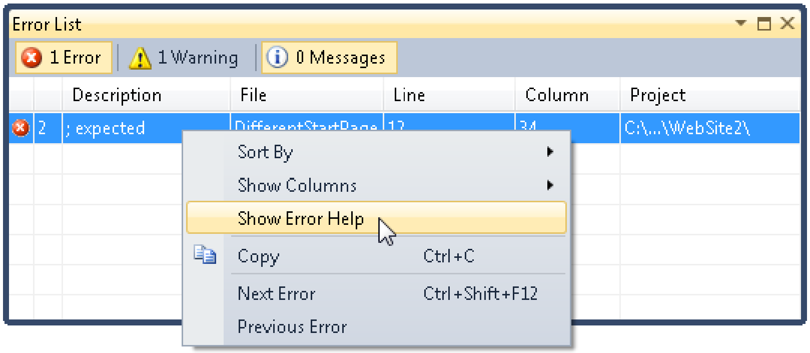


		Description	File	Line	Column	Project
✖	2	; expected	DifferentStartPage	12	34	C:\...\WebSite2\

AX.103 Show Error Help from Errors List Window

DEFAULT	F1
VISUAL BASIC 6	F1
VISUAL C# 2005	F1
VISUAL C++ 2	F1
VISUAL C++ 6	F1
VISUAL STUDIO 6	F1
WINDOWS	F1
MENU	[Context Menu] | Show Error Help
COMMAND	Help.F1Help
VERSIONS	2005, 2008, 2010
CODE	vstipTool0020

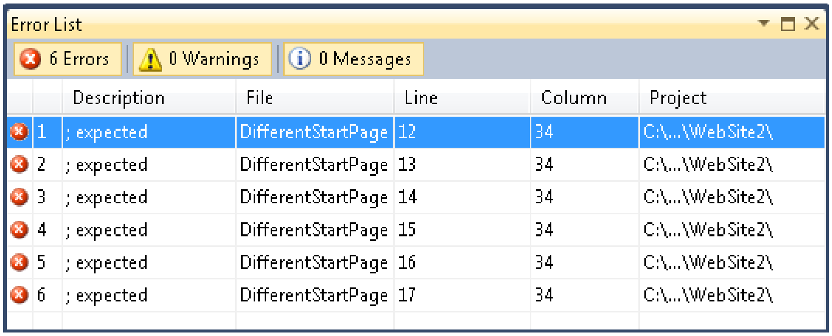
You can get help on any error in the Errors List window by right-clicking any error and choosing Show Error Help. This opens the documentation, if available, with details about the specific error:



AX.104 Hide or Show Error List When the Build Finishes with Errors

WINDOWS	Alt,T, O
MENU	Tools | Options | Projects and Solutions | Default
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0022

By default, the Error List window appears when build errors have occurred:



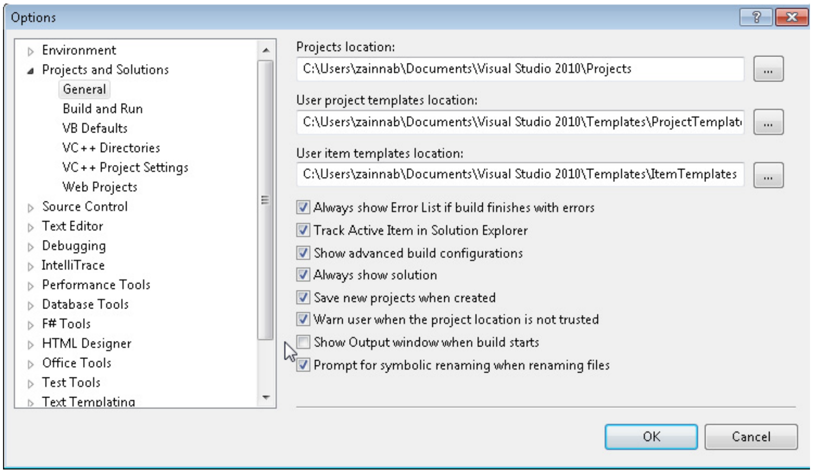
		Description	File	Line	Column	Project
✖	1	; expected	DifferentStartPage	12	34	C:\...\WebSite2\
✖	2	; expected	DifferentStartPage	13	34	C:\...\WebSite2\
✖	3	; expected	DifferentStartPage	14	34	C:\...\WebSite2\
✖	4	; expected	DifferentStartPage	15	34	C:\...\WebSite2\
✖	5	; expected	DifferentStartPage	16	34	C:\...\WebSite2\
✖	6	; expected	DifferentStartPage	17	34	C:\...\WebSite2\

You can change this behavior by going to Tools | Options | Projects And Solutions | General and clearing the Always Show Error List If Build Finishes With Errors check box.

AX.105 Show the Output Window During Build

WINDOWS	Alt,T, O
MENU	Tools | Options | Projects and Solutions | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0045

When you do a build, by default, it always shows the Output window. If you don't want the Output window to show up every time you do a build, you can go to Tools | Options | Projects And Solutions | General and clear the Show Output Window When Build Starts check box:



Projects location:
C:\Users\zainnab\Documents\Visual Studio 2010\Projects

User project templates location:
C:\Users\zainnab\Documents\Visual Studio 2010\Templates\ProjectTemplat

User item templates location:
C:\Users\zainnab\Documents\Visual Studio 2010\Templates\ItemTemplates

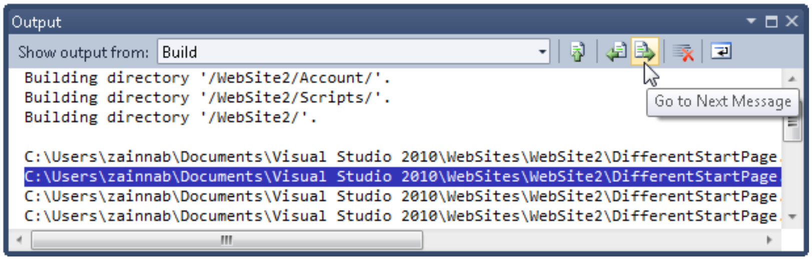
☑ Always show Error List if build finishes with errors
☑ Track Active Item in Solution Explorer
☑ Show advanced build configurations
☑ Always show solution
☑ Save new projects when created
☑ Warn user when the project location is not trusted
☐ Show Output window when build starts
☑ Prompt for symbolic renaming when renaming files

Of course, you can bring up the Output window anytime by pressing Ctrl+Alt+O.

AX.106 Navigate Among Errors in the Output Window

DEFAULT	F8 (next); Shift+F8 (previous)
VISUAL BASIC 6	[no shortcut]
VISUAL C# 2005	F8 (next); Shift+F8 (previous)
VISUAL C++ 2	F4 (next); Shift+F4 (previous)
VISUAL C++ 6	F8 (next); F4 (next)
VISUAL STUDIO 6	F8 (next); F12 (next); Shift+F8 (previous); Shift+F12 (previous)
COMMAND	Edit.GoToNextLocation; Edit.GoToPrevLocation;
VERSIONS	2005, 2008, 2010
CODE	vstipTool0043

Did you know that you can use toolbar buttons on the Output window to navigate between errors?

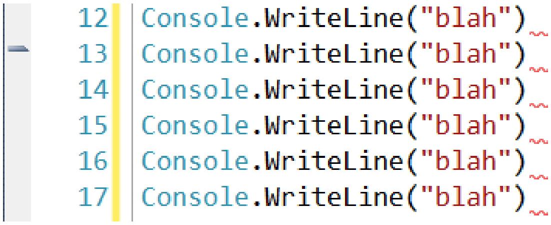


Show output from: Build

```
Building directory '/WebSite2/Account/'.
Building directory '/WebSite2/Scripts/'.
Building directory '/WebSite2/'.

C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite2\DifferentStartPage
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite2\DifferentStartPage
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite2\DifferentStartPage
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite2\DifferentStartPage
```

Also, assuming the Output window is active, you can use F8 (next) and Shift+F8 (previous) to navigate among the errors as well.

The cursor in the editor automatically follows you as you go through the errors, places the cursor where you can make changes so that you just start typing, and has an indicator in the far left margin to show your current position:

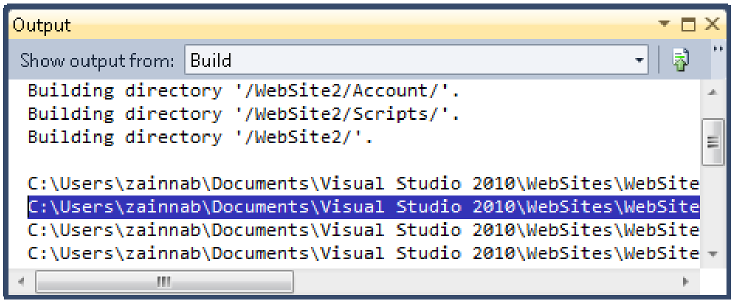


```
Console.WriteLine("blah")
Console.WriteLine("blah")
Console.WriteLine("blah")
Console.WriteLine("blah")
Console.WriteLine("blah")
Console.WriteLine("blah")
```

AX.107 Customize the Output Window

WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | Fonts and Colors | Show settings for | Output Window
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0044

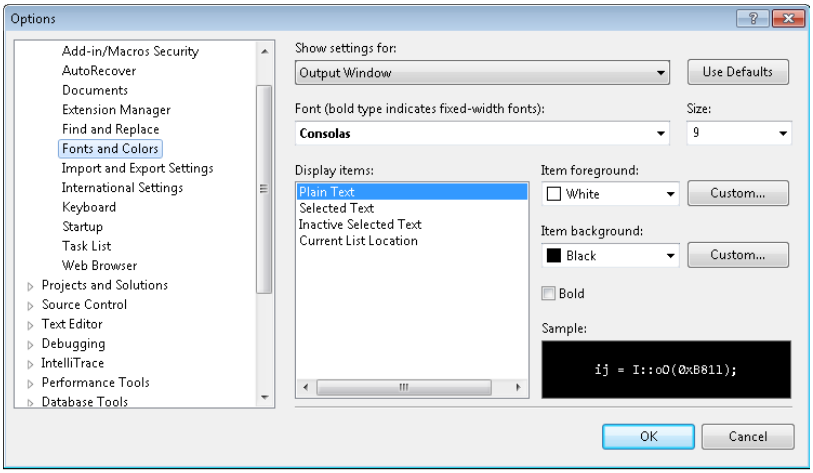
Working with the Output window is fairly common. Personalizing the window to your needs can definitely improve your work day. The Output window can be modified from its original configuration:



Show output from: Build

```
Building directory '/WebSite2/Account/'.
Building directory '/WebSite2/Scripts/'.
Building directory '/WebSite2/'.

C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
```

Just go to Tools | Options | Fonts And Colors | Show Settings For | Output Window. You can change the font type, color, and size for a variety of display items, as shown in the following illustration:



Show settings for: Output Window

Font (bold type indicates fixed-width fonts): **Consolas**

Size: 9

Display items:
- Plain Text
- Selected Text
- Inactive Selected Text
- Current List Location

Item foreground: White

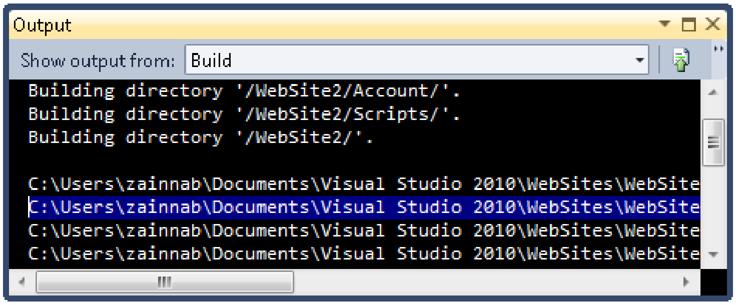
Item background: Black

☐ Bold

Sample:

```
ij = I::oO(0xB811);
```

Here's the result in the Output window:



```
Building directory '/WebSite2/Account/'.
Building directory '/WebSite2/Scripts/'.
Building directory '/WebSite2/'.

C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
C:\Users\zainnab\Documents\Visual Studio 2010\WebSites\WebSite
```

Obviously, you'll want to experiment with combinations that suit you.

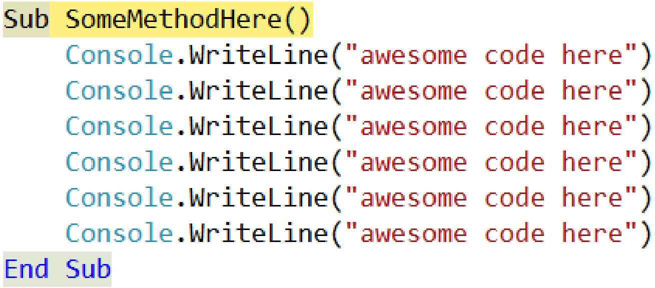
AX.108 Step Out of or Over a Method

DEFAULT	Shift+F11 (step out); F10 (step over)
VISUAL BASIC 6	Shift+F11 (step out); Ctrl+Shift+F8 (step out); F10 (step over); Shift+F8 (step over)
VISUAL C# 2005	Shift+F11 (step out); F10 (step over)
VISUAL C++ 2	Shift+F7 (step out); F10 (step over)
VISUAL C++ 6	Shift+F11 (step out); F10 (step over)
VISUAL STUDIO 6	Shift+F11 (step out); F10 (step over)
WINDOWS	Alt,D, T (step out); Alt, D, O (step over)
MENU	Debug | Step Out; Debug | Step Over
COMMAND	Debug.StepOut ; Debug.StepOver
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0035

When debugging your code, you often come across a call to another method:

```
Sub Main()

    Console.WriteLine("blah")
    Console.WriteLine("blah")

    SomeMethodHere()

    Console.WriteLine("blah")
    Console.WriteLine("blah")
    Console.WriteLine("blah")
End Sub
```

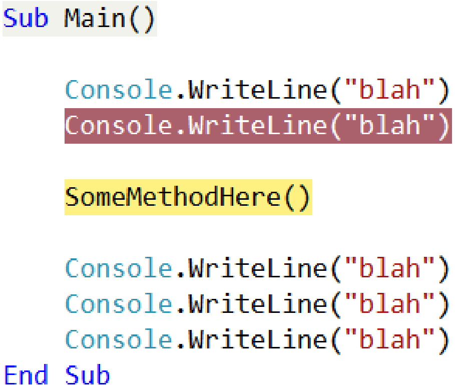
At this point, if you want to see what is in the method, you can step into it by pressing F11:



```
Sub SomeMethodHere()
    Console.WriteLine("awesome code here")
    Console.WriteLine("awesome code here")
    Console.WriteLine("awesome code here")
    Console.WriteLine("awesome code here")
    Console.WriteLine("awesome code here")
    Console.WriteLine("awesome code here")
End Sub
```

Step Out

Once inside, you might decide you don't really need to go through all the code. You can, at any time, step out by pressing Shift+F11, which finishes execution of the current method and returns you to the original call:



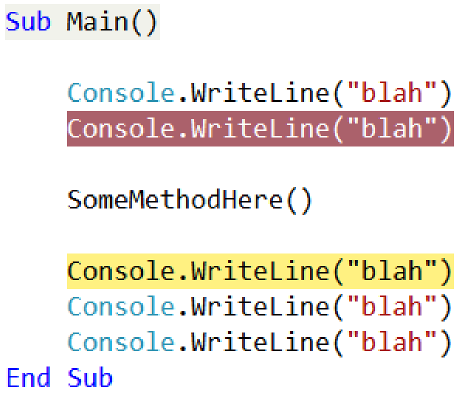
```
Sub Main()

    Console.WriteLine("blah")
    Console.WriteLine("blah")

    SomeMethodHere()

    Console.WriteLine("blah")
    Console.WriteLine("blah")
    Console.WriteLine("blah")
End Sub
```

From here, you can continue on as you normally would with your debugging.

Step Over

You have another option when you get to a method call:

You can decide that the method should just run without having to look at what is inside it. If you want to run the method and move to the next line of code, just press F10 to step over that method:



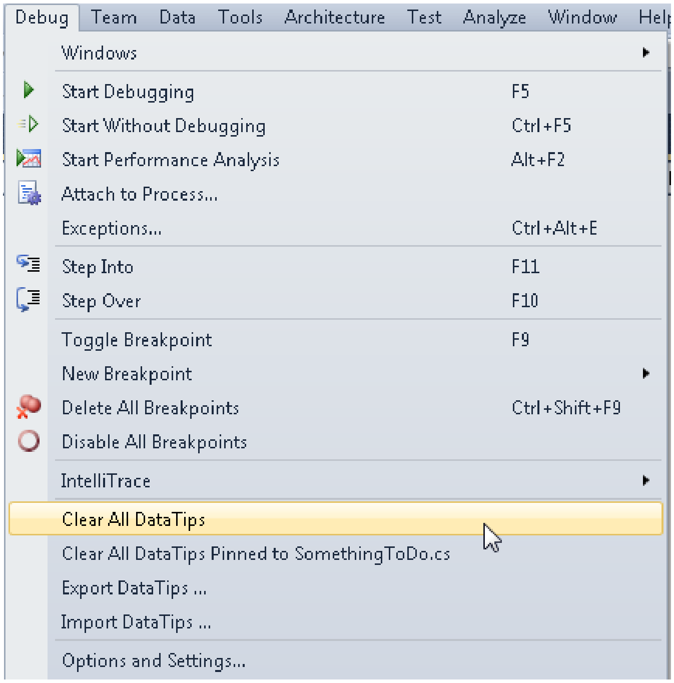
```
Sub Main()

    Console.WriteLine("blah")
    Console.WriteLine("blah")

    SomeMethodHere()

    Console.WriteLine("blah")
    Console.WriteLine("blah")
    Console.WriteLine("blah")
End Sub
```

You can now navigate more effectively by using these techniques to determine how deep you want to get into method calls.

AX.109 Clearing Your DataTips

WINDOWS	Alt,D, A, A, ENTER (clear all)
MENU	Debug | Clear All DataTips; Debug | Clear All DataTips Pinned to [file]
COMMAND	Debug.ClearAllDataTips
VERSIONS	2010
CODE	vstipDebug0014

As you create your DataTips, you might find that it is necessary to clear them out when you are done. You can clear your DataTips in a couple of ways from the Debug menu:



Clear All DataTips

This command does what it says and clears *all* (pinned and floating) DataTips in the solution.

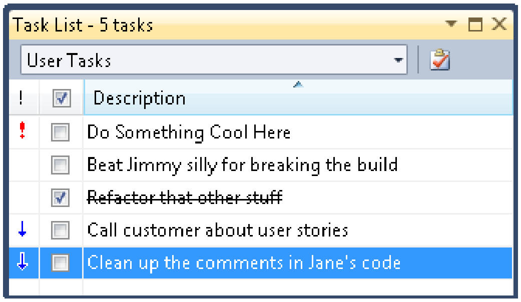
Clear All DataTips Pinned to [File Name]

Clear only the pinned DataTips in [file name]. This option appears only if you have at least one pinned DataTip in a file. The referenced file is the one you have currently open in the editor.

AX.110 Create User Tasks in the Task List

DEFAULT	Ctrl+\, Ctrl+T; Ctrl+\, T
VISUAL BASIC 6	Ctrl+\, Ctrl+T; Ctrl+\, T; Ctrl+ALT+K
VISUAL C# 2005	Ctrl+\, Ctrl+T; Ctrl+\, T; Ctrl+W,Ctrl+T; Ctrl+W, T
VISUAL C++ 2	Ctrl+\, Ctrl+T; Ctrl+\, T
VISUAL C++ 6	Ctrl+\, Ctrl+T; Ctrl+\, T
VISUAL STUDIO 6	Ctrl+\, Ctrl+T; Ctrl+\, T; Ctrl+Alt+K
WINDOWS	Alt, V, K
MENU	View | Task List
COMMAND	View.TaskList
VERSIONS	2005, 2008, 2010
CODE	vstipTool0027

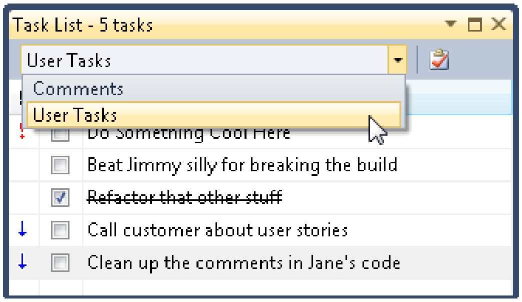
Creating tasks is useful when you need reminders for getting work done. User Tasks is a checklist of "to do" items that are stored in the Solution User Options (.suo) file. They are per-user settings and are not typically checked into source control:



Task List - 5 tasks
User Tasks

! ☑ Description
! ☐ Do Something Cool Here
☐ Beat Jimmy silly for breaking the build
☑ ~~Refactor that other stuff~~
↓ ☐ Call customer about user stories
↓ ☐ Clean up the comments in Jane's code

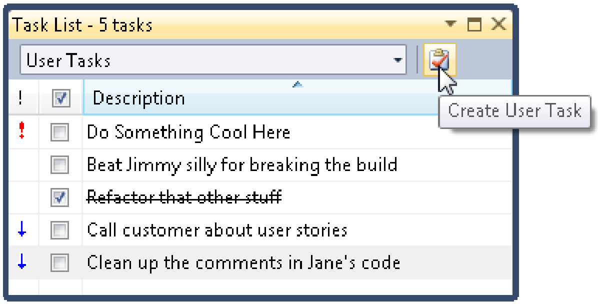
To begin, make sure that User Tasks is selected from the drop-down list in the Task List (Ctrl+\, T):



Task List - 5 tasks
User Tasks

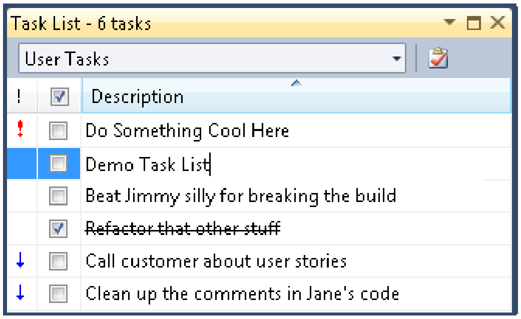
! ☐ Do Something Cool Here
☐ Beat Jimmy silly for breaking the build
☑ ~~Refactor that other stuff~~
↓ ☐ Call customer about user stories
↓ ☐ Clean up the comments in Jane's code

Next, click the Create User Task button:



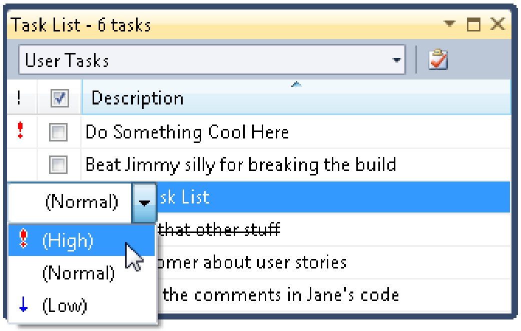
!	☑	Description
!	☐	Do Something Cool Here
	☐	Beat Jimmy silly for breaking the build
	☑	~~Refactor that other stuff~~
↓	☐	Call customer about user stories
↓	☐	Clean up the comments in Jane's code

Now you can enter any task you want:



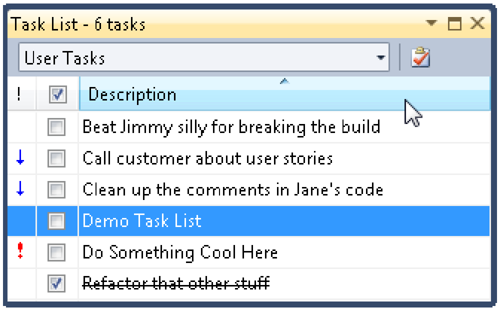
!	☑	Description
!	☐	Do Something Cool Here
	☐	Demo Task List
	☐	Beat Jimmy silly for breaking the build
	☑	~~Refactor that other stuff~~
↓	☐	Call customer about user stories
↓	☐	Clean up the comments in Jane's code

You can easily edit any task by double-clicking or pressing Enter while in it. Also, you can set priority levels on them by clicking in the box to the far left of any task:



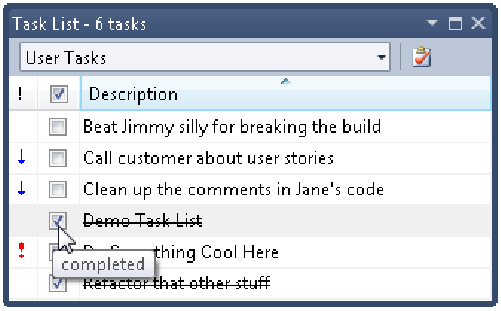
!	☑	Description
!	☐	Do Something Cool Here
	☐	Beat Jimmy silly for breaking the build
		sk List
		~~that other stuff~~
		omer about user stories
		the comments in Jane's code

Naturally, you can sort them by any column, such as Description or Priority:



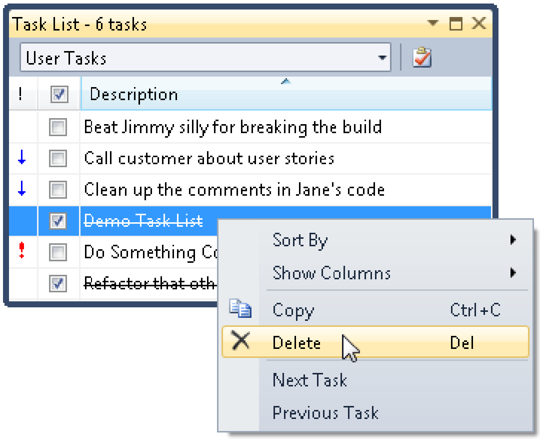
!	☑	Description
	☐	Beat Jimmy silly for breaking the build
↓	☐	Call customer about user stories
↓	☐	Clean up the comments in Jane's code
	☐	Demo Task List
!	☐	Do Something Cool Here
	☑	~~Refactor that other stuff~~

When you are done with a task, you can just click the check box to mark it complete:



!	☑	Description
	☐	Beat Jimmy silly for breaking the build
↓	☐	Call customer about user stories
↓	☐	Clean up the comments in Jane's code
	☑	~~Demo Task List~~
!	☐	Do Something Cool Here
	☑	~~Refactor that other stuff~~

Or you can right-click the task and delete it:



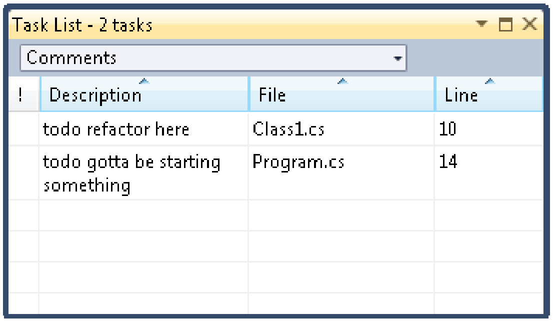
!	☑	Description
	☐	Beat Jimmy silly for breaking the build
↓	☐	Call customer about user stories
↓	☐	Clean up the comments in Jane's code
	☑	~~Demo Task List~~
!	☐	Do Something C
	☑	~~Refactor that oth~~

Note A command called CreateUserTask is available, but it doesn't accept any arguments and creates only a blank task. For this reason, I didn't list it in the summary information.

AX.111 Show the Full File Path in the Task List

WINDOWS	Alt,T, O
MENU	Tools | Options
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0033

Sometimes you have comments and/or shortcuts with descriptions and/or file names that are the same but in different solutions. Knowing the full path to the file can help you determine which item you are looking at in the Task List. This tip can help you with the items that are part of the Comments and Shortcuts areas. Normally, an item in the Task List doesn't have the full path:

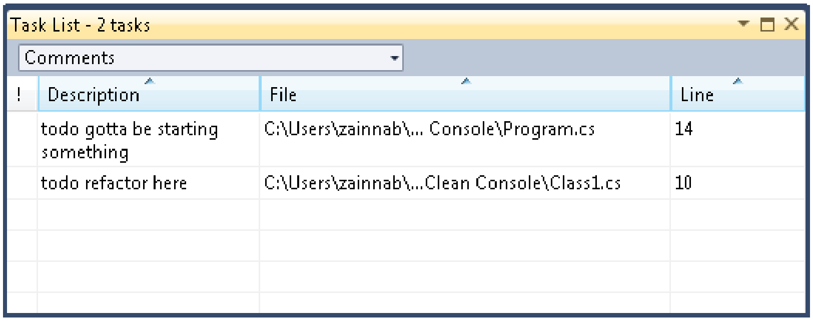


Comments

!	Description	File	Line
	todo refactor here	Class1.cs	10
	todo gotta be starting something	Program.cs	14

You can enable the full path by going to Tools | Options | Environment | Task List | Task List Options and clearing the Hide Full File Paths check box:

Now you can see the full file path in the Task List and clearly see where a comment or shortcut resides:



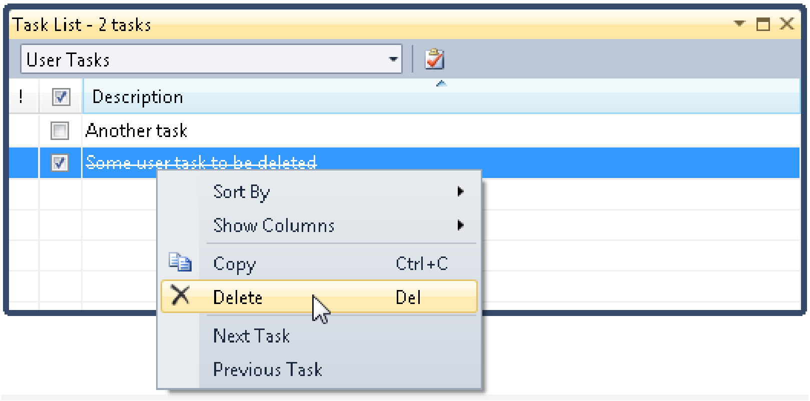
Comments

!	Description	File	Line
	todo gotta be starting something	C:\Users\zainnab\... Console\Program.cs	14
	todo refactor here	C:\Users\zainnab\...Clean Console\Class1.cs	10

AX.112 Disable the Prompt for Deleting Items from the Task List

WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | Task List | Task List
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0031

When dealing with the Task List, removing items is quite common. To delete an item from the Task List, simply right-click any item and choose Delete:

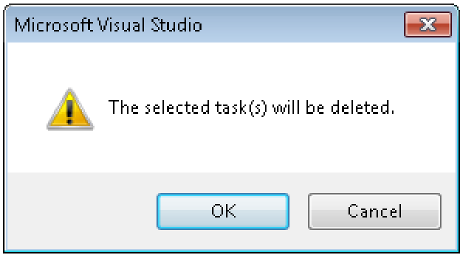

Task List - 2 tasks

User Tasks

☐ Another task

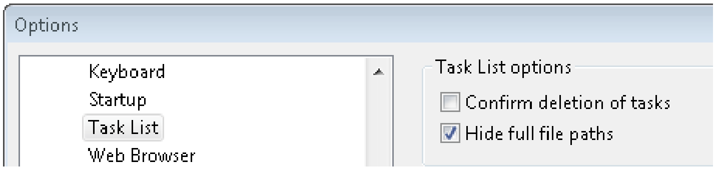
☑ ~~Some user task to be deleted~~

The following prompt appears:


Microsoft Visual Studio

The selected task(s) will be deleted.

You can turn the prompt dialog box off by going to Tools | Options | Environment | Task List | Task List Options and clearing the Confirm Deletion Of Tasks check box:


Options

Task List options

☐ Confirm deletion of tasks

☑ Hide full file paths

Now when you delete tasks, you have no dialog box to contend with. Of course, this means that you can also now accidentally delete a task without anything to stop you, so use this option at your own risk.

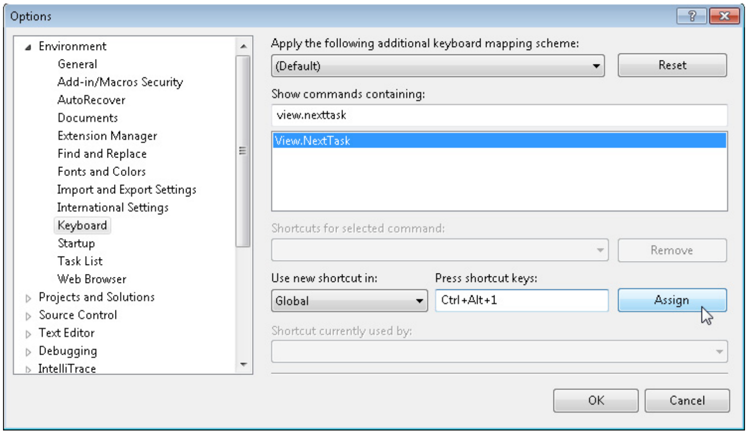
AX.113 Navigate Task List Entries with the Keyboard

DEFAULT	F8 (next location); Shift+F8 (prev location); [no shortcut] (next task); [no shortcut] (prev task)
VISUAL BASIC 6	[no shortcut] (next location); [no shortcut] (prev location); Ctrl+Shift+F12 (next task); [no shortcut](prev task)
VISUAL C# 2005	F8 (next location); Shift+F8 (prev location); [no shortcut] (next task); [no shortcut] (prev task)
VISUAL C++ 2	F4 (next location); Shift+F4 (prev location); [no shortcut] (next task); [no shortcut] (prev task)
VISUAL C++ 6	F8 (next location); F4 (next location); [no shortcut] (next task); [no shortcut] (prev task)
VISUAL STUDIO 6	F8 (next location); F12 (next location); Shift+F8 (prev location); Shift+F12 (prev location); [no shortcut] (next task); [no shortcut] (prev task)
COMMAND	Edit.GoToNextLocation; Edit.GoToPrevLocation; View.NextTask; View.PreviousTask
VERSIONS	2005, 2008, 2010
CODE	vstipTool0092

F8 and Shift+F8 are the universal “next” and “previous” keyboard shortcuts for items in tool window lists. For example, when the Task List is up, these keyboard commands move the focus between task items. If the Errors window is up, the commands move the focus between errors. However, you cannot use these commands to switch to the Task List from the Errors window.

What if you wanted a set of keyboard shortcuts that switched to the Task List and then navigated between them? Just go to Tools | Options | Environment | Keyboard, and assign shortcut keys to the View.NextTask and View.PreviousTask commands, as shown in the following illustration.

> **Note** For more information about assigning shortcut keys, see vstipTool0063 (“Keyboard Shortcuts: Creating New Shortcuts,” page 127).



Apply the following additional keyboard mapping scheme: (Default)

Show commands containing: view.nexttask

View.NextTask

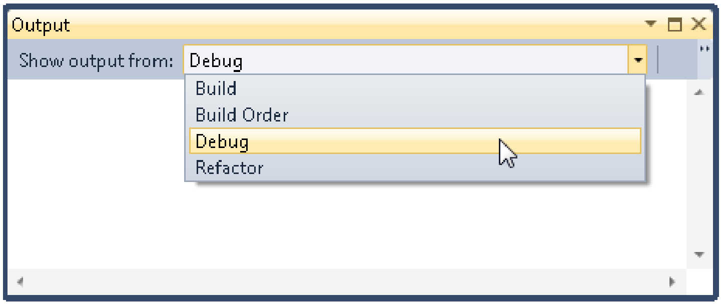
Use new shortcut in: Global

Press shortcut keys: Ctrl+Alt+1

AX.114 Navigating Between Output Window Panes with the Keyboard

COMMAND	Window.NextSubPane; Window.PreviousSubPane
VERSIONS	2005, 2008, 2010
CODE	vstipTool0091

The Output window can have several panes. How many depends on your context:



Show output from: Debug

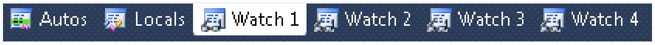
You can use Window.[Next / Previous]Subpane to move between these panes. However, the commands are not bound, by default, to any keyboard mappings. That is easily solved by going to Tools | Options | Environment | Keyboard and assigning shortcut keys to the commands.

Note For more information about assigning shortcut keys, see vstipTool0063 ("Keyboard Shortcuts: Creating New Shortcuts," page 127).

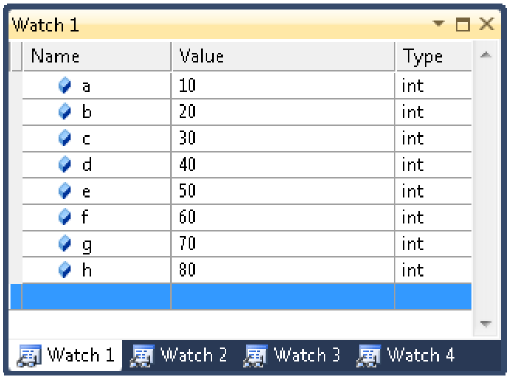
AX.115 The Watch Window: Moving Values Between Watch Windows

DEFAULT	Ctrl+Alt+W,1; Ctrl+Alt+W,[2-4]
VISUAL BASIC 6	Ctrl+Alt+W,1; Ctrl+Alt+W,[2-4]
VISUAL C# 2005	Ctrl+Alt+W,1; Ctrl+D, Ctrl+W; Ctrl+D, W; Ctrl+Alt+W,[2-4]
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+Alt+W,1; Ctrl+Alt+W,[2-4]
VISUAL STUDIO 6	Ctrl+Alt+W,1; Ctrl+Alt+W,[2-4]
WINDOWS	Alt,D, W, W, [1-4]
MENU	Debug | Windows | Watch | Watch [1,2,3,4]
COMMAND	Debug.Watch[1,2,3,4]
VERSIONS	2005, 2008, 2010
CODE	vstipTool0105

You can have up to four Watch windows in Visual Studio. The reason you get all these windows is so that you can easily organize your watches into groups. Opening a particular window is easy enough: Just press Ctrl+Alt+W and then select the number of the window you want (1, 2, 3, or 4):



The only problem is that anytime you use Add Watch or QuickWatch, the watch expression always gets added to Watch 1:



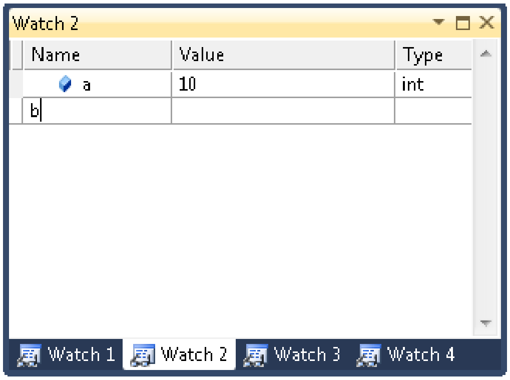
Name	Value	Type
a	10	int
b	20	int
c	30	int
d	40	int
e	50	int
f	60	int
g	70	int
h	80	int

Moving Between Windows

What if you want to move to another window? Let’s look at some of your options.

Type it in

You can just go to the window you want and type in the value you are looking for:


Watch 2

Name	Value	Type
a	10	int
b		

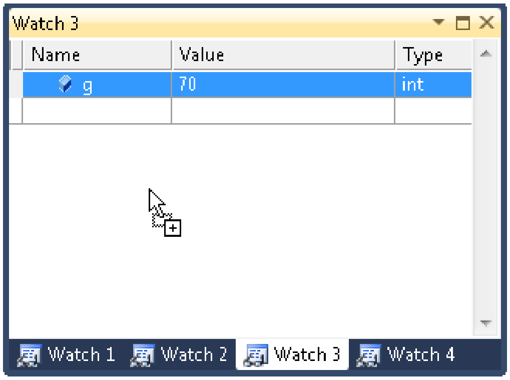
Copy and paste

You can copy and paste the value from one window to another:

Note You can copy but not cut in the Watch window. You have to delete the original value from Watch 1 if you want to actually "move" it.

Click and drag

You can click and crag to copy a value from one window to another:

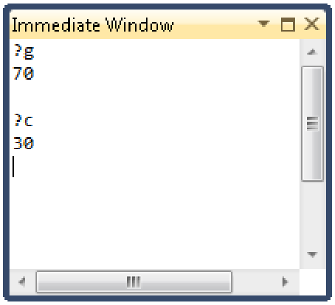

Watch 3

Name	Value	Type
g	70	int

AX.116 The Immediate Window: Simple Printing and Changing Values

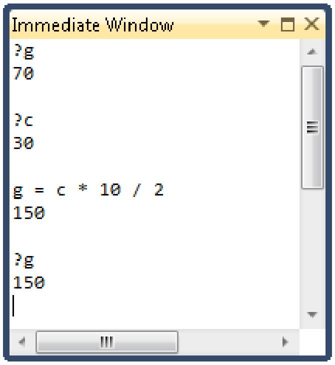
DEFAULT	Ctrl+Alt+I
VISUAL BASIC 6	Ctrl+Alt+I; Ctrl+G
VISUAL C# 2005	Ctrl+Alt+I; Ctrl+D, Ctrl+I; Ctrl+D, I
WINDOWS	Alt,D, W, I
MENU	Debug | Windows | Immediate
COMMAND	Debug.Immediate
VERSIONS	2005, 2008, 2010
CODE	vstipTool0094

The Immediate Window's versatility is great, and it can give instant feedback with information you might need to help with your application. The first thing people usually learn when using it is how to print values. Just go into debug mode, and use either **Debug.Print(variable)** or (more commonly) just **?variable**:



```
?g
70

?c
30
```

Additionally, you can modify values:



```
?g
70

?c
30

g = c * 10 / 2
150

?g
150
```

This is pretty basic assignment in these examples, but you can run pretty much any valid code to change values.

AX.117 The Immediate Window: Working with Members

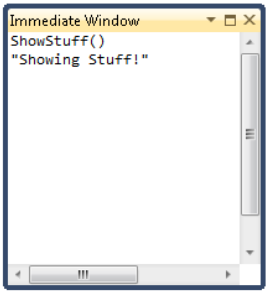
DEFAULT	Ctrl+Alt+I
VISUAL BASIC 6	Ctrl+Alt+I; Ctrl+G
VISUAL C# 2005	Ctrl+Alt+I; Ctrl+D, Ctrl+I; Ctrl+D, I
VISUAL C++ 2	Ctrl+Alt+I
VISUAL C++ 6	Ctrl+Alt+I
VISUAL STUDIO 6	Ctrl+Alt+I
WINDOWS	Alt,D, W, I
MENU	Debug | Windows | Immediate
COMMAND	Debug.Immediate
VERSIONS	2005, 2008, 2010
LANGUAGES	C#, VB
CODE	vstipTool0095

When using the Immediate Window, you work with class and object members directly. Both the traditional usage in Debug mode and the lesser-known use in Design mode is available.

Debug

You can use any method or property as long as it is in context. So, for example, when you are in debug mode, you can call any method that is in scope:

```
class Person
{
    public string ScopeFun()
    {
        return "blah";
    }

    private string ShowStuff()
    {
        return "Showing Stuff!";
    }
}
```



```
ShowStuff()
"Showing Stuff!"
```

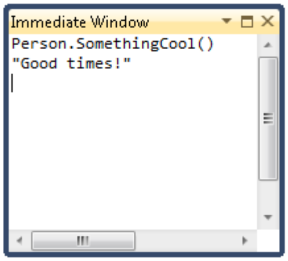
Design

Warning When working with members at design time, a build will occur. This might have unintended consequences, so make sure you have experimented with this feature a bit before you use it on production code.

> **Note** You cannot use design time expression evaluation in project types that require starting up an execution environment, including Visual Studio Tools for Office projects, web projects, Smart Device projects, and SQL projects.

A lesser-known feature is that you can work with properties and methods while in design mode. If you have static ("Shared" in VB) methods on a class, for example, you can just execute them without going into debug mode:

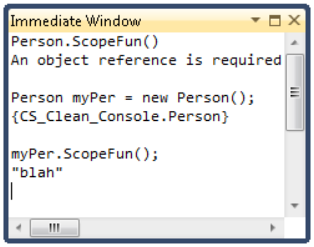
```
class Person
{
    public static string SomethingCool()
    {
        return "Good times!";
    }

    public string ScopeFun()
    {
        return "blah";
    }

    private string ShowStuff()
    {
        return "Showing Stuff!";
    }
}
```



```
Person.SomethingCool()
"Good times!"
```

For object members, you need to create an instance of the object before working with the members:

```
class Person
{
    public static string SomethingCool()
    {
        return "Good times!";
    }

    public string ScopeFun()
    {
        return "blah";
    }

    private string ShowStuff()
    {
        return "Showing Stuff!";
    }
}
```



```
Person.ScopeFun()
An object reference is required

Person myPer = new Person();
{CS_Clean_Console.Person}

myPer.ScopeFun();
"blah"
```

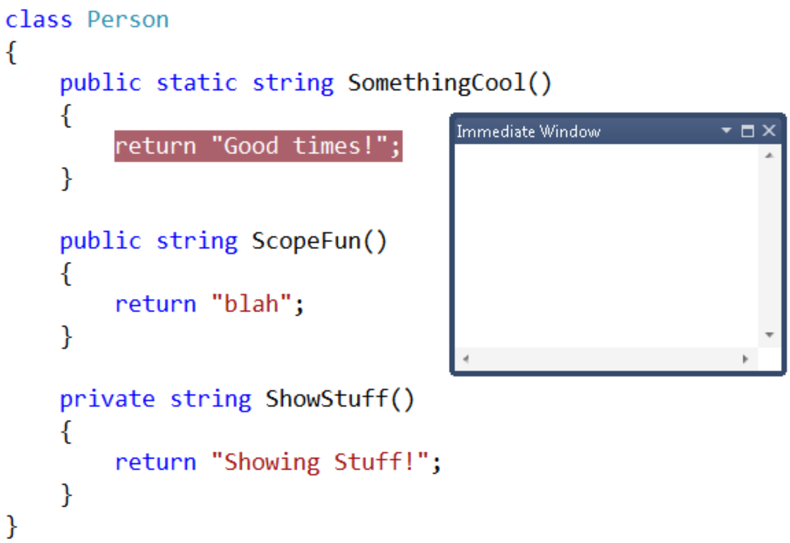
AX.118 The Immediate Window: Design-Time Breakpoints

DEFAULT	Ctrl+Alt+I
VISUAL BASIC 6	Ctrl+Alt+I; Ctrl+G
VISUAL C# 2005	Ctrl+Alt+I; Ctrl+D, Ctrl+I; Ctrl+D, I
VISUAL C++ 2	Ctrl+Alt+I
VISUAL C++ 6	Ctrl+Alt+I
VISUAL STUDIO 6	Ctrl+Alt+I
WINDOWS	Alt,D, W, I
MENU	Debug | Windows | Immediate
COMMAND	Debug.Immediate
VERSIONS	2005, 2008, 2010
LANGUAGES	C#, VB
CODE	vstipTool0096

In vstipTool0095 ("The Immediate Window: Working with Members," page A179), I demonstrated that you could do design-time execution of members. I thought it would be instructive to mention that you can also use this technique to hit breakpoints in your code.

For example, assume you have an application that has a class with a static method.

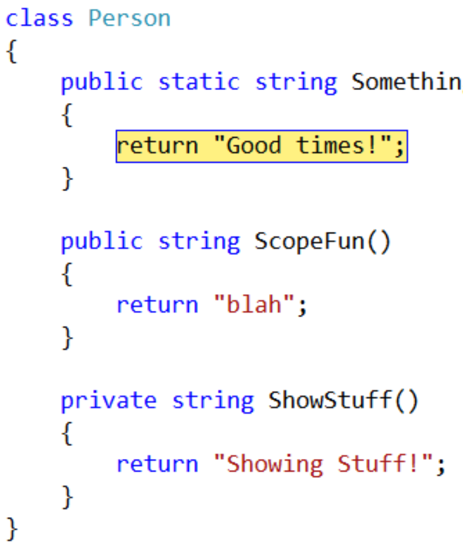
Now set a breakpoint on a line of code:



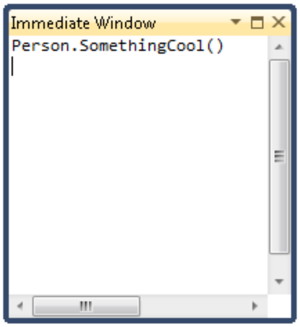
```
class Person
{
    public static string SomethingCool()
    {
        return "Good times!";
    }

    public string ScopeFun()
    {
        return "blah";
    }

    private string ShowStuff()
    {
        return "Showing Stuff!";
    }
}
```

Then, in design mode, execute the method from the Immediate Window.

It executes the code and stops at the breakpoint, ready for you to continue debugging:



```
class Person
{
    public static string SomethingCool()
    {
        return "Good times!";
    }

    public string ScopeFun()
    {
        return "blah";
    }

    private string ShowStuff()
    {
        return "Showing Stuff!";
    }
}
```



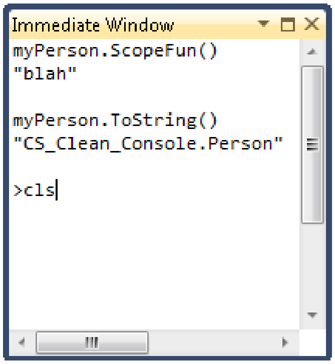
```
Person.SomethingCool()
```

This is an interesting feature of design-time execution, which you can use to quickly get to an area for debugging.

AX.119 The Immediate Window: Running Commands

DEFAULT	Ctrl+Alt+I
VISUAL BASIC 6	Ctrl+Alt+I; Ctrl+G
VISUAL C# 2005	Ctrl+Alt+I; Ctrl+D, Ctrl+I; Ctrl+D, I
VISUAL C++ 2	Ctrl+Alt+I
VISUAL C++ 6	Ctrl+Alt+I
VISUAL STUDIO 6	Ctrl+Alt+I
WINDOWS	Alt,D, W, I
MENU	Debug | Windows | Immediate
COMMAND	Debug.Immediate
VERSIONS	2005, 2008, 2010
CODE	vstipTool0098

You can run commands when you are in the Immediate Window. Just type **>[command]** to run any command. When I want to quickly clear out the Immediate Window, one of my favorite commands to run is **>cls**:



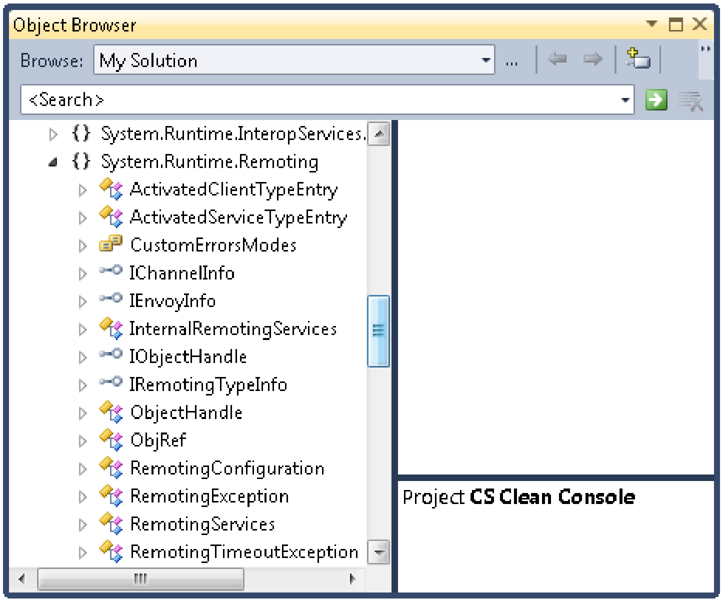
```
myPerson.ScopeFun()
"blah"

myPerson.ToString()
"CS_Clean_Console.Person"

>cls
```

Any valid command can be run in this way, so you can run any command you want from the Immediate Window.

AX.120 Class View and Object Browser Icons

VERSIONS	2005, 2008, 2010
CODE	vstipTool0076

You often encounter icons to represent symbols in a variety of places, such as when you use the Object Browser:



Browse: My Solution

Project **CS Clean Console**

The help documentation lists out the icons for you and I have listed them in the following table for your reference. You can see them online at *http://msdn.microsoft.com/en-us/library/y47ychfe.aspx*.

Class View and Object Browser Icons:

Icon	Description	Icon	Description
{ }	Namespace		Method or function
	Class		Operator
	Interface		Property
	Structure		Field or variable
	Union		Event
	Enum		Constant
	TypeDef		Enum item
	Module		Map item
	Intrinsic		External declaration
	Delegate		Macro
	Exception	<T>	Template
≡	Map	!	Unknown or error
	Global		Type forwarding
	Extension method		

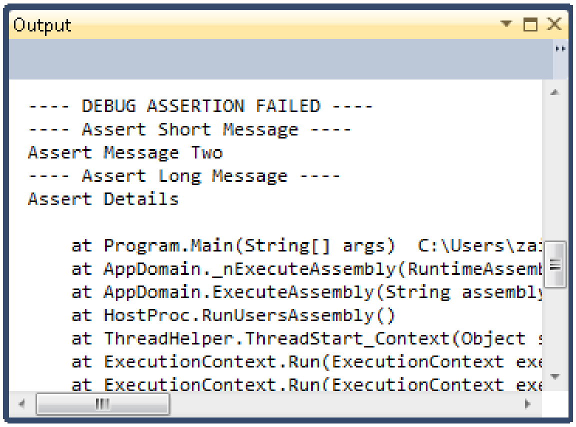
Modifier Icons:

Icon	Description
<No Signal Icon>	Public. Accessible from anywhere in this component and from any component that references it.
	Protected. Accessible from the containing class or type, or those derived from the containing class or type.
	Private. Accessible only in the containing class or type.
	Internal. Accessible only from this component.
	Friend. Accessible only from the project.
	Shortcut. A shortcut to the object.

AX.121 Output Window vs. Immediate Window

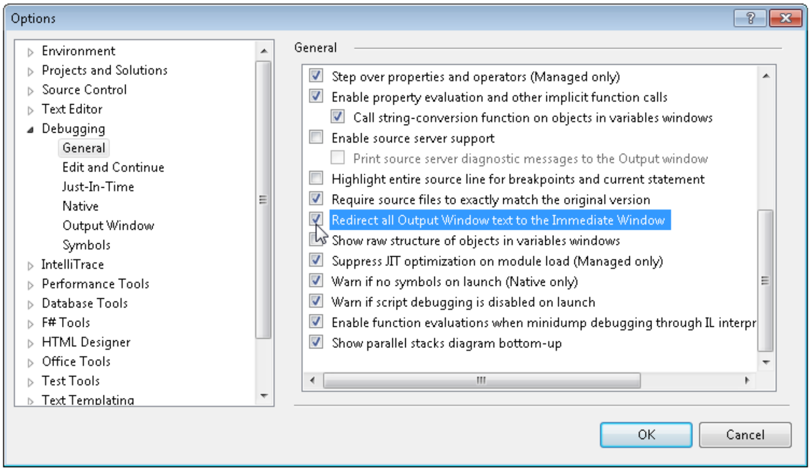
WINDOWS	Alt,T, O
MENU	Tools | Options | Debugging | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipTool0046

Depending on your settings, you might want to redirect Output window messages that you create to the Immediate window, or vice versa. So, consider your messages (Asserts, for example) that currently go to the Output window:



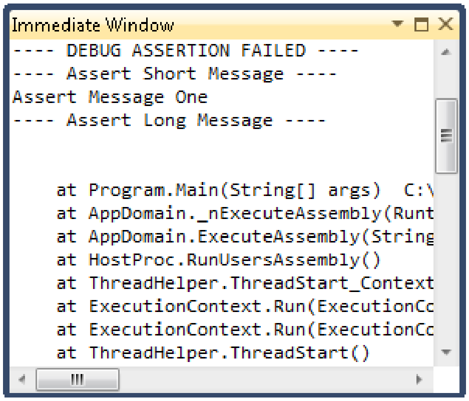
```
---- DEBUG ASSERTION FAILED ----
---- Assert Short Message ----
Assert Message Two
---- Assert Long Message ----
Assert Details

    at Program.Main(String[] args)  C:\Users\za
    at AppDomain._nExecuteAssembly(RuntimeAssem
    at AppDomain.ExecuteAssembly(String assembl
    at HostProc.RunUsersAssembly()
    at ThreadHelper.ThreadStart_Context(Object
    at ExecutionContext.Run(ExecutionContext exe
    at ExecutionContext.Run(ExecutionContext exe
```

Go to Tools | Options | Debugging | General, and select the Redirect All Output Window Text To The Immediate Window check box:



- ☑ Step over properties and operators (Managed only)
- ☑ Enable property evaluation and other implicit function calls
  - ☑ Call string-conversion function on objects in variables windows
- ☐ Enable source server support
  - ☐ Print source server diagnostic messages to the Output window
- ☐ Highlight entire source line for breakpoints and current statement
- ☑ Require source files to exactly match the original version
- ☑ Redirect all Output Window text to the Immediate Window
- ☐ Show raw structure of objects in variables windows
- ☑ Suppress JIT optimization on module load (Managed only)
- ☑ Warn if no symbols on launch (Native only)
- ☑ Warn if script debugging is disabled on launch
- ☑ Enable function evaluations when minidump debugging through IL interpr
- ☑ Show parallel stacks diagram bottom-up

Now your messages go to the Immediate window instead of the Output window:



```
---- DEBUG ASSERTION FAILED ----
---- Assert Short Message ----
Assert Message One
---- Assert Long Message ----


    at Program.Main(String[] args)  C:\
    at AppDomain._nExecuteAssembly(Runt
    at AppDomain.ExecuteAssembly(String
    at HostProc.RunUsersAssembly()
    at ThreadHelper.ThreadStart_Context
    at ExecutionContext.Run(ExecutionCo
    at ExecutionContext.Run(ExecutionCo
    at ThreadHelper.ThreadStart()
```

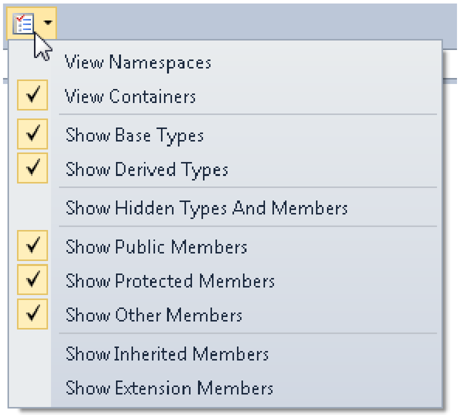
Note Not all information is redirected to the Immediate window. In this example, the results of your Assert are redirected, but some system information is always shown in the Output window.

AX.122 The Object Browser: Settings

DEFAULT	Ctrl+Alt+J
VISUAL BASIC 6	Ctrl+Alt+J; F2
VISUAL C# 2005	Ctrl+Alt+J; Ctrl+W, Ctrl+J; Ctrl+W, J
VISUAL C++ 2	Ctrl+Alt+J; Shift+Alt+F1
VISUAL C++ 6	Ctrl+Alt+J
VISUAL STUDIO 6	Ctrl+Alt+B; F2
WINDOWS	Alt,V, J
MENU	View | Object Browser
COMMAND	View.ObjectBrowser
VERSIONS	2005, 2008, 2010
CODE	vstipTool0080

Object Browser settings are a critical part of your browsing experience. They determine what you see and how much detail exists. This tip shows you how to use the settings to your advantage.

First, the settings button is located to the far right on the toolbar and looks like a sheet of paper with a check mark in it. There is quite a bit to look at, as you can see:



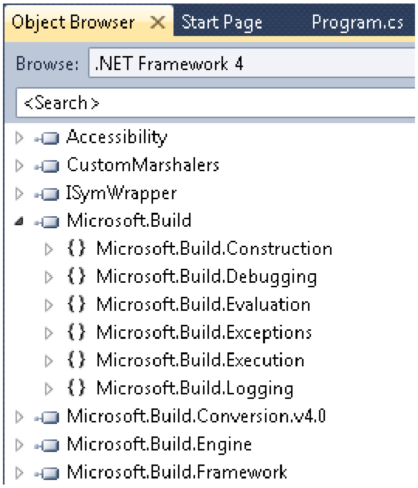
View Namespaces
✓ View Containers
✓ Show Base Types
✓ Show Derived Types
Show Hidden Types And Members
✓ Show Public Members
✓ Show Protected Members
✓ Show Other Members
Show Inherited Members
Show Extension Members

Note The options you see are based on your context in the Object Browser, so not all options may be currently available.

Containers vs. Namespaces

View Containers

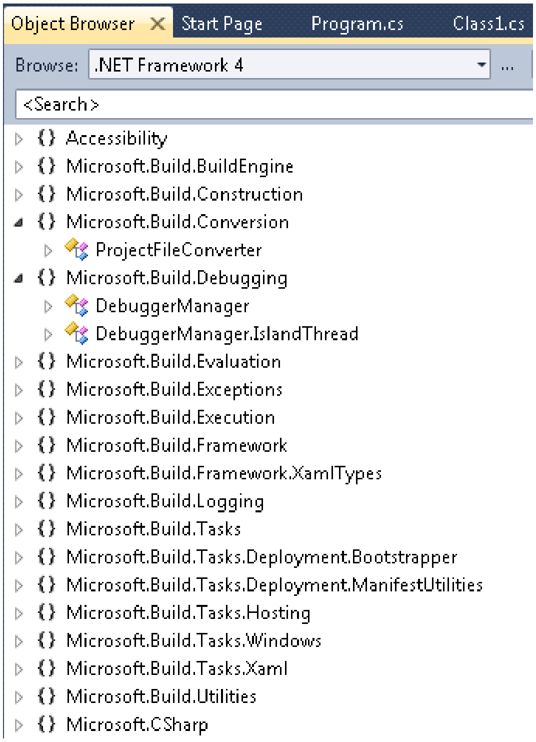
Sets the highest-level items in the Objects pane to physical containers, such as components, assemblies, source browser (.bsc) files, and output type libraries (.tlb). These expand to show the namespaces that are contained:



Object Browser
Start Page
Program.cs
Browse: .NET Framework 4
<Search>
Accessibility
CustomMarshalers
ISymWrapper
Microsoft.Build
{} Microsoft.Build.Construction
{} Microsoft.Build.Debugging
{} Microsoft.Build.Evaluation
{} Microsoft.Build.Exceptions
{} Microsoft.Build.Execution
{} Microsoft.Build.Logging
Microsoft.Build.Conversion.v4.0
Microsoft.Build.Engine
Microsoft.Build.Framework

View Namespaces

Sets the highest-level items in the Objects pane to namespaces. Namespaces stored in multiple physical containers are merged. These expand to show the items that are contained:

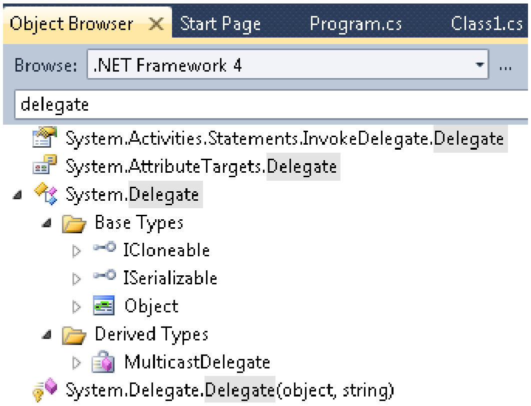


.NET Framework 4

<Search>

- Accessibility
- Microsoft.Build.BuildEngine
- Microsoft.Build.Construction
- Microsoft.Build.Conversion
  - ProjectFileConverter
- Microsoft.Build.Debugging
  - DebuggerManager
  - DebuggerManager.IslandThread
- Microsoft.Build.Evaluation
- Microsoft.Build.Exceptions
- Microsoft.Build.Execution
- Microsoft.Build.Framework
- Microsoft.Build.Framework.XamlTypes
- Microsoft.Build.Logging
- Microsoft.Build.Tasks
- Microsoft.Build.Tasks.Deployment.Bootstrapper
- Microsoft.Build.Tasks.Deployment.ManifestUtilities
- Microsoft.Build.Tasks.Hosting
- Microsoft.Build.Tasks.Windows
- Microsoft.Build.Tasks.Xaml
- Microsoft.Build.Utilities
- Microsoft.CSharp

Base and Derived Types



.NET Framework 4

delegate

- System.Activities.Statements.InvokeDelegate.Delegate
- System.AttributeTargets.Delegate
- System.Delegate
  - Base Types
    - ICloneable
    - ISerializable
    - Object
  - Derived Types
    - MulticastDelegate
- System.Delegate.Delegate(object, string)

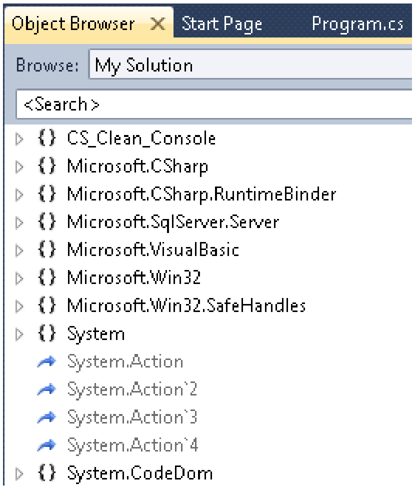
Show Base Types

Toggles showing the Base Types folder and contents.

Show Derived Types

Toggles showing the Derived Types folder and contents and is available only for Visual C++ projects and the .NET Framework.

Hidden Types and Members



CS_Clean_Console
Microsoft.CSharp
Microsoft.CSharp.RuntimeBinder
Microsoft.SqlServer.Server
Microsoft.VisualBasic
Microsoft.Win32
Microsoft.Win32.SafeHandles
System
System.Action
System.Action`2
System.Action`3
System.Action`4
System.CodeDom

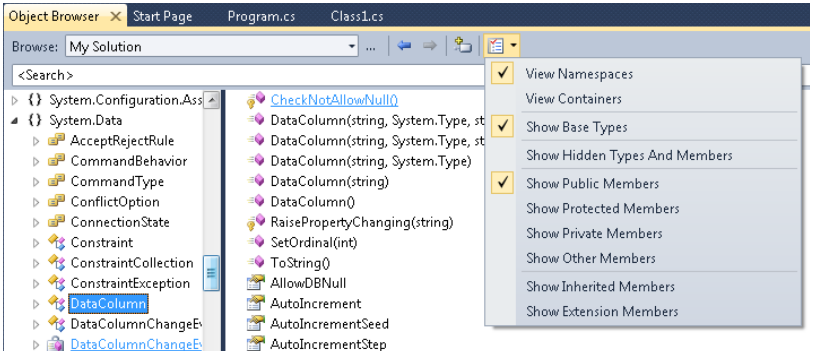
Show Hidden Types And Members

Toggles display of hidden types in the Objects pane and hidden members in the Members pane. Hidden items show up as dimmed items in the lists.

Public, Protected, Private, and Other Members

Show Public Members

Displays members that are public or protected:



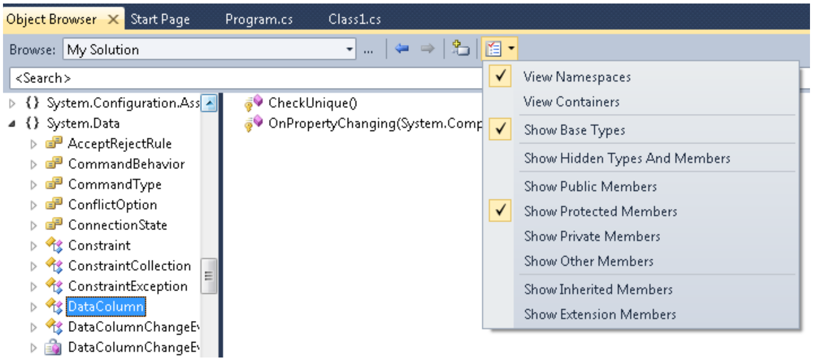
System.Configuration.Ass
System.Data
AcceptRejectRule
CommandBehavior
CommandType
ConflictOption
ConnectionState
Constraint
ConstraintCollection
ConstraintException
DataColumn
DataColumnChangeE
DataColumnChangeE

CheckNotAllowNull()
DataColumn(string, System.Type, st
DataColumn(string, System.Type, st
DataColumn(string, System.Type)
DataColumn(string)
DataColumn()
RaisePropertyChanging(string)
SetOrdinal(int)
ToString()
AllowDBNull
AutoIncrement
AutoIncrementSeed
AutoIncrementStep

✓ View Namespaces
View Containers
✓ Show Base Types
Show Hidden Types And Members
✓ Show Public Members
Show Protected Members
Show Private Members
Show Other Members
Show Inherited Members
Show Extension Members

Show Protected Members

Displays members that are protected:



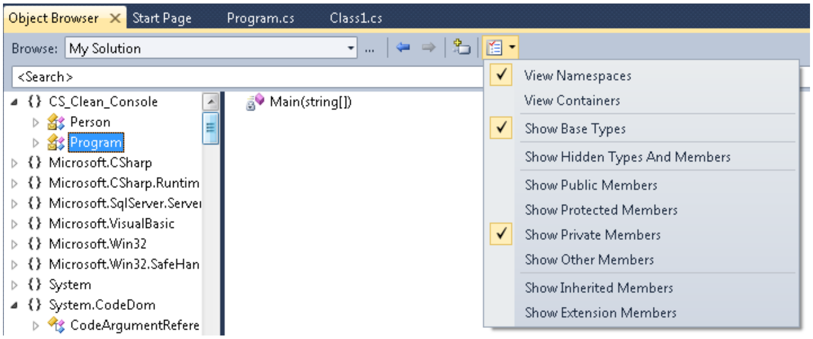
System.Configuration.Ass
System.Data
AcceptRejectRule
CommandBehavior
CommandType
ConflictOption
ConnectionState
Constraint
ConstraintCollection
ConstraintException
DataColumn
DataColumnChangeE
DataColumnChangeE

CheckUnique()
OnPropertyChanging(System.Comp

✓ View Namespaces
View Containers
✓ Show Base Types
Show Hidden Types And Members
Show Public Members
✓ Show Protected Members
Show Private Members
Show Other Members
Show Inherited Members
Show Extension Members

Show Private Members

Displays members with private accessibility:



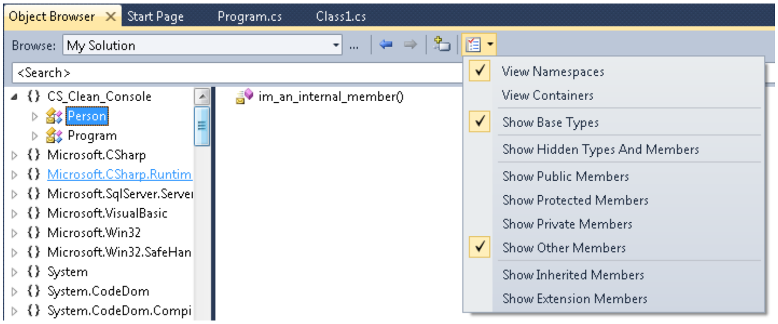
CS_Clean_Console
Person
Program
Microsoft.CSharp
Microsoft.CSharp.Runtim
Microsoft.SqlServer.Serve
Microsoft.VisualBasic
Microsoft.Win32
Microsoft.Win32.SafeHan
System
System.CodeDom
CodeArgumentRefere

Main(string[])

✓ View Namespaces
View Containers
✓ Show Base Types
Show Hidden Types And Members
Show Public Members
Show Protected Members
✓ Show Private Members
Show Other Members
Show Inherited Members
Show Extension Members

Show Other Members

Displays members that do not fall into the category of public, protected, private, or inherited:

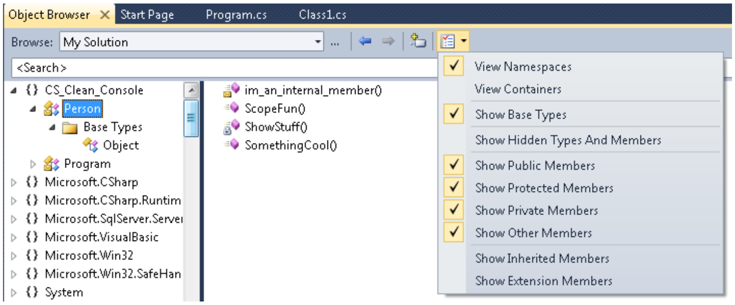


Inherited Members and Extension Methods

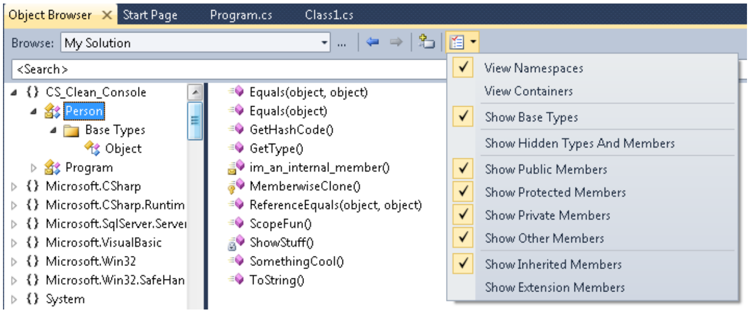
Show Inherited Members

Toggles display of inherited members in the Members pane, as shown in the following illustrations:

Off:

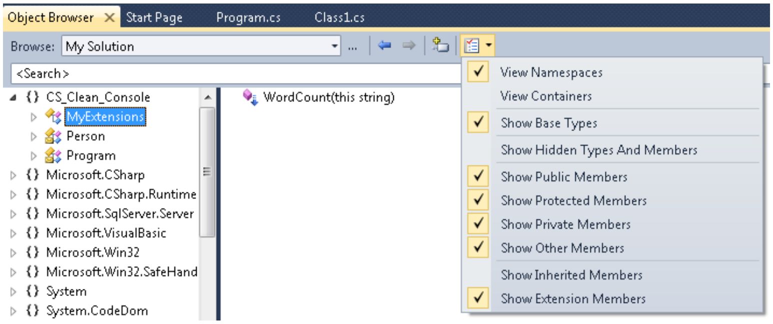


On:



Show Extension Methods

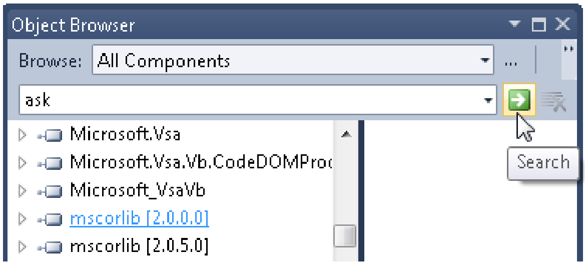
Toggles the display of extension methods in the Members pane:



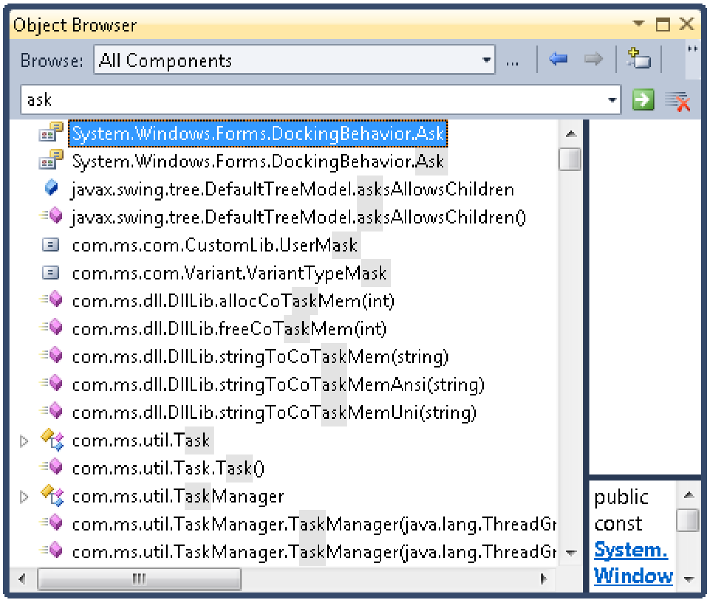
AX.123 The Object Browser: Search

DEFAULT	Ctrl+K, Ctrl+R (goto to search)
VISUAL BASIC 6	Ctrl+K, Ctrl+R (goto to search)
VISUAL C# 2005	[no shortcut] (goto to search)
VISUAL C++ 2	[no shortcut] (goto to search)
VISUAL C++ 6	Ctrl+K, Ctrl+R (goto to search)
VISUAL STUDIO 6	Ctrl+K, Ctrl+R (goto to search)
COMMAND	View.ObjectBrowserGoToSearchCombo; View.ObjectBrowserClearSearch; View.ObjectBrowserSearch
VERSIONS	2005, 2008, 2010
CODE	vstipTool0081

When you use the Object Browser, typically you need to find something fast. Search is a great way to find what you are looking for within the current browsing scope. To use search, just type the string you are looking for into the Search box and press Enter or click the Search button:

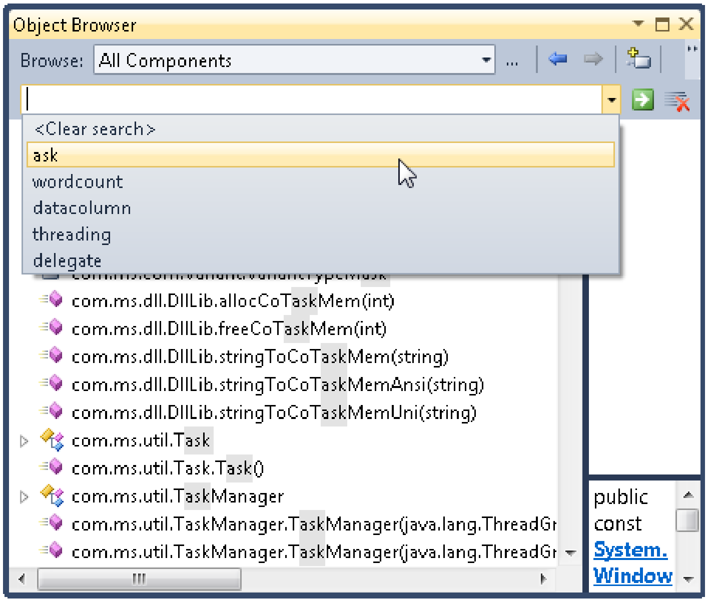


Searches are a “contains” operation and are not case-specific. For example, typing in the search term **ask** highlights the substring in the results. The search utility filters the Objects pane to show only those items that contain the search string:



Object Browser
Browse: All Components
ask
System.Windows.Forms.DockingBehavior.Ask
System.Windows.Forms.DockingBehavior.Ask
javax.swing.tree.DefaultTreeModel.asksAllowsChildren
javax.swing.tree.DefaultTreeModel.asksAllowsChildren()
com.ms.com.CustomLib.UserMask
com.ms.com.Variant.VariantTypeMask
com.ms.dll.DllLib.allocCoTaskMem(int)
com.ms.dll.DllLib.freeCoTaskMem(int)
com.ms.dll.DllLib.stringToCoTaskMem(string)
com.ms.dll.DllLib.stringToCoTaskMemAnsi(string)
com.ms.dll.DllLib.stringToCoTaskMemUni(string)
com.ms.util.Task
com.ms.util.Task.Task()
com.ms.util.TaskManager
com.ms.util.TaskManager.TaskManager(java.lang.ThreadGr
com.ms.util.TaskManager.TaskManager(java.lang.ThreadGr
public
const
System.
Window

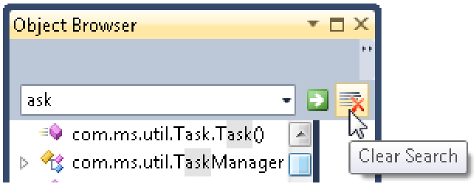
You can repeat any previous search by clicking the drop-down list arrow in the Search area:



Object Browser
Browse: All Components
<Clear search>
ask
wordcount
datacolumn
threading
delegate
com.ms.dll.DllLib.allocCoTaskMem(int)
com.ms.dll.DllLib.freeCoTaskMem(int)
com.ms.dll.DllLib.stringToCoTaskMem(string)
com.ms.dll.DllLib.stringToCoTaskMemAnsi(string)
com.ms.dll.DllLib.stringToCoTaskMemUni(string)
com.ms.util.Task
com.ms.util.Task.Task()
com.ms.util.TaskManager
com.ms.util.TaskManager.TaskManager(java.lang.ThreadGr
com.ms.util.TaskManager.TaskManager(java.lang.ThreadGr
public
const
System.
Window

This list persists even after Visual Studio is closed because the values are stored in the registry (HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\<version>\Object_Browser):

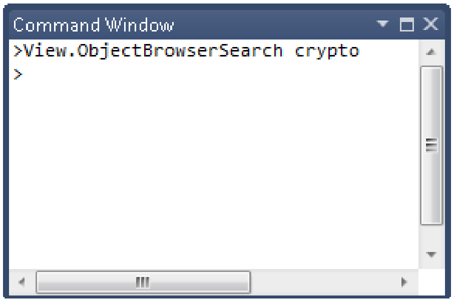
You can clear any search (and remove the filter on the Objects pane) by clicking the Clear Search button to the right of the Search box:



View.ObjectBrowserSearch Command

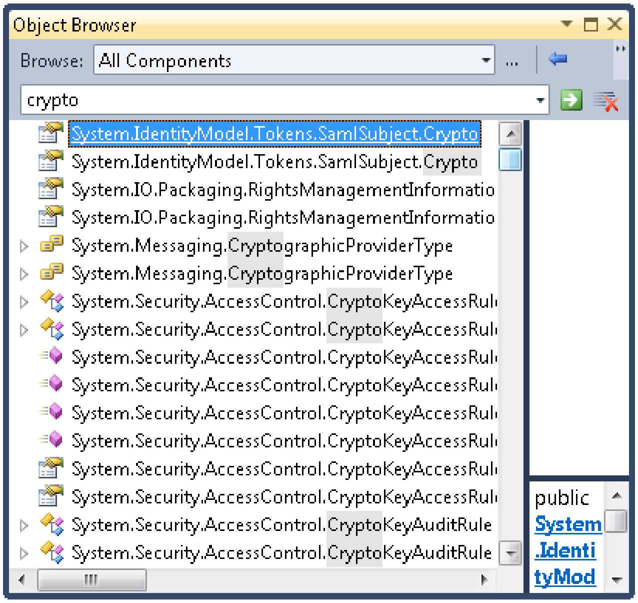
You can invoke the View.ObjectBrowserSearch command to quickly do a search by using a command. This is particularly useful if you have a search you perform frequently because you can create command aliases for your common search strings. See vstipTool0068 ("Understanding Commands: Aliases," page 113) for more information about command aliases.

The command is relatively straightforward. Just type in **View.ObjectBrowserSearch [search string]**:



```
>View.ObjectBrowserSearch crypto
>
```

This yields a result based on the current Object Browser settings:


Browse: All Components

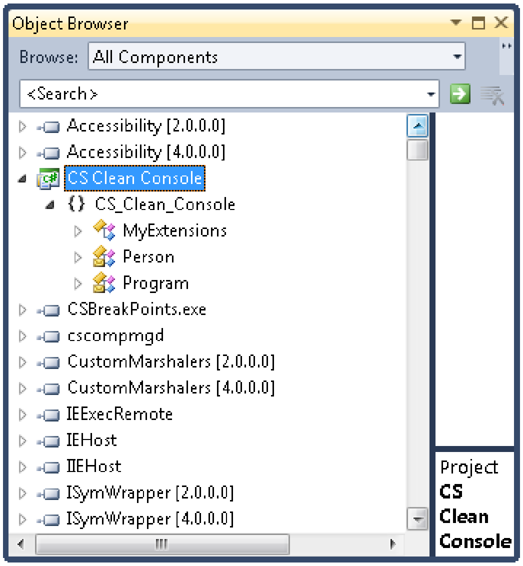
crypto
Object Browser
System.IdentityModel.Tokens.SamlSubject.Crypto
System.IdentityModel.Tokens.SamlSubject.Crypto
System.IO.Packaging.RightsManagementInformatio
System.IO.Packaging.RightsManagementInformatio
System.Messaging.CryptographicProviderType
System.Messaging.CryptographicProviderType
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAccessRul
System.Security.AccessControl.CryptoKeyAuditRule
System.Security.AccessControl.CryptoKeyAuditRule
public
System
.Identi
tyMod

AX.124 The Object Browser: Objects Pane

DEFAULT	Ctrl+Alt+J
VISUAL BASIC 6	Ctrl+Alt+J; F2
VISUAL C# 2005	Ctrl+Alt+J; Ctrl+W, Ctrl+J; Ctrl+W, J
VISUAL C++ 2	Ctrl+Alt+J; Shift+Alt+F1
VISUAL C++ 6	Ctrl+Alt+J
VISUAL STUDIO 6	Ctrl+Alt+B; F2
WINDOWS	Alt,V, J
MENU	View | Object Browser
COMMAND	View.ObjectBrowser
VERSIONS	2005, 2008, 2010
CODE	vstipTool0082

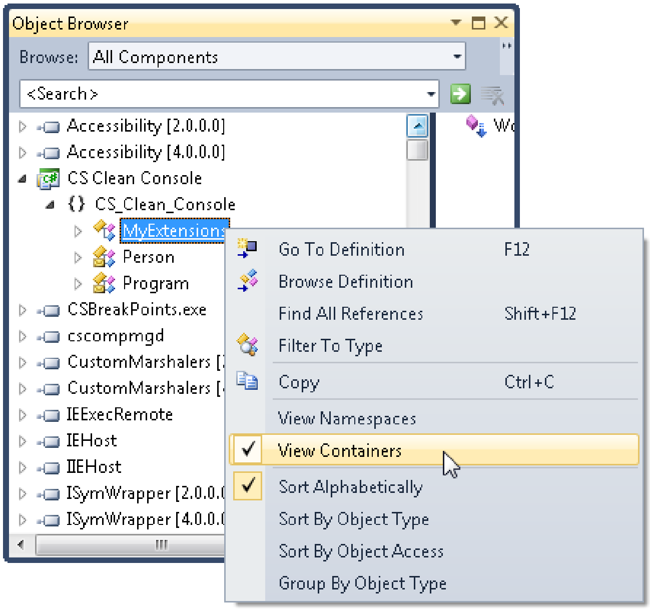
When working with the Object Browser, you will inevitably find yourself in the Objects pane located just below the Search area:

AX.124 The Object Browser: Objects Pane



The Objects pane displays an expandable list of symbols whose top-level nodes represent components or namespaces (based on your choice in the settings) available in the current browsing scope. These top-level nodes typically contain symbols that contain other symbols. To expand or collapse a node selected in the list, click its arrow sign or press Enter.

When you right-click in the Objects pane, you can see a list of options. What you see depends on the item chosen, but it generally looks something like the following:



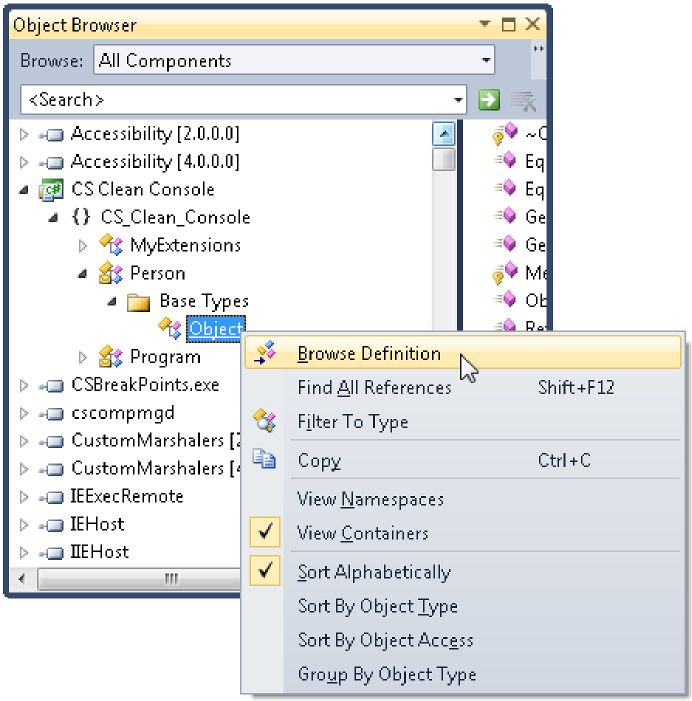
Following are descriptions of the possible options and what they do.

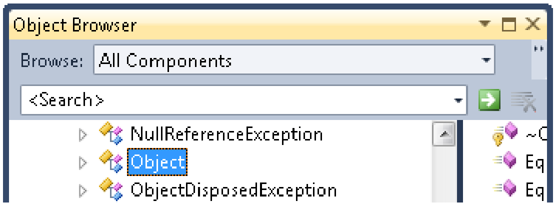
Go To Definition

Takes you to the line of code where the item is defined:

Browse Definition

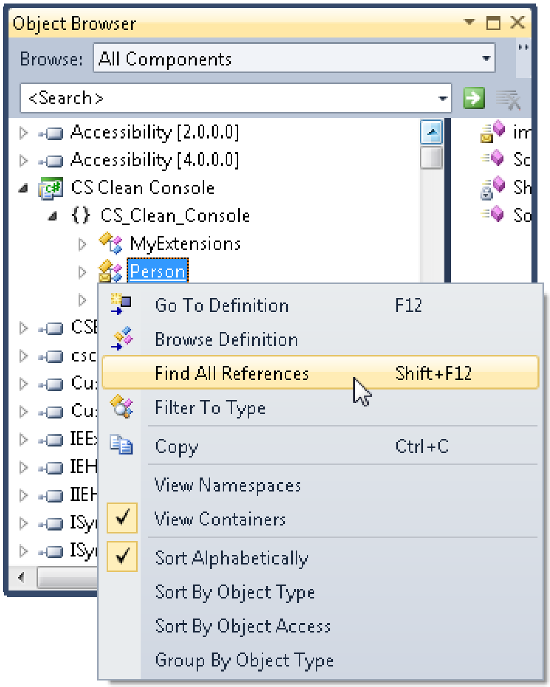
Takes you to the primary node (typically top level) for the selected symbol in the Object Browser:

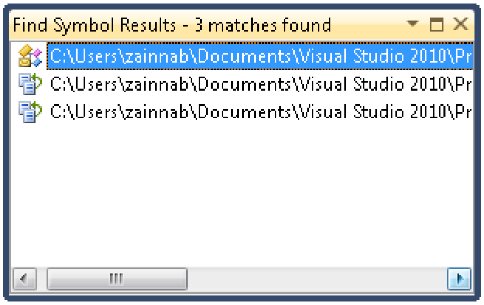




Find All References

Performs a search on the currently selected object symbol by using the current browsing scope with results shown in the Find Symbol Results dialog box:

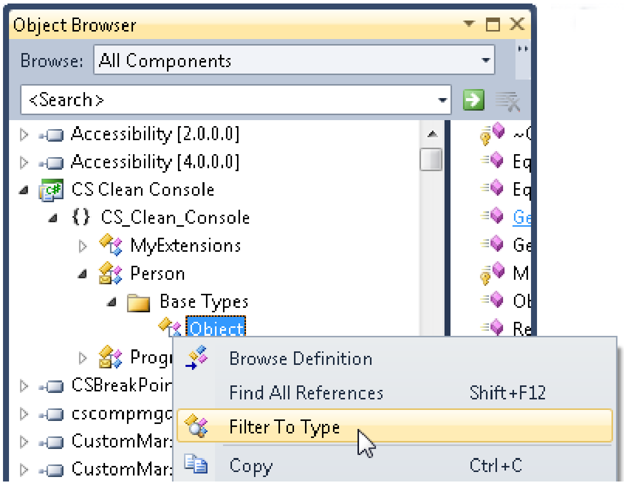


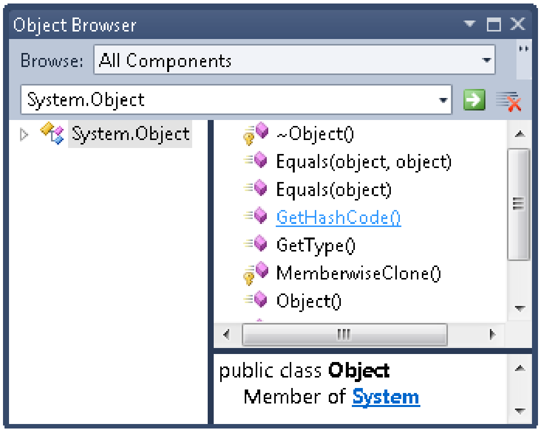


C:\Users\zainnab\Documents\Visual Studio 2010\Pr
C:\Users\zainnab\Documents\Visual Studio 2010\Pr
C:\Users\zainnab\Documents\Visual Studio 2010\Pr

Filter To Type

Shows only the selected type in the Objects pane. Essentially, it makes the selected item the top-level item in the pane. It is particularly useful for focusing on a single namespace or component:





public class **Object**
Member of **System**

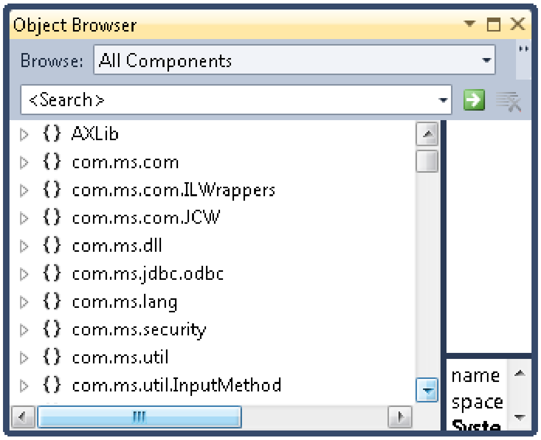
Filter To Type searches only on the item selected, and you can take the filter off by clicking the Clear Search button to the right of the Search box:

Copy

Copies a symbol reference that can be pasted into a designer and also copies the full path and name of the selected item to the clipboard.

View Namespaces

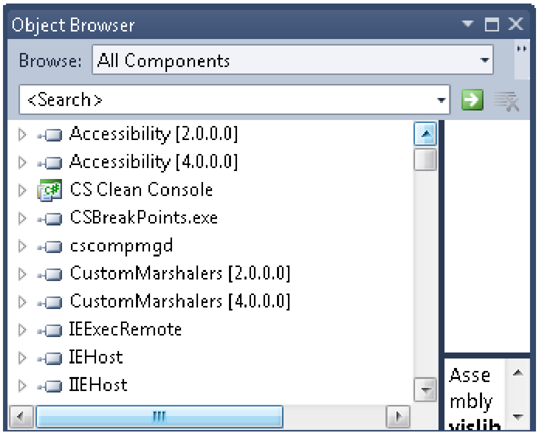
Sets the highest-level items in the Objects pane to logical namespaces. Namespaces stored in multiple physical containers are merged:



AXLib
com.ms.com
com.ms.com.ILWrappers
com.ms.com.JCW
com.ms.dll
com.ms.jdbc.odbc
com.ms.lang
com.ms.security
com.ms.util
com.ms.util.InputMethod

View Containers

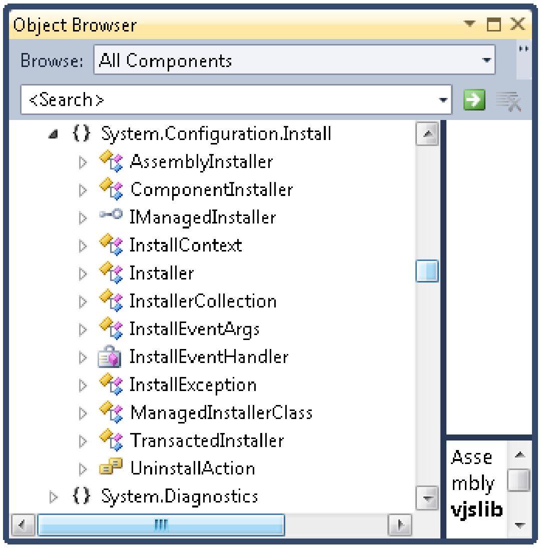
Sets the highest-level items in the Objects pane to physical containers, such as projects, components, assemblies, source browser (.bsc) files, and output type libraries (.tlb). These can be expanded to show the namespaces they contain:



Accessibility [2.0.0.0]
Accessibility [4.0.0.0]
CS Clean Console
CSBreakPoints.exe
cscompmgd
CustomMarshalers [2.0.0.0]
CustomMarshalers [4.0.0.0]
IEExecRemote
IEHost
IIEHost

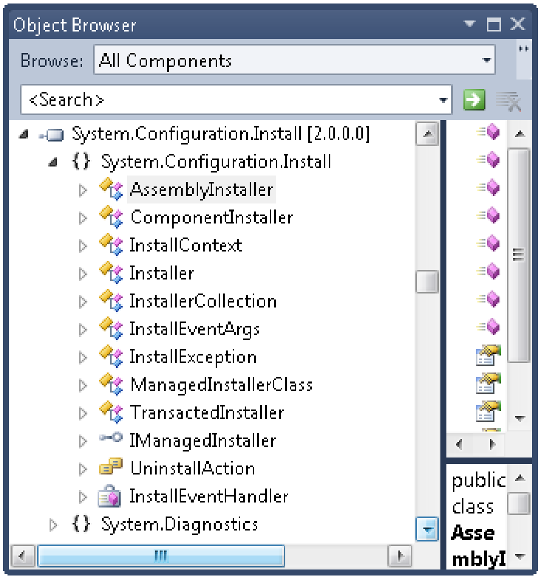
Sort Alphabetically

Lists items alphabetically by their names in ascending order:



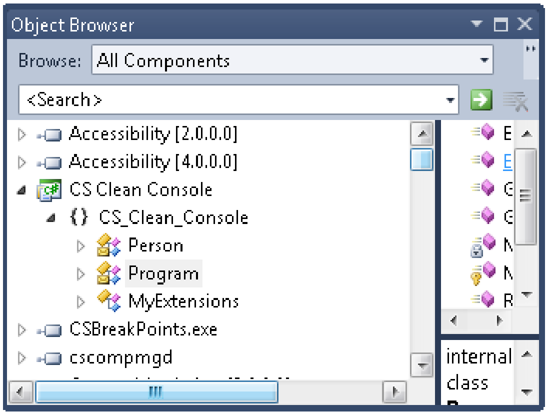
Sort By Object Type

Lists items in order of their type, such as base classes, followed by derived classes, interfaces, methods, and so forth:



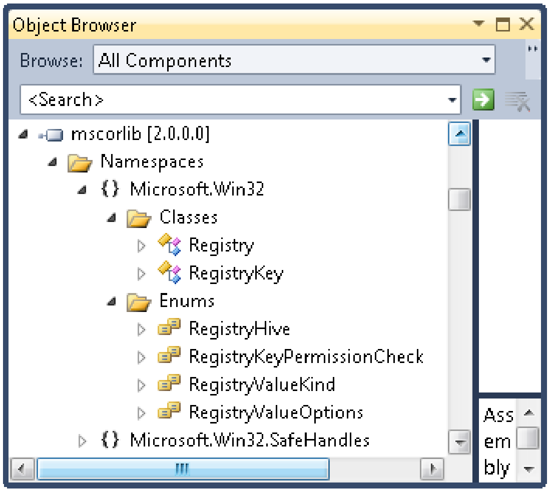
Sort By Object Access

Lists items in order of their access type, such as public or private:



Group By Object Type

Sorts items into groups by type, such as classes, interfaces, properties, methods, and so on. This is a great organizational feature:



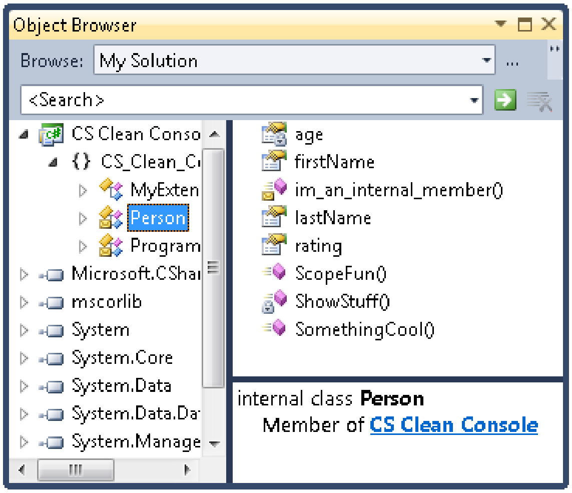
Go To Declaration

Takes you to the declaration of the symbol in the code. This is available only in Visual C++ projects.

AX.125 The Object Browser: Members Pane

DEFAULT	Ctrl+Alt+J
VISUAL BASIC 6	Ctrl+Alt+J; F2
VISUAL C# 2005	Ctrl+Alt+J; Ctrl+W, Ctrl+J; Ctrl+W, J
VISUAL C++ 2	Ctrl+Alt+J; Shift+Alt+F1
VISUAL C++ 6	Ctrl+Alt+J
VISUAL STUDIO 6	Ctrl+Alt+B; F2
WINDOWS	Alt, V, J
MENU	View | Object Browser
COMMAND	View.ObjectBrowser
VERSIONS	2005, 2008, 2010
CODE	vstipTool0083

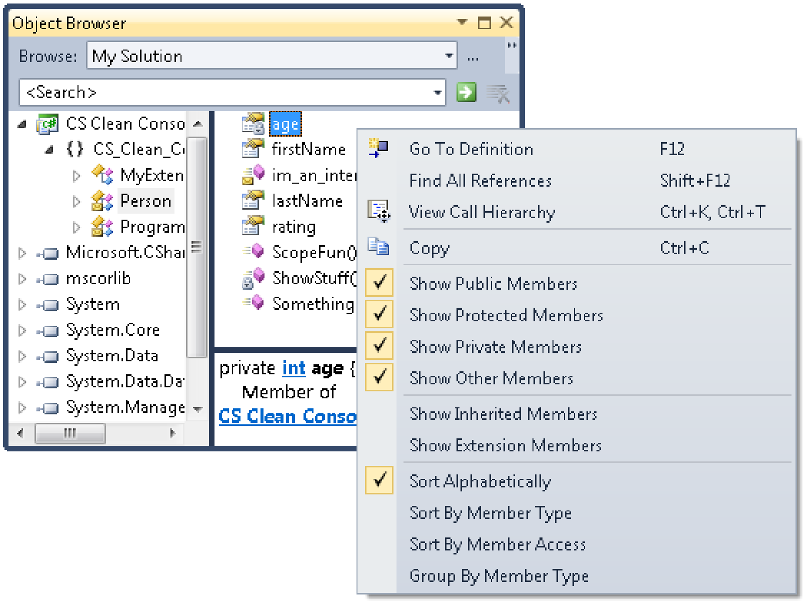
Each object can contain members such as properties, methods, events, constants, variables, and enum values. Selecting an object in the Objects pane (left) displays its members in the Members pane (right):



Browse: My Solution

internal class **Person**
Member of **CS Clean Console**

While in the Members pane, you can do several things. Many of these activities are duplicates of actions you can take in other areas of the Object Browser, so I will reference those areas to avoid duplication:



Go To Definition, Find All References, and Copy

These are the same as in the Objects pane (vstipTool0082, “The Object Browser: Objects Pane,” page A195).

View Call Hierarchy

This command is unique to the Members pane and opens up the Call Hierarchy window (vstipTool0005, “Using the Call Hierarchy,” page 310).

Show *

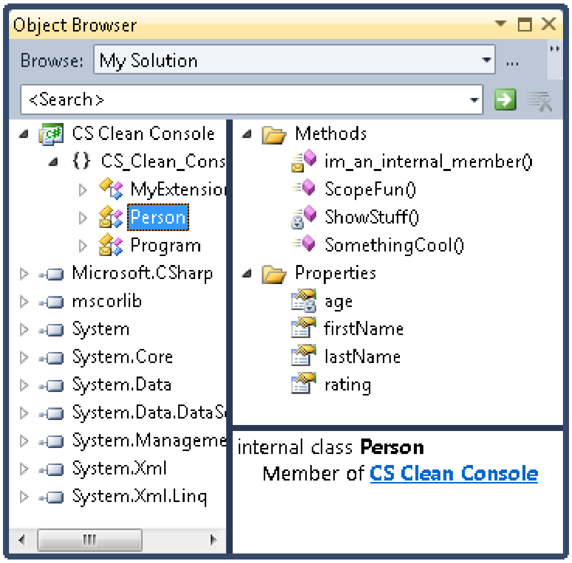
These options are the same as the ones available in the Object Browser settings (vstipTool0080, “The Object Browser: Settings,” page A186).

Sort *

These options are the same as the ones available in the Objects pane (vstipTool0082, “The Object Browser: Objects pane,” page 124).

Group By Member Type

This is similar to the same option in the Objects pane but different enough to warrant a quick look. When you use this option, members are grouped into their respective types:

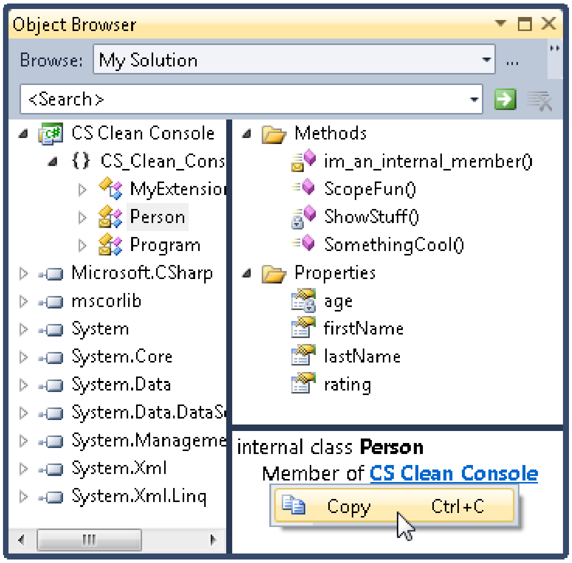


internal class **Person**
Member of CS Clean Console

AX.126 The Object Browser: Description Pane

DEFAULT	Ctrl+Alt+J
VISUAL BASIC 6	Ctrl+Alt+J; F2
VISUAL C# 2005	Ctrl+Alt+J; Ctrl+W, Ctrl+J; Ctrl+W, J
VISUAL C++ 2	Ctrl+Alt+J; Shift+Alt+F1
VISUAL C++ 6	Ctrl+Alt+J
VISUAL STUDIO 6	Ctrl+Alt+B; F2
WINDOWS	Alt, V, J
MENU	View | Object Browser
COMMAND	View.ObjectBrowser
VERSIONS	2005, 2008, 2010
CODE	vstipTool0084

The Description pane (bottom right) displays detailed information about the currently selected item (Objects pane) or member (Members pane). You can copy (right-click anywhere in the pane) data from the Description pane to the clipboard and then paste it into the code editor:



internal class **Person**
Member of CS Clean Console

The information displayed depends on the selection and can include the following:

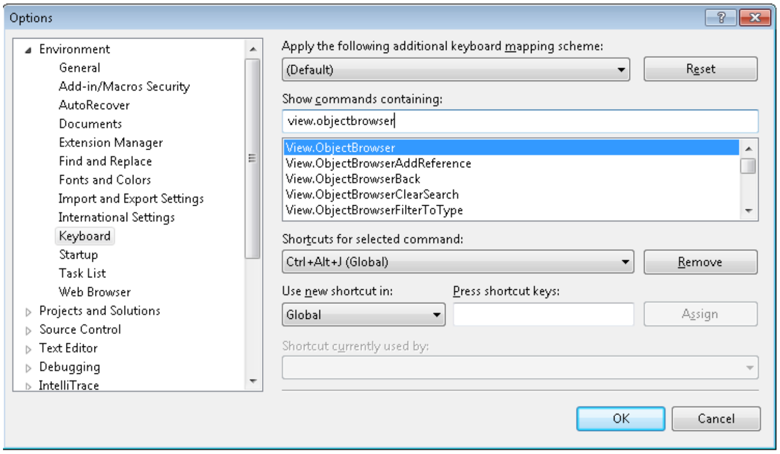
- Name and parent object
- Properties and attributes
- Syntax in the programming language of the active project
- Links to related objects and members
- Descriptions, comments, and Help text
- Version of the .NET Framework in which the object or member is included

---

AX.127 The Object Browser: Creating a Keyboard Shortcut for Add To References

COMMAND	View.ObjectBrowserAddReference
VERSIONS	2005, 2008, 2010
CODE	vstipTool0085

A command is available for almost anything you can do in the Object Browser. You can see this by going to Tools | Options | Keyboard and typing in **view.objectbrowser** in the Show Commands Containing area:



Apply the following additional keyboard mapping scheme: (Default)

Show commands containing: view.objectbrowser

View.ObjectBrowser
View.ObjectBrowserAddReference
View.ObjectBrowserBack
View.ObjectBrowserClearSearch
View.ObjectBrowserFilterToType

Shortcuts for selected command: Ctrl+Alt+J (Global)

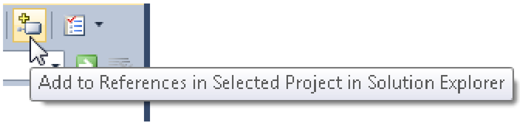
Use new shortcut in: Global

Press shortcut keys:

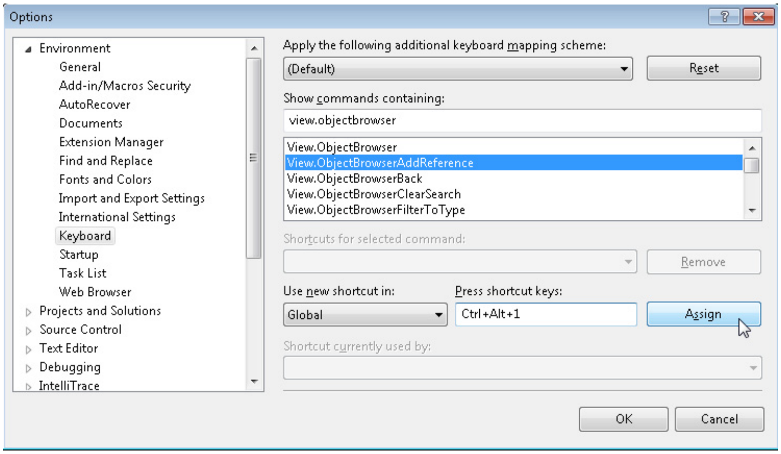
Shortcut currently used by:

This means that you can create a shortcut key for all kinds of activities. (See vstipTool0063 , “Keyboard Shortcuts: Creating New Shortcuts,” page 127.) I’ll provide a short example here.

So let’s say you want to make it easy to get the Add To References In Selected Project In Solution Explorer functionality available on the toolbar in a keyboard shortcut:



Just go to Tools | Options | Keyboard, type **view.objectbrowseraddreference** in the Show Commands Containing area, enter the shortcut you want to use, and click Assign:



Apply the following additional keyboard mapping scheme: (Default)

Show commands containing: view.objectbrowser

View.ObjectBrowser
View.ObjectBrowserAddReference
View.ObjectBrowserBack
View.ObjectBrowserClearSearch
View.ObjectBrowserFilterToType

Shortcuts for selected command:

Use new shortcut in: Global

Press shortcut keys: Ctrl+Alt+1

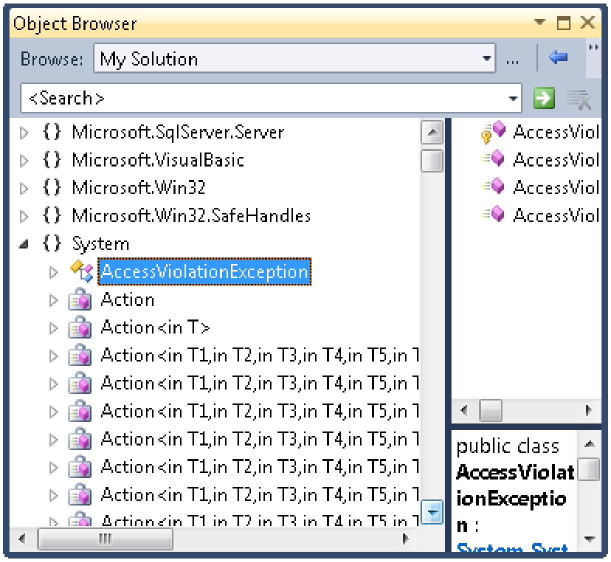
Shortcut currently used by:

You now have a shortcut key you can use anytime you want instead of having to use the toolbar.

AX.128 The Object Browser: Type-Ahead Selection

DEFAULT	Ctrl+Alt+J
VISUAL BASIC 6	Ctrl+Alt+J; F2
VISUAL C# 2005	Ctrl+Alt+J; Ctrl+W, Ctrl+J; Ctrl+W, J
VISUAL C++ 2	Ctrl+Alt+J; Shift+Alt+F1
VISUAL C++ 6	Ctrl+Alt+J
VISUAL STUDIO 6	Ctrl+Alt+B; F2
WINDOWS	Alt,V, J
MENU	View | Object Browser
COMMAND	View.ObjectBrowser
VERSIONS	2005, 2008, 2010
CODE	vstipTool0086

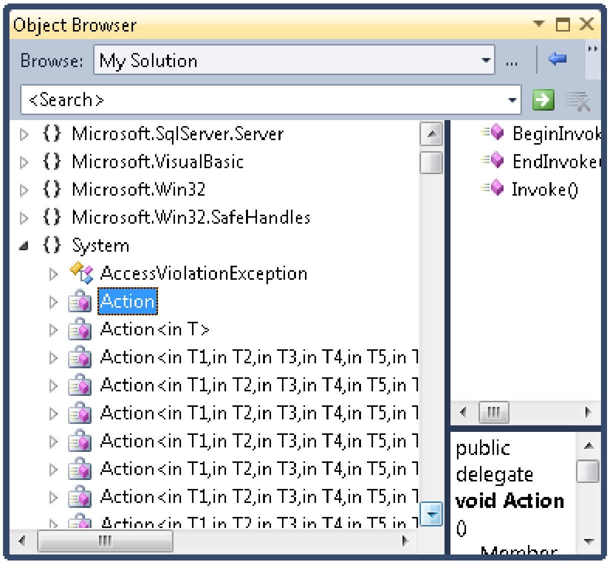
Type-ahead support is available in the Object Browser lists. For example, I have a list of items in the Objects pane:



Browse: My Solution

Object Browser
<Search>
Microsoft.SqlServer.Server
Microsoft.VisualBasic
Microsoft.Win32
Microsoft.Win32.SafeHandles
System
AccessViolationException
Action
Action<in T>
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
AccessViol
AccessViol
AccessViol
AccessViol
public class
AccessViolat
ionExceptio
n :

I type **act**, and I get the following result:

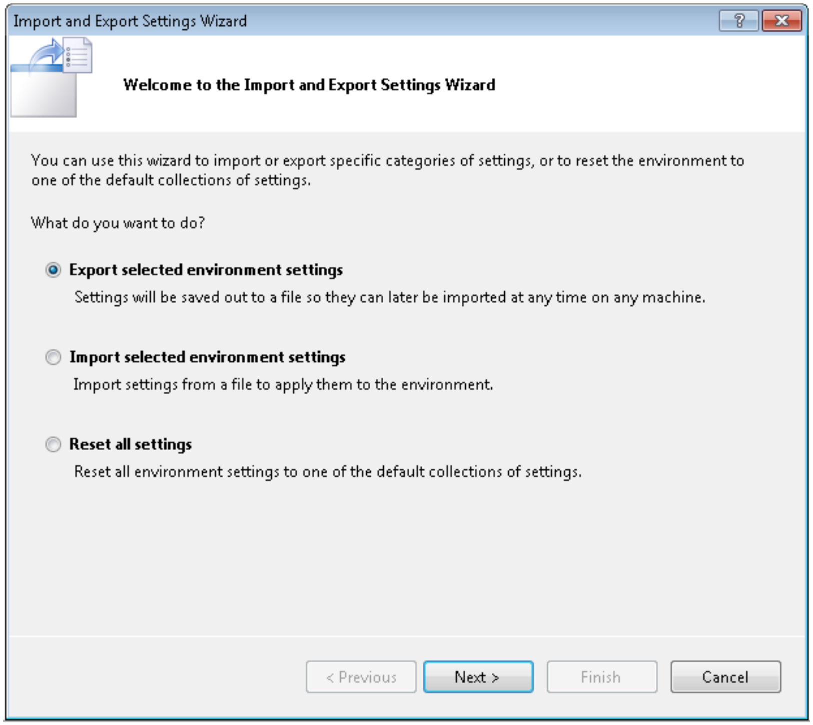

Object Browser
Browse: My Solution
<Search>
Microsoft.SqlServer.Server
Microsoft.VisualBasic
Microsoft.Win32
Microsoft.Win32.SafeHandles
System
AccessViolationException
Action
Action<in T>
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
Action<in T1,in T2,in T3,in T4,in T5,in T
BeginInvok
EndInvoke
Invoke()
public delegate **void Action** ()
Member

As you can see, it takes me to the first item that begins with *act*. I can continue typing until I find exactly what I want, or I can browse from there.

AX.129 The Object Browser: Exporting Your Settings

WINDOWS	Alt,T, I
MENU	Tools | Import and Export Settings
COMMAND	Tools.ImportandExportSettings
VERSIONS	2005, 2008, 2010
CODE	vstipTool0087

After you have the Object Browser configured the way you want it, you probably want to export the settings. In fact, you might have several different configurations you use, depending on the circumstances. For more information about exporting, see vstipEnv0021 ("Exporting Your Environment Settings," page 6). For now, let's just use a quick example to get your Object Browser settings exported. First, go to Tools | Import And Export Settings and choose Export Selected Environment Settings:



Import and Export Settings Wizard

Welcome to the Import and Export Settings Wizard

You can use this wizard to import or export specific categories of settings, or to reset the environment to one of the default collections of settings.

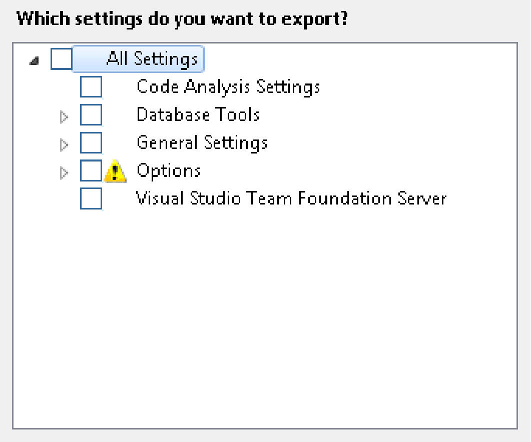
What do you want to do?

● **Export selected environment settings**
Settings will be saved out to a file so they can later be imported at any time on any machine.

○ **Import selected environment settings**
Import settings from a file to apply them to the environment.

○ **Reset all settings**
Reset all environment settings to one of the default collections of settings.

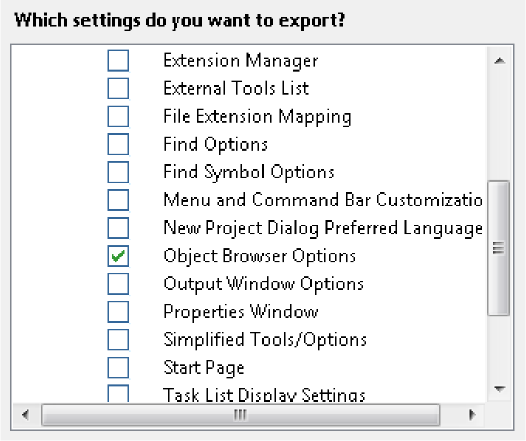
Click Next, and then clear the All Settings box to clear out all the currently selected items:



Which settings do you want to export?

- ☐ All Settings
  - ☐ Code Analysis Settings
  - ☐ Database Tools
  - ☐ General Settings
  - ☐ Options
  - ☐ Visual Studio Team Foundation Server

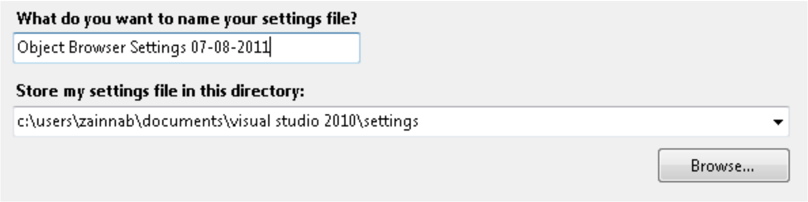
Now select Object Browser Options under General Settings:



Which settings do you want to export?

- ☐ Extension Manager
- ☐ External Tools List
- ☐ File Extension Mapping
- ☐ Find Options
- ☐ Find Symbol Options
- ☐ Menu and Command Bar Customizatio
- ☐ New Project Dialog Preferred Language
- ☑ Object Browser Options
- ☐ Output Window Options
- ☐ Properties Window
- ☐ Simplified Tools/Options
- ☐ Start Page
- ☐ Task List Display Settings

Click Next, and then give the .vssettings file a name and the path to store it in:



What do you want to name your settings file?
Object Browser Settings 07-08-2011

Store my settings file in this directory:
c:\users\zainnab\documents\visual studio 2010\settings

Click Finish, click Close, and you are all set to go. Anytime you need these settings again, you can import them.

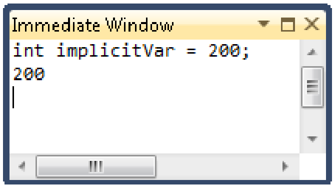
AX.130 The Immediate Window: Implicit Variables

DEFAULT	Ctrl+Alt+I
VISUAL BASIC 6	Ctrl+Alt+I; Ctrl+G
VISUAL C# 2005	Ctrl+Alt+I; Ctrl+D, Ctrl+I; Ctrl+D, I
VISUAL C++ 2	Ctrl+Alt+I
VISUAL C++ 6	Ctrl+Alt+I
VISUAL STUDIO 6	Ctrl+Alt+I
WINDOWS	Alt,D, W, I
MENU	Debug | Windows | Immediate
COMMAND	Debug.Immediate
VERSIONS	2005, 2008, 2010
LANGUAGES	C#, VB
CODE	vstipTool0100

In the Immediate Window, you can create implicit variables for use in your debugging efforts. These implicit variables never go out of scope and can be treated as any other variable. They are approached differently in VB and C#.

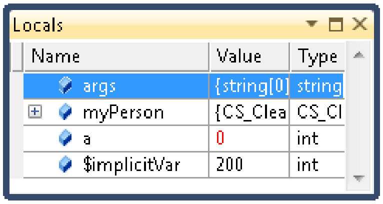
C#

To create an implicit variable in C#, just declare any variable in the Immediate window:



```
int implicitVar = 200;
200
```

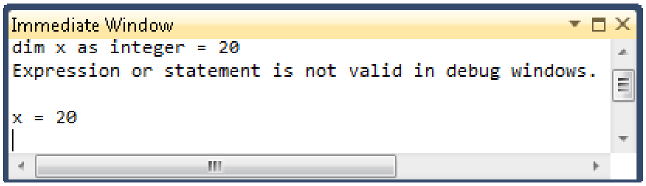
If you are in debug mode, you can actually see the variable in your Locals window. Implicit variables show up with a "$" character in front of them:



Name	Value	Type
args	{string[0]	string
myPerson	{CS_Clea	CS_Cl
a	0	int
$implicitVar	200	int

VB

In VB, you cannot declare implicit variables in the Immediate Window. But if you use an undeclared variable, an implicit variable is created automatically. Unfortunately, VB implicit variables are not listed in the Locals window:



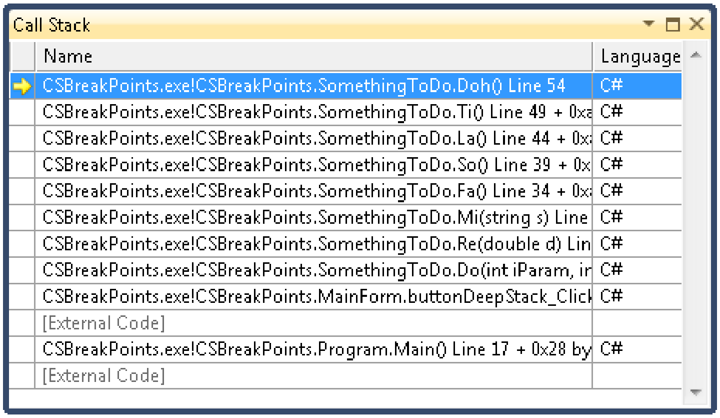
```
dim x as integer = 20
Expression or statement is not valid in debug windows.

x = 20
```

AX.131 Show External Code

VERSIONS	2005, 2008, 2010
CODE	vstipDebug0031

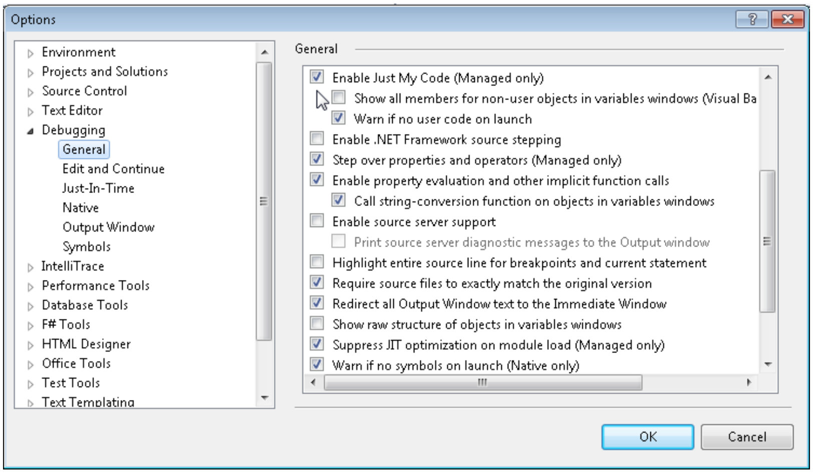
The Call Stack window provides an option to show external code. Let's start with the basics. When you are in break mode and you look at a "normal" call stack, you see the following:

Note The Call Stack window (Ctrl+Alt+C) is only available while debugging.



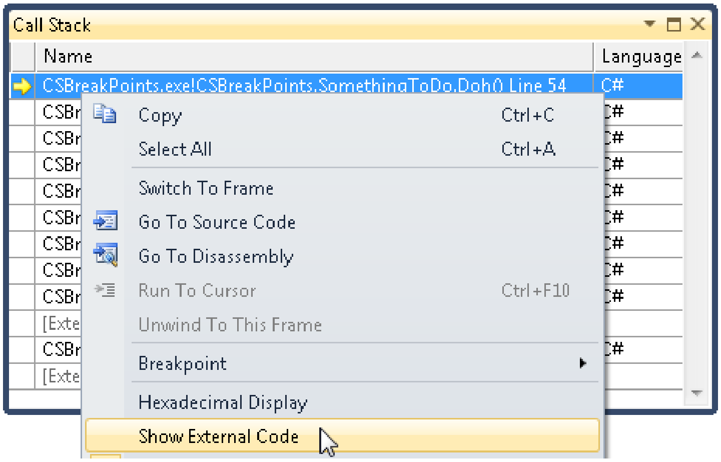
Name	Language
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Doh() Line 54	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Ti() Line 49 + 0xa	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.La() Line 44 + 0x	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.So() Line 39 + 0x	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Fa() Line 34 + 0x	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Mi(string s) Line	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Re(double d) Lin	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Do(int iParam, ir	C#
CSBreakPoints.exe!CSBreakPoints.MainForm.buttonDeepStack_Click	C#
[External Code]	
CSBreakPoints.exe!CSBreakPoints.Program.Main() Line 17 + 0x28 by	C#
[External Code]	

Let's define what "normal" is in this case. Essentially, what you see here is determined by the Enable Just My Code (Managed Only) setting in Tools | Options | Debugging | General:

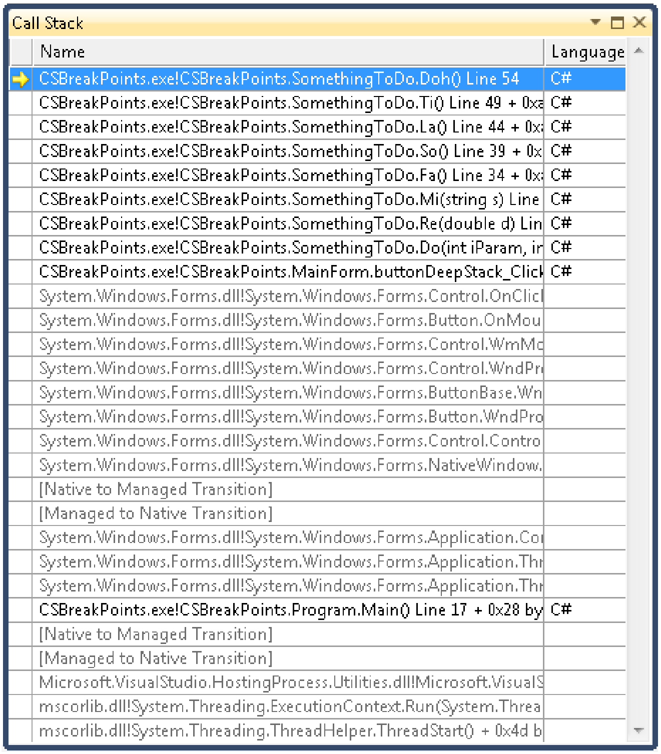


☑ Enable Just My Code (Managed only)
  ☐ Show all members for non-user objects in variables windows (Visual Ba
  ☑ Warn if no user code on launch
☐ Enable .NET Framework source stepping
☑ Step over properties and operators (Managed only)
☑ Enable property evaluation and other implicit function calls
  ☑ Call string-conversion function on objects in variables windows
☐ Enable source server support
  ☐ Print source server diagnostic messages to the Output window
☐ Highlight entire source line for breakpoints and current statement
☑ Require source files to exactly match the original version
☑ Redirect all Output Window text to the Immediate Window
☐ Show raw structure of objects in variables windows
☑ Suppress JIT optimization on module load (Managed only)
☑ Warn if no symbols on launch (Native only)

This setting is on by default, and it is the reason you see the "[External Code]" sections in your Call Stack. Selecting Enable Just My Code (Managed Only) means that you want to see your code without any information getting in the way. If you want to see the details of "[External Code]," just right-click anywhere in the Call Stack and choose Show External Code:

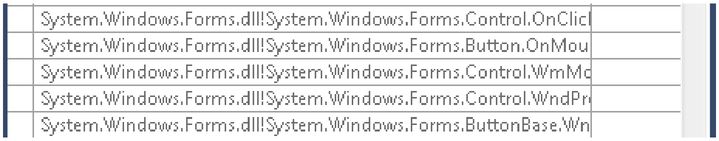


Now you should be able to see the external calls:


Call Stack

Name	Language
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Doh() Line 54	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Ti() Line 49 + 0xa	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.La() Line 44 + 0x	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.So() Line 39 + 0x	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Fa() Line 34 + 0x	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Mi(string s) Line	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Re(double d) Lin	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Do(int iParam, ir	C#
CSBreakPoints.exe!CSBreakPoints.MainForm.buttonDeepStack_Click	C#
System.Windows.Forms.dll!System.Windows.Forms.Control.OnClicl	
System.Windows.Forms.dll!System.Windows.Forms.Button.OnMou	
System.Windows.Forms.dll!System.Windows.Forms.Control.WmMc	
System.Windows.Forms.dll!System.Windows.Forms.Control.WndPr	
System.Windows.Forms.dll!System.Windows.Forms.ButtonBase.Wn	
System.Windows.Forms.dll!System.Windows.Forms.Button.WndPro	
System.Windows.Forms.dll!System.Windows.Forms.Control.Contro	
System.Windows.Forms.dll!System.Windows.Forms.NativeWindow.	
[Native to Managed Transition]	
[Managed to Native Transition]	
System.Windows.Forms.dll!System.Windows.Forms.Application.Co	
System.Windows.Forms.dll!System.Windows.Forms.Application.Thr	
System.Windows.Forms.dll!System.Windows.Forms.Application.Thr	
CSBreakPoints.exe!CSBreakPoints.Program.Main() Line 17 + 0x28 by	C#
[Native to Managed Transition]	
[Managed to Native Transition]	
Microsoft.VisualStudio.HostingProcess.Utilities.dll!Microsoft.VisualS	
mscorlib.dll!System.Threading.ExecutionContext.Run(System.Threa	
mscorlib.dll!System.Threading.ThreadHelper.ThreadStart() + 0x4d b	

The grey part is where "[External Code]" used to be. Let's zoom in on a couple of the entries:



```
System.Windows.Forms.dll!System.Windows.Forms.Control.OnClicl
System.Windows.Forms.dll!System.Windows.Forms.Button.OnMou
System.Windows.Forms.dll!System.Windows.Forms.Control.WmMc
System.Windows.Forms.dll!System.Windows.Forms.Control.WndPr
System.Windows.Forms.dll!System.Windows.Forms.ButtonBase.Wn
```

Notice that now you are looking into the details of what is happening. It remains this way until you turn it off again.

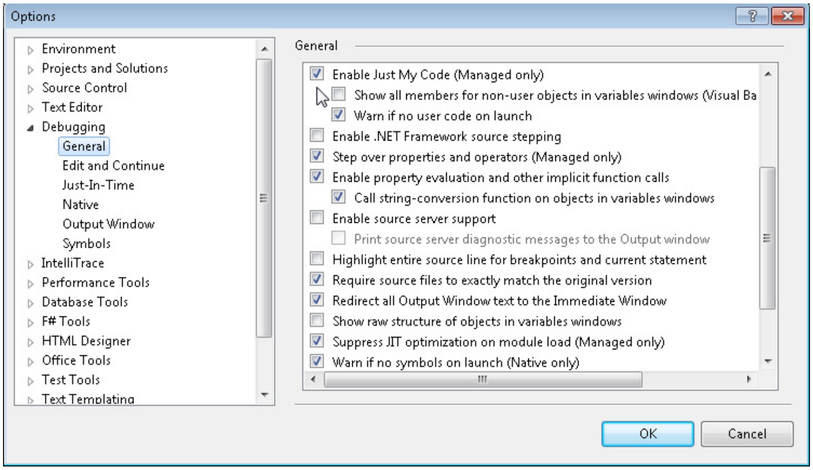
By the way, if you ever need to turn this feature off, just right-click the Call Stack again and select Show External Code. It turns off this feature, and you are back to the original view.

There is no option to turn this feature on and off, but just for reference, this setting is stored in the registry at HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\10.0\Debugger under ShowExternalCode.

AX.132 Understanding Just My Code

WINDOWS	Alt,T, O
MENU	Tools | Options
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
LANGUAGES	C++ (managed only), C#, VB
CODE	vstipDebug0032

You often want to debug just the code you have written and exclude any Framework or external component code that you are using. If you go to Tools | Options | Debugging | General, you can find an option to Enable Just My Code (Managed Only):

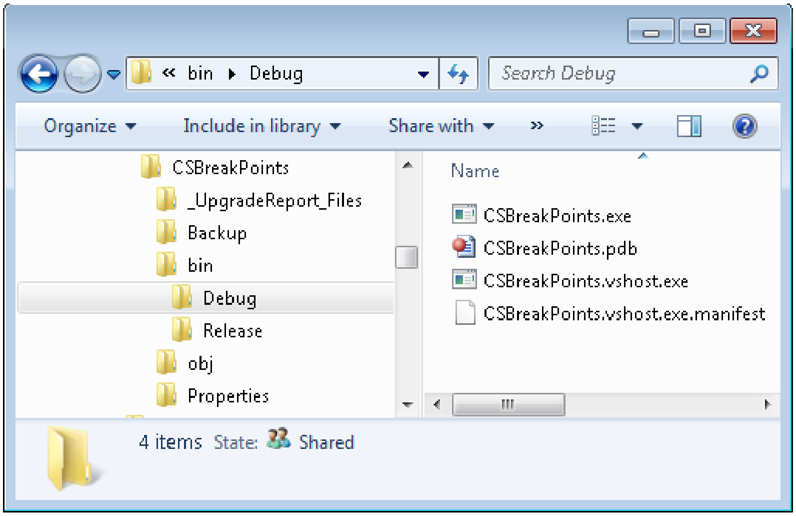


☑ Enable Just My Code (Managed only)
  ☐ Show all members for non-user objects in variables windows (Visual Ba
  ☑ Warn if no user code on launch
☐ Enable .NET Framework source stepping
☑ Step over properties and operators (Managed only)
☑ Enable property evaluation and other implicit function calls
  ☑ Call string-conversion function on objects in variables windows
☐ Enable source server support
  ☐ Print source server diagnostic messages to the Output window
☐ Highlight entire source line for breakpoints and current statement
☑ Require source files to exactly match the original version
☑ Redirect all Output Window text to the Immediate Window
☐ Show raw structure of objects in variables windows
☑ Suppress JIT optimization on module load (Managed only)
☑ Warn if no symbols on launch (Native only)

Lots of people wonder what "Just My Code" really means. So let's start with what the documentation says (*http://msdn.microsoft.com/en-us/library/h5e30exc.aspx*):

"To distinguish user code from non-user code, Just My Code looks at three things: DBG Files, PDB files, and optimization."

DBG and PDB files



One way to figure out what is "your code" is to look and see whether it has DBG and/or PDB files. In case you didn't know, DBG files have been superseded by the PDB format.

The PDB extension stands for "program database." It holds the debugging information that was introduced in Visual C++ version 1.0. You can find out more about PDB files at *http://support.microsoft.com/kb/121366/en-us*. This is typically deep-level information about the source, so if you have these files, either it's your code or someone trusted you enough to give them to you. In other words, it is basically "yours."

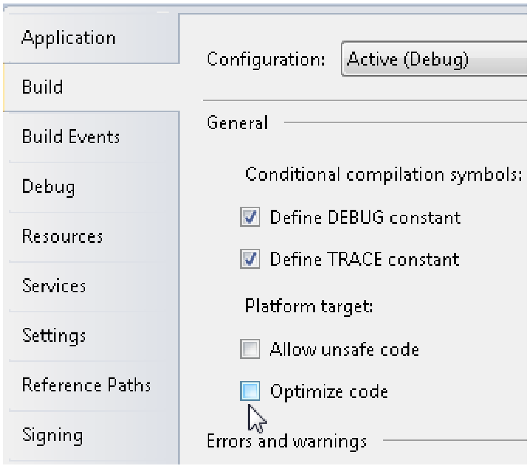
Optimization

When optimization is turned off (the default setting for Debug builds), it factors into the code being considered "yours" as well. As stated in the documentation mentioned earlier, the optimization "option enables or disables optimizations performed by the compiler to make your output file smaller, faster, and more efficient."

Many optimizations are available, such as optimizing for application speed or the size of your program. You can get see the full list of features at *http://msdn.microsoft.com/en-us/library/k1ack8f1.aspx*. Basically, optimization does many things that are great for a shipping application but not for one that you are currently debugging.

C#

In C#, this is found in the Project properties on the Build tab:



Configuration: Active (Debug)

Conditional compilation symbols:

☑ Define DEBUG constant

☑ Define TRACE constant

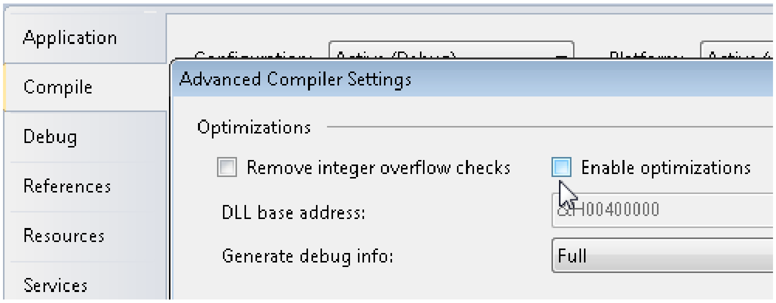
Platform target:

☐ Allow unsafe code

☐ Optimize code

VB

In VB, it is in the Project properties on the Compile tab, but you have to click the Advanced Compiler Settings button:



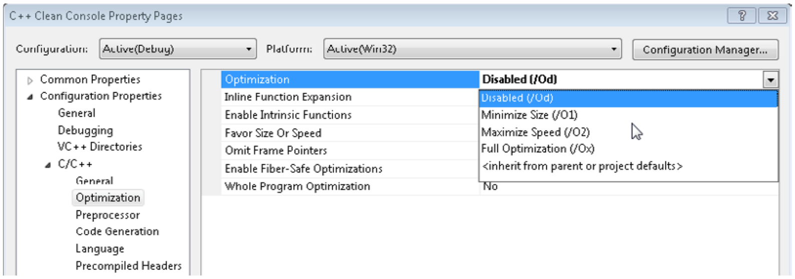
☐ Remove integer overflow checks ☐ Enable optimizations

DLL base address: &H00400000

Generate debug info: Full

C++

In C++, you can find optimization options under the Project properties, under C/C++, Optimization:



C++ Clean Console Property Pages

Configuration: Active(Debug)

Platform: Active(Win32)

Optimization: Disabled (/Od)

Whole Program Optimization: No

Finally

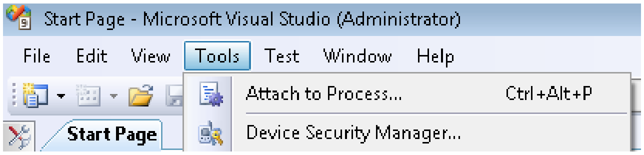
So if the PDB information is there and optimization is *not* turned on, the code is considered “yours” as far as Visual Studio is concerned.

AX.133 Attach To Process (Tools vs. Debug Menu)

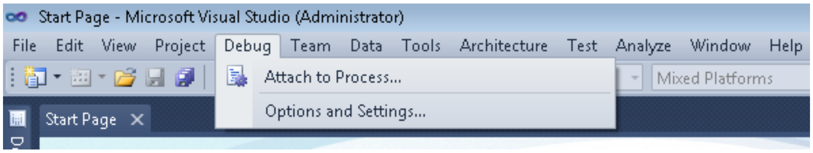
DEFAULT	Ctrl+Alt+P
VISUAL STUDIO 6	Ctrl+Alt+P; Ctrl+Shift+R
VISUAL C# 2005	Ctrl+Alt+P
VISUAL C++ 2	Ctrl+Alt+P
VISUAL C++ 6	Ctrl+Alt+P
VISUAL STUDIO 6	Ctrl+Alt+P; Ctrl+Shift+R
WINDOWS	Alt, T, P; Alt, D, P, Enter
MENU	Tools | Attach to Process; Debug | Attach to Process
COMMAND	Tools.AttachtoProcess; Debug.AttachtoProcess
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0033

I decided to figure out what the difference is between Debug | Attach To Process and Tools | Attach To Process. I’ll spare you the suspense: They are the same.

OK so why are they there? The answer is simple: Prior to Visual Studio 2010, when you didn’t have a project open in Visual Studio, it would not show the Debug menu. So the only way you could use Attach To Process was to use the Tools menu:


Start Page - Microsoft Visual Studio (Administrator)
File
Edit
View
Tools
Test
Window
Help
Attach to Process... Ctrl+Alt+P
Device Security Manager...
Start Page

Beginning in Visual Studio 2010, the Debug menu is available even when a project isn't open:



Essentially, the redundancy is unnecessary in Visual Studio 2010, but it actually served a purpose in prior versions.

AX.134 The Immediate Window: Running WinDbg and SOS (Son of Strike) Commands

DEFAULT	Ctrl+Alt+I
VISUAL BASIC 6	Ctrl+Alt+I; Ctrl+G
VISUAL C# 2005	Ctrl+Alt+I; Ctrl+D, Ctrl+I; Ctrl+D, I
VISUAL C++ 2	Ctrl+Alt+I
VISUAL C++ 6	Ctrl+Alt+I
VISUAL STUDIO 6	Ctrl+Alt+I
WINDOWS	Alt,D, W, I
MENU	Debug | Windows | Immediate
COMMAND	Debug.Immediate
VERSIONS	2005, 2008, 2010
CODE	vstipTool0097

It would take a long time to go into detail about WinDbg and SOS (Son of Strike), so I will avoid that here. I want to give you a quick view into how SOS works in the Immediate Window. If you want to get hard-core with debugging, the absolute best places to learn are these two blogs:

- John Robbins, at Wintellect: *http://www.wintellect.com/CS/blogs/jrobbins/default.aspx*
- Tess Ferrandez, an ASP.NET Escalation Engineer at Microsoft: *http://blogs.msdn.com/b/tess*

Loading SOS in the Immediate Window

With that said, let's take a look at what it takes to get SOS going in the Immediate Window when using 32-bit and 64-bit architectures. As we go along, I also want to show you the messages you commonly encounter when trying to set this up. First, open the Immediate Window (Ctrl+Alt+I), type in **.load sos**, and press Enter.

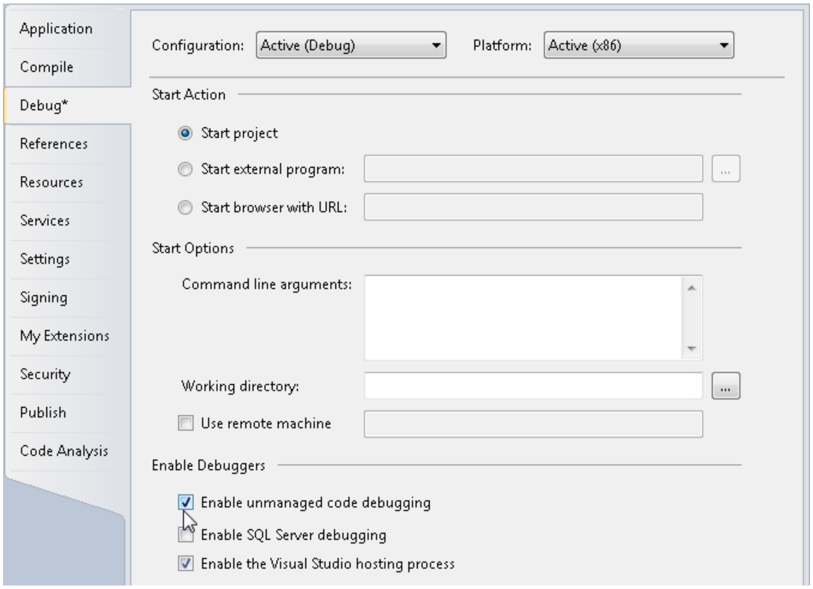
32-bit

You most likely get the following message:

"SOS not available while Managed only debugging. To load SOS, enable unmanaged debugging in your project properties."

To fix this, go to your project properties page and click the Debug tab.

Then select Enable Unmanaged Code Debugging:



Configuration: Active (Debug)
Platform: Active (x86)

Start Action

● Start project
○ Start external program:
○ Start browser with URL:

Start Options

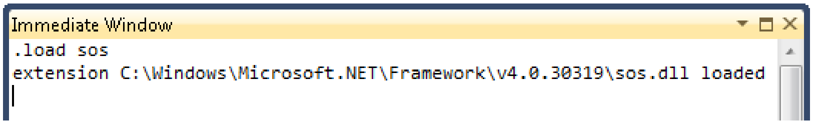
Command line arguments:
Working directory:
☐ Use remote machine

Enable Debuggers

☑ Enable unmanaged code debugging
☐ Enable SQL Server debugging
☑ Enable the Visual Studio hosting process

Now go back to the Immediate Window, and type **.load sos** again. It might take a few seconds, but eventually you should see the following message:

Note Your version might be different based on the CLR being used.



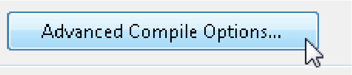
```
.load sos
extension C:\Windows\Microsoft.NET\Framework\v4.0.30319\sos.dll loaded
```

64-bit

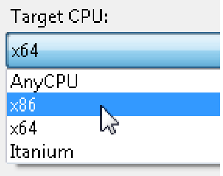
You might see the following message:

"Error during command: extension C:\Windows\Microsoft.NET\Framework64\v4.0.30319\sos.dll could not load (error 193)"

This means that you are attempting to debug an x64 (64-bit) application. Visual Studio currently offers no support for interop (managed/unmanaged) debugging on x64. You can fix this in VB by going to the project properties (Compile tab) and clicking the Advanced Compile Options button:



In C#, you would go to the Build tab of your project properties instead. Then, under Target CPU, choose x86:

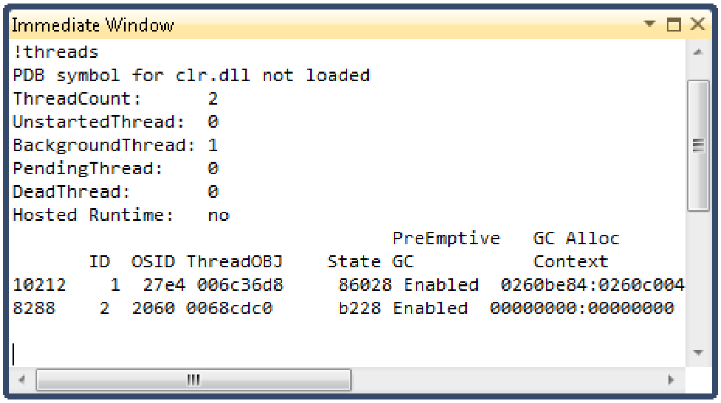


Using SOS

Once you have Son of Strike loaded, there are a variety of WinDbg commands you can leverage. I'll cover some of the more interesting ones in this section.

Threads and symbols

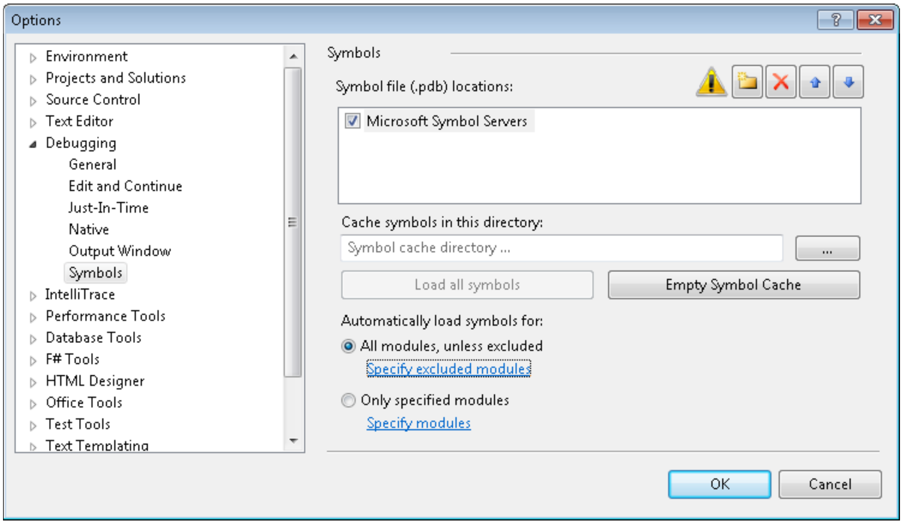
Type **!threads** to see threading info:



```
!threads
PDB symbol for clr.dll not loaded
ThreadCount:      2
UnstartedThread:  0
BackgroundThread: 1
PendingThread:    0
DeadThread:       0
Hosted Runtime:   no
                                   PreEmptive   GC Alloc
       ID  OSID ThreadOBJ   State GC           Context
10212    1  27e4 006c36d8     86028 Enabled  0260be84:0260c004
8288    2  2060  0068cdc0     b228 Enabled  00000000:00000000
```

Do you see the error that says "PDB symbol for clr.dll not loaded"? This is a common error that is trying to tell you that you need to get symbols. The easiest way to do this is to go to Tools | Options | Debugging | Symbols and select the Microsoft Symbol Servers check box in the Symbol File (.pdb) Locations area:

> **Warning** There is definitely a performance hit when loading symbols, so be prepared to have some delays as you debug.



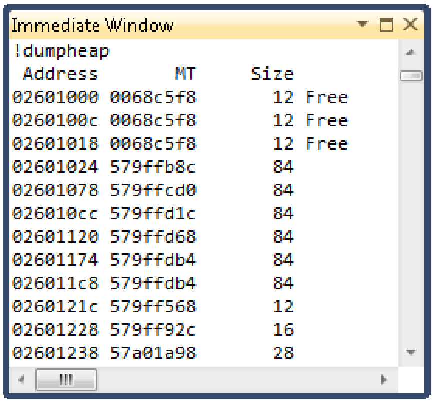
☑ Microsoft Symbol Servers

● All modules, unless excluded

○ Only specified modules

Dump the managed heap

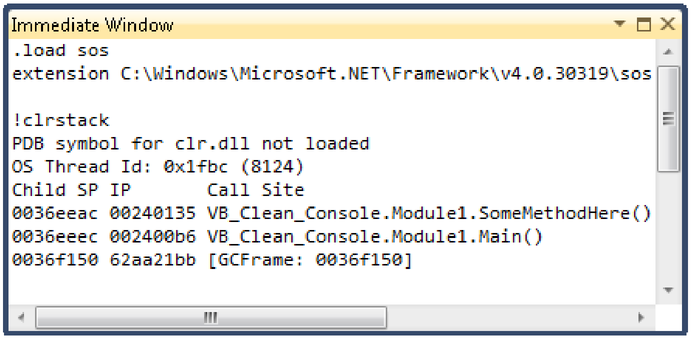
You can dump the managed heap by typing **!dumpheap**. Just watch out—this is pretty verbose output by default:



```
!dumpheap
 Address       MT     Size
02601000 0068c5f8       12 Free
0260100c 0068c5f8       12 Free
02601018 0068c5f8       12 Free
02601024 579ffb8c       84
02601078 579ffcd0       84
026010cc 579ffd1c       84
02601120 579ffd68       84
02601174 579ffdb4       84
026011c8 579ffdb4       84
0260121c 579ff568       12
02601228 579ff92c       16
02601238 57a01a98       28
```

Current thread call stack

If you want to display the call stack for the current thread, you can use **!clrstack**. The following illustration shows a sample of the output:



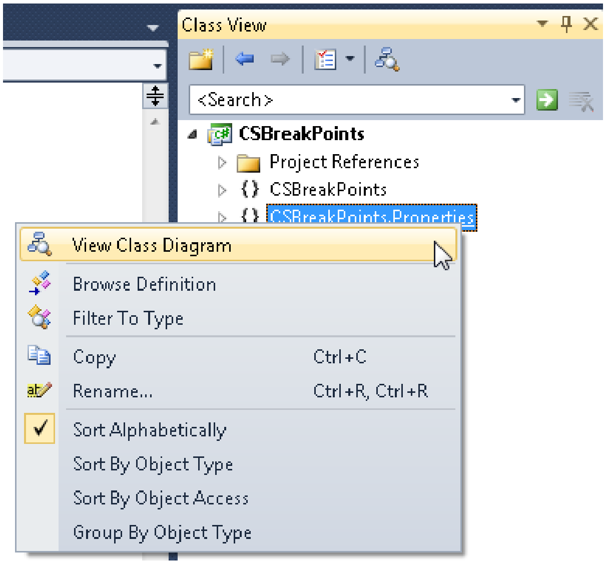
```
.load sos
extension C:\Windows\Microsoft.NET\Framework\v4.0.30319\sos

!clrstack
PDB symbol for clr.dll not loaded
OS Thread Id: 0x1fbc (8124)
Child SP IP       Call Site
0036eeac 00240135 VB_Clean_Console.Module1.SomeMethodHere()
0036eeec 002400b6 VB_Clean_Console.Module1.Main()
0036f150 62aa21bb [GCFrame: 0036f150]
```

These are just a few of the commands you can use. You can get as deep or as shallow into this as you want, but the moral of this story is that you can run WinDbg and SOS commands from the Immediate Window and do some serious debugging from within the IDE.

AX.135 Creating a Class Diagram from Class View

MENU	[Context Menu] | View Class Diagram
COMMAND	ClassViewContextMenus.ClassViewItem. ViewClassDiagram
VERSIONS	2005, 2008, 2010
LANGUAGES	C#, VB
CODE	vstipTool0112

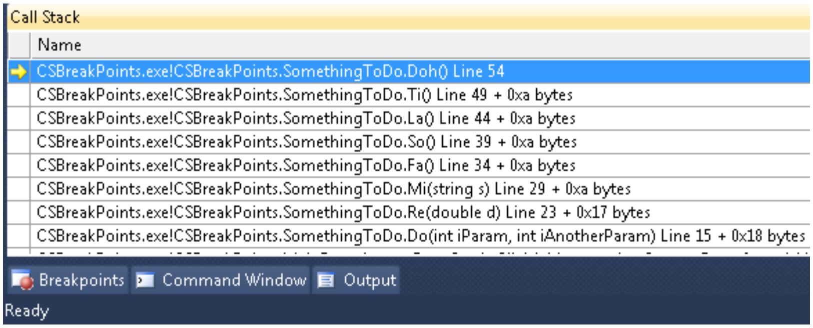
If you like using class diagrams, you can easily create them by using the Class View (Ctrl+Shift+C) window. Just right-click a namespace or class, for example, and choose View Class Diagram:



AX.136 Placing the Call Stack and Call Hierarchy Windows

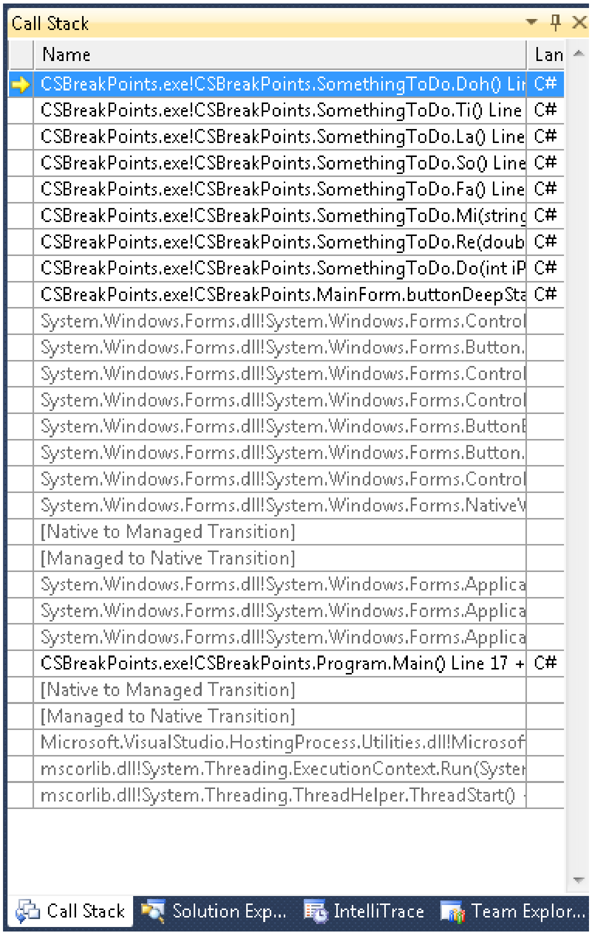
VERSIONS	2005, 2008, 2010
CODE	vstipTool0115

When you are working with the Call Stack or Call Hierarchy windows, they can sometimes get a little lengthy. Usually you see them docked at the bottom. This is a great feature, but not much fun if you want to look at, say, 20 lines in the stack. You might find it useful to dock the window with Solution Explorer (which by default is docked to the right of your screen). Just drag the tab toward Solution Explorer, and place it with the other tabs:



Name
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Doh() Line 54
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Ti() Line 49 + 0xa bytes
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.La() Line 44 + 0xa bytes
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.So() Line 39 + 0xa bytes
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Fa() Line 34 + 0xa bytes
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Mi(string s) Line 29 + 0xa bytes
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Re(double d) Line 23 + 0x17 bytes
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Do(int iParam, int iAnotherParam) Line 15 + 0x18 bytes

Now you are all set, and you can see much more information while you work:



Name	Lan
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Doh() Lir	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Ti() Line	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.La() Line	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.So() Line	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Fa() Line	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Mi(strin	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Re(doub	C#
CSBreakPoints.exe!CSBreakPoints.SomethingToDo.Do(int iP	C#
CSBreakPoints.exe!CSBreakPoints.MainForm.buttonDeepSta	C#
System.Windows.Forms.dll!System.Windows.Forms.Control	
System.Windows.Forms.dll!System.Windows.Forms.Button.	
System.Windows.Forms.dll!System.Windows.Forms.Control	
System.Windows.Forms.dll!System.Windows.Forms.Control	
System.Windows.Forms.dll!System.Windows.Forms.ButtonE	
System.Windows.Forms.dll!System.Windows.Forms.Button.	
System.Windows.Forms.dll!System.Windows.Forms.Control	
System.Windows.Forms.dll!System.Windows.Forms.NativeV	
[Native to Managed Transition]	
[Managed to Native Transition]	
System.Windows.Forms.dll!System.Windows.Forms.Applica	
System.Windows.Forms.dll!System.Windows.Forms.Applica	
System.Windows.Forms.dll!System.Windows.Forms.Applica	
CSBreakPoints.exe!CSBreakPoints.Program.Main() Line 17 +	C#
[Native to Managed Transition]	
[Managed to Native Transition]	
Microsoft.VisualStudio.HostingProcess.Utilities.dll!Microsof	
mscorlib.dll!System.Threading.ExecutionContext.Run(Syster	
mscorlib.dll!System.Threading.ThreadHelper.ThreadStart()	

Also, recall that this has no impact on your Design Mode experience because the window layouts are different. (See vstipEnv0052, “Window Layouts: Design, Debug, and Full Screen,” page 91)

AX.137 Delete All Breakpoints

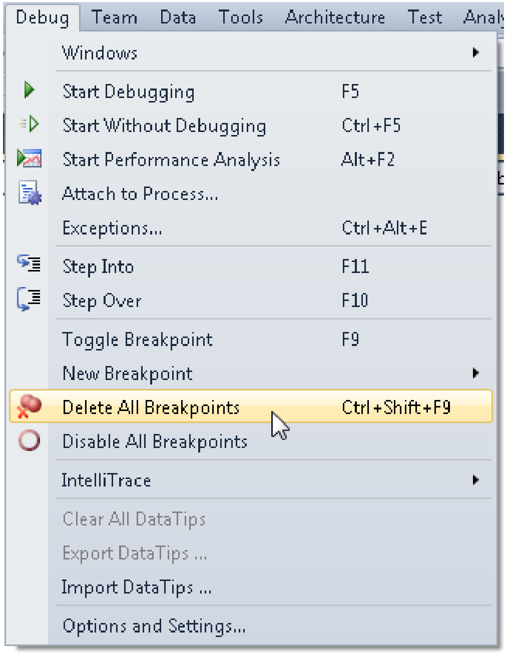
DEFAULT	Ctrl+Shift+F9
VISUAL BASIC 6	Ctrl+Shift+F9
VISUAL C# 2005	Ctrl+Shift+F9
VISUAL C++ 2	[no shortcut]
VISUAL C++ 6	Ctrl+Shift+F9
VISUAL STUDIO 6	Ctrl+Shift+F9
WINDOWS	Alt,D, D
MENU	Debug | Delete All Breakpoints
COMMAND	Debug.DeleteAllBreakpoints
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0025

You can delete all your breakpoints at once. You have a couple of options for doing this.

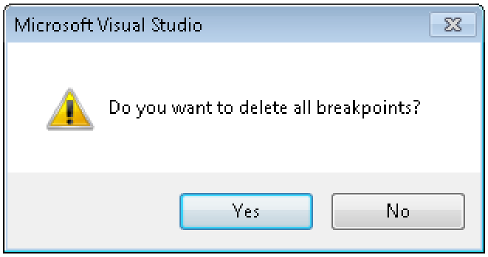
Warning If you want them back, make sure that you export your breakpoints before you use either of these options. See vstipDebug0003, “How to Import and Export Breakpoints” on page 329, for information about how to back up your breakpoints.

Debug | Delete All Breakpoints; Ctrl+Shift+F9

If you use the Debug menu or Ctrl+Shift+F9, the behavior hasn’t changed from previous versions: All breakpoints get deleted:



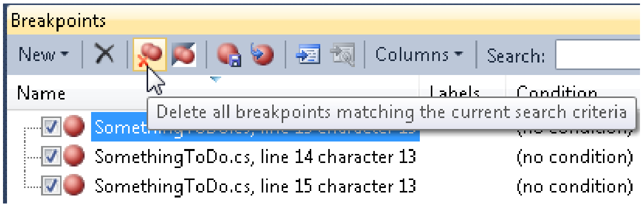
You will get the following dialog box to verify that you want to delete *all* the breakpoints:



Do you want to delete all breakpoints?

Breakpoints Window

In Visual Studio 2010, you can delete all breakpoints that match the current search criteria. Only those breakpoints that are currently visible in the Breakpoints window are deleted. This is a powerful concept that allows you to remove unwanted breakpoints in bulk that are no longer needed, but still keep the ones you will use later:



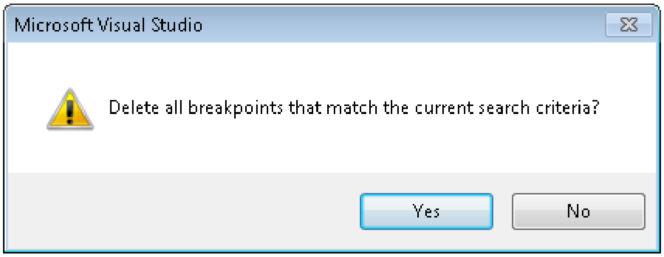
Delete all breakpoints matching the current search criteria

☑ Somet

☑ SomethingToDo.cs, line 14 character 13 (no condition)

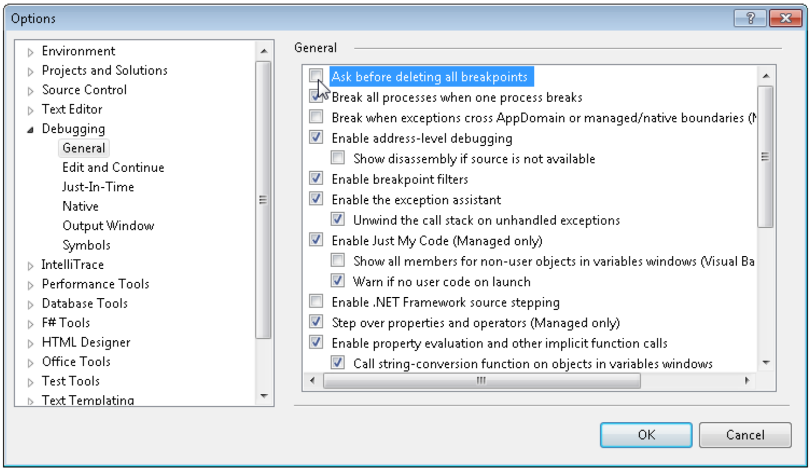
☑ SomethingToDo.cs, line 15 character 13 (no condition)

You get the following dialog box to verify that you want to delete the breakpoints. Notice the additional text concerning the current search criteria:



Delete all breakpoints that match the current search criteria?

You will always get a dialog box to verify that you want to delete all the breakpoints. You can turn this off by going to Tools | Options | Debugging | General and clearing the Ask Before Deleting All Breakpoints check box:



- ☐ Ask before deleting all breakpoints
- ☑ Break all processes when one process breaks
- ☐ Break when exceptions cross AppDomain or managed/native boundaries (
- ☑ Enable address-level debugging
  - ☐ Show disassembly if source is not available
- ☑ Enable breakpoint filters
- ☑ Enable the exception assistant
  - ☑ Unwind the call stack on unhandled exceptions
- ☑ Enable Just My Code (Managed only)
  - ☐ Show all members for non-user objects in variables windows (Visual Ba
  - ☑ Warn if no user code on launch
- ☐ Enable .NET Framework source stepping
- ☑ Step over properties and operators (Managed only)
- ☑ Enable property evaluation and other implicit function calls
  - ☑ Call string-conversion function on objects in variables windows

Warning Just because you can turn it off doesn't mean you should turn it off. I don't recommend it.

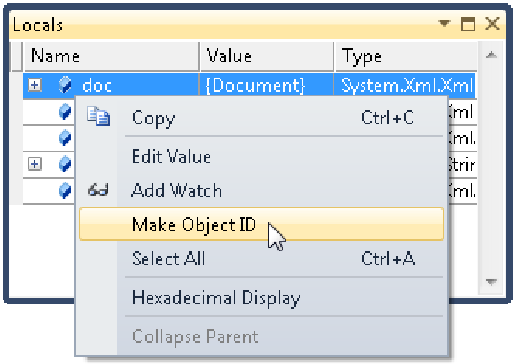
AX.138 Make Object ID

COMMAND	DebuggerContextMenus.AutosWindow.MakeObjectID
VERSIONS	2008, 2010
LANGUAGES	C#, VB
CODE	vstipDebug0015

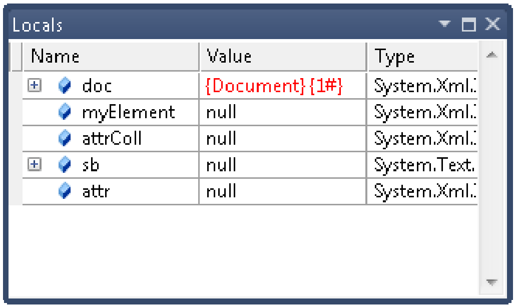
Note The idea for this tip came originally from John Robbins at Wintellect (*http://www.wintellect.com*).

Ever want to track an object even if it is out of scope? How about seeing whether an object has been garbage collected? Well, you can do it with Object IDs. Just follow these steps:

1. Set a breakpoint in your code where you can get to an object variable that is in scope.
2. Run your code and let it stop at the breakpoint.
3. In your Locals or Autos Window, right-click the object variable and choose Make Object ID from the context menu:



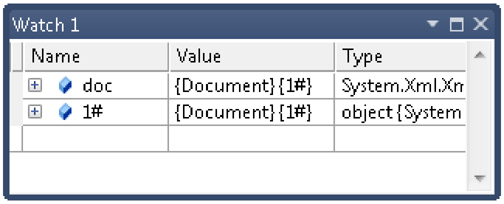
You should now see something new in the Value column:



Name	Value	Type
doc	{Document}{1#}	System.Xml.
myElement	null	System.Xml.
attrColl	null	System.Xml.
sb	null	System.Text.
attr	null	System.Xml.

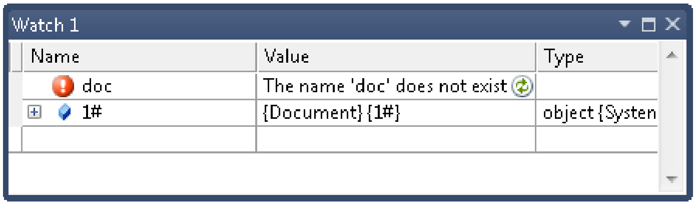
That new value is the Object ID that was generated. Let's see how it works.

To experiment for this experiment, type both the object variable and the new Object ID in your Watch window:



Name	Value	Type
doc	{Document}{1#}	System.Xml.X
1#	{Document}{1#}	object {System

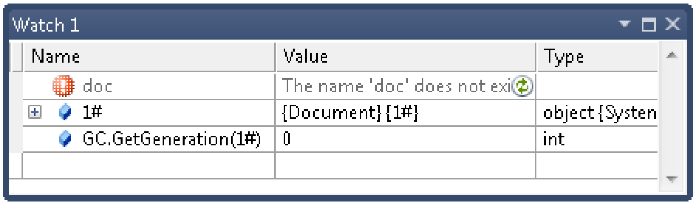
Now if you go to another method where the object variable (in this case "doc") is out of scope, you get the following result:



Name	Value	Type
doc	The name 'doc' does not exist	
1#	{Document}{1#}	object {Syste

Notice that the variable name "doc" is out of scope, but I can still track the Object ID that shows the location in memory that object variable pointed to. This is very handy for looking at objects for full lifecycle.

Also, you can do some interesting things. For example, you can see which generation the memory space is currently in for purposes of garbage collection:



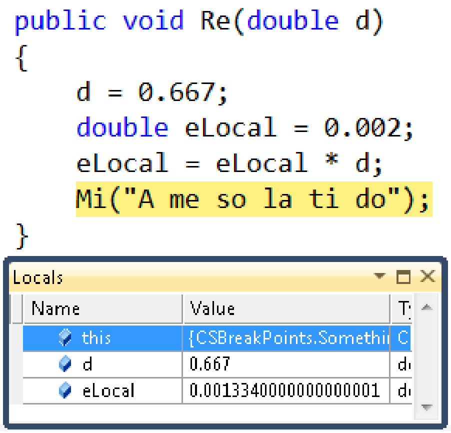
Name	Value	Type
doc	The name 'doc' does not exi	
1#	{Document}{1#}	object{Systen
GC.GetGeneration(1#)	0	int

If you want to learn more about Object IDs, see the excellent blog post at *http://blogs.msdn.com/b/jimgries/archive/2005/11/16/493431.aspx*.

AX.139 Change Values from the Locals Window

DEFAULT	Ctrl+Alt+V, L
VISUAL BASIC 6	Ctrl+Alt+V, L
VISUAL C# 2005	Ctrl+Alt+V, L; Ctrl+D, Ctrl+L; Ctrl+D, L
VISUAL C++ 2	Ctrl+Alt+V, L; Alt+3
VISUAL C++ 6	Ctrl+Alt+V, L; Alt+4
VISUAL STUDIO 6	Ctrl+Alt+V, L; Ctrl+Alt+L
WINDOWS	Alt,D, W, L
MENU	Debug | Windows | Locals
COMMAND	Debug.Locals
VERSIONS	2005, 2008, 2010
CODE	vstipTool0102

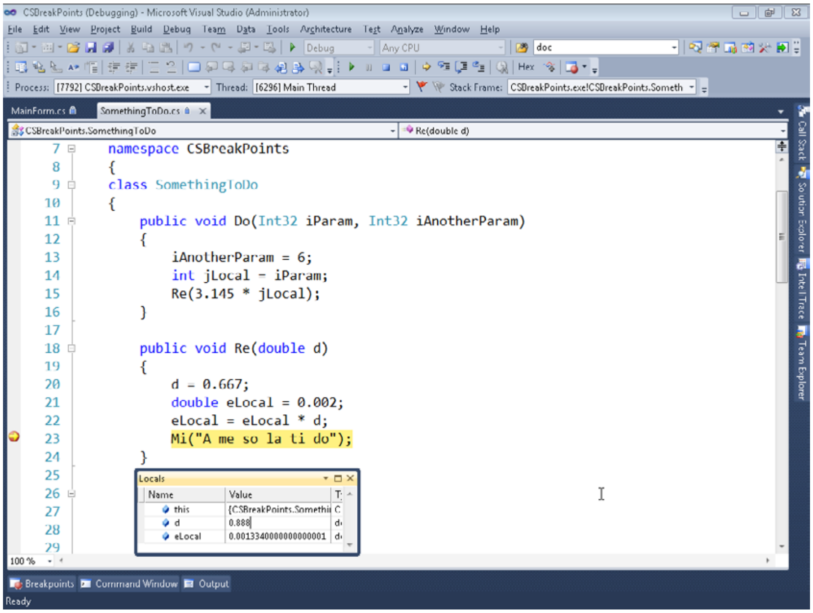
You can use the Locals window to change values while debugging in break mode. Just find any variable you want to change in your Locals window:



```
public void Re(double d)
{
    d = 0.667;
    double eLocal = 0.002;
    eLocal = eLocal * d;
    Mi("A me so la ti do");
}
```

Name	Value	T
this	{CSBreakPoints.Somethi	C
d	0.667	d
eLocal	0.0013340000000000001	d

In this case, let's change "d" to another value:

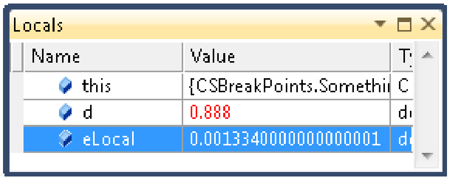

CSBreakPoints (Debugging) - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
Any CPU
doc
Hex
Process: [7792] CSBreakPoints.vshost.exe
Thread: [6296] Main Thread
Stack Frame: CSBreakPoints.exe!CSBreakPoints.Someth
MainForm.cs
SomethingToDo.cs
CSBreakPoints.SomethingToDo
Re(double d)
100 %
Breakpoints
Command Window
Output
Ready
Call Stack
Solution Explorer
IntelliTrace
Team Explorer

```
namespace CSBreakPoints
{
class SomethingToDo
{
    public void Do(Int32 iParam, Int32 iAnotherParam)
    {
        iAnotherParam = 6;
        int jLocal = iParam;
        Re(3.145 * jLocal);
    }

    public void Re(double d)
    {
        d = 0.667;
        double eLocal = 0.002;
        eLocal = eLocal * d;
        Mi("A me so la ti do");
    }
```

Locals

Name	Value	T
this	{CSBreakPoints.Somethi	C
d	0.888	d
eLocal	0.0013340000000000001	d

The changed value turns red afterward to indicate that it has been modified:



Locals

Name	Value	T
this	{CSBreakPoints.Somethi	C
d	0.888	d
eLocal	0.0013340000000000001	d

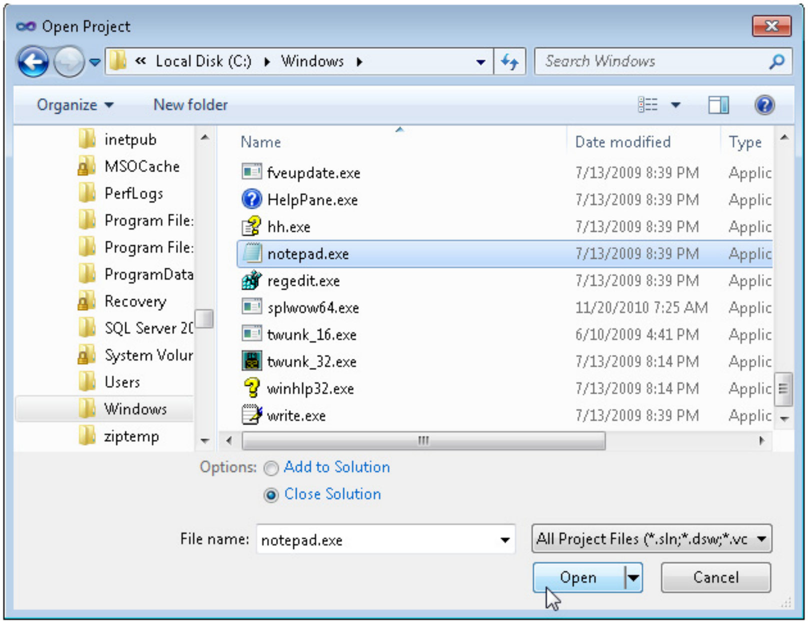
AX.140 Debug Executable Without Using Attach to Process

DEFAULT	Ctrl+Shift+O
VISUAL BASIC 6	Ctrl+Shift+O; Ctrl+O
VISUAL C# 2005	Ctrl+Shift+O
VISUAL C++ 2	Ctrl+Shift+O
VISUAL C++ 6	Ctrl+Shift+O
VISUAL STUDIO 6	Ctrl+O
WINDOWS	Alt,F, O, P; Alt,F, D, N
MENU	File | Open Project/Solution; File | Add | New Project
COMMAND	File.OpenProject; File.AddNewProject
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0034

Note You might need to start Visual Studio with administrator rights before you can use this tip. You will get a warning message that allows you to elevate privileges if this is the case.

You probably already know about the Attach To Process menu items on the Debug and Tools menus, but what if, for example, the process fails before you can attach to it? Maybe it fails on startup, or it runs too fast for you to catch it. Did you know you can create a solution for executables?

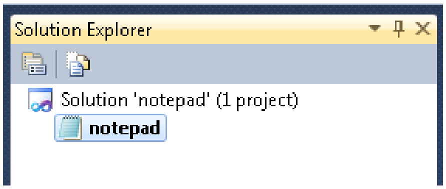
It's easy to do. Just find the executable you want to create a solution for by going to File | Open Project/Solution:



Name	Date modified	Type
fveupdate.exe	7/13/2009 8:39 PM	Applic
HelpPane.exe	7/13/2009 8:39 PM	Applic
hh.exe	7/13/2009 8:39 PM	Applic
notepad.exe	7/13/2009 8:39 PM	Applic
regedit.exe	7/13/2009 8:39 PM	Applic
splwow64.exe	11/20/2010 7:25 AM	Applic
twunk_16.exe	6/10/2009 4:41 PM	Applic
twunk_32.exe	7/13/2009 8:14 PM	Applic
winhlp32.exe	7/13/2009 8:14 PM	Applic
write.exe	7/13/2009 8:39 PM	Applic

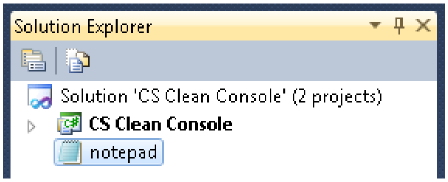
Options: ○ Add to Solution
● Close Solution

File name: notepad.exe
All Project Files (*.sln;*.dsw;*.vc



Solution 'notepad' (1 project)
notepad

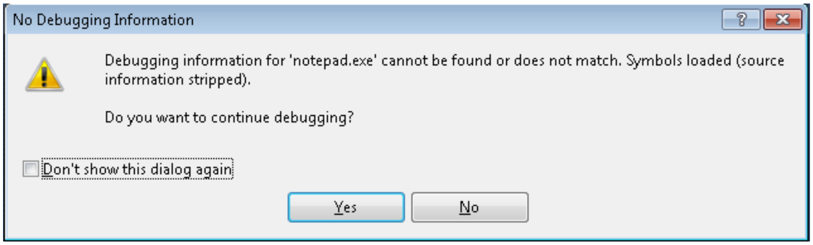
Or, if you have a solution open already, go to File | Add | Existing Project:



Now you can run the executable just like any other project by pressing F5. If you have multiple projects, make sure to set it as the startup.

When you are debugging an executable without the source code, the available debugging features are limited, whether you attach to a running executable or add the executable to a Visual Studio solution.

If the executable was built without debug information in a compatible format, available features are further limited. If you have the source code, the best approach is to import the source code into Visual Studio and create a debug build of the executable in Visual Studio:



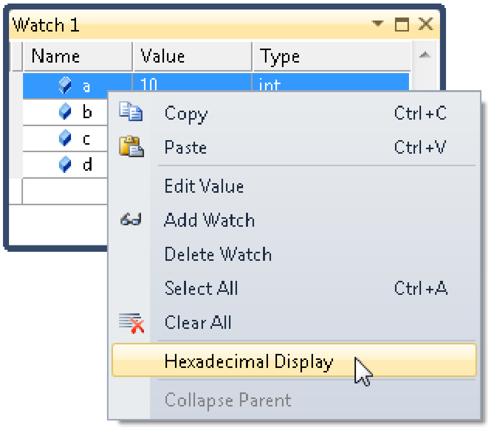
Debugging information for 'notepad.exe' cannot be found or does not match. Symbols loaded (source information stripped).

Do you want to continue debugging?

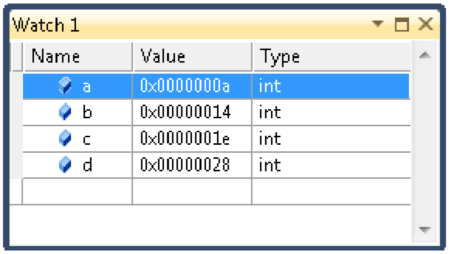
AX.141 The Watch Window: Hexadecimal Display

MENU	[Context Menu] | Hexadecimal Display
COMMAND	Debug.HexadecimalDisplay
VERSIONS	2005, 2008, 2010
CODE	vstipTool0110

All the variable windows (Locals, Autos, QuickWatch, and Watch) support showing the hexadecimal display for values. These values are particularly useful when you are dealing with data that requires any hex input for values. In any variable window, you can right-click anywhere and choose Hexadecimal Display:



You now have hex values displayed for the values in all the variable windows:


Watch 1

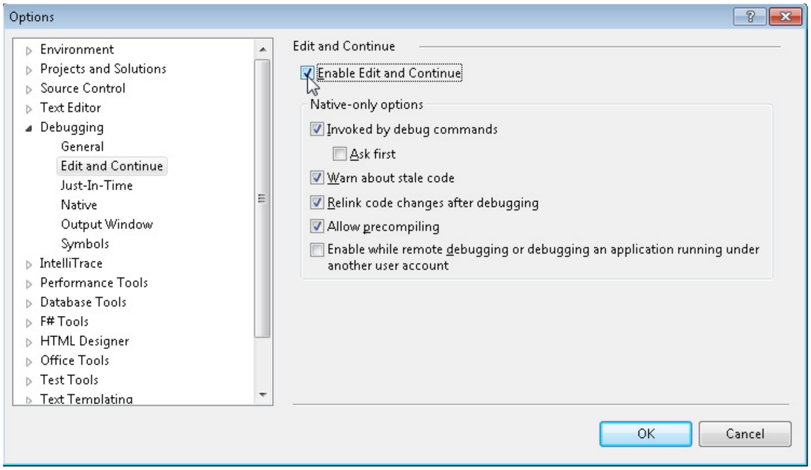
Name	Value	Type
a	0x0000000a	int
b	0x00000014	int
c	0x0000001e	int
d	0x00000028	int

You can repeat this action to turn the hex display off.

AX.142 Edit And Continue

WINDOWS	Alt,T, O
MENU	Tools | Options
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipDebug0038

Did you know you can edit your code while you are debugging? You can do it with the Edit And Continue (EnC) feature. First, you can find this option under Tools | Options | Debugging | Edit And Continue:



Edit and Continue

☑ Enable Edit and Continue

Native-only options

☑ Invoked by debug commands

☐ Ask first

☑ Warn about stale code

☑ Relink code changes after debugging

☑ Allow precompiling

☐ Enable while remote debugging or debugging an application running under another user account

However, in many language-specific scenarios, you can't use this feature. For more detailed information, see the topic "Edit and Continue" at *http://msdn.microsoft.com/en-us/library/bcew296c(v=VS.100).aspx*.

The preceding caveat notwithstanding, this is an interesting feature that allows you to edit code while you are debugging and to continue execution without having to do a full recompile of the code.

Disclaimer

I would be remiss if I didn't mention that there are some people who don't think using Edit And Continue is a good idea. John Robbins, whose opinion I respect a great deal, is one of those people, and he makes some compelling arguments why you might not want to use this feature all the time. You can find John's post at *http://www.wintellect.com/CS/blogs/jrobbins/archive/2004/10/17/c-edit-and-continue-announced.aspx*.

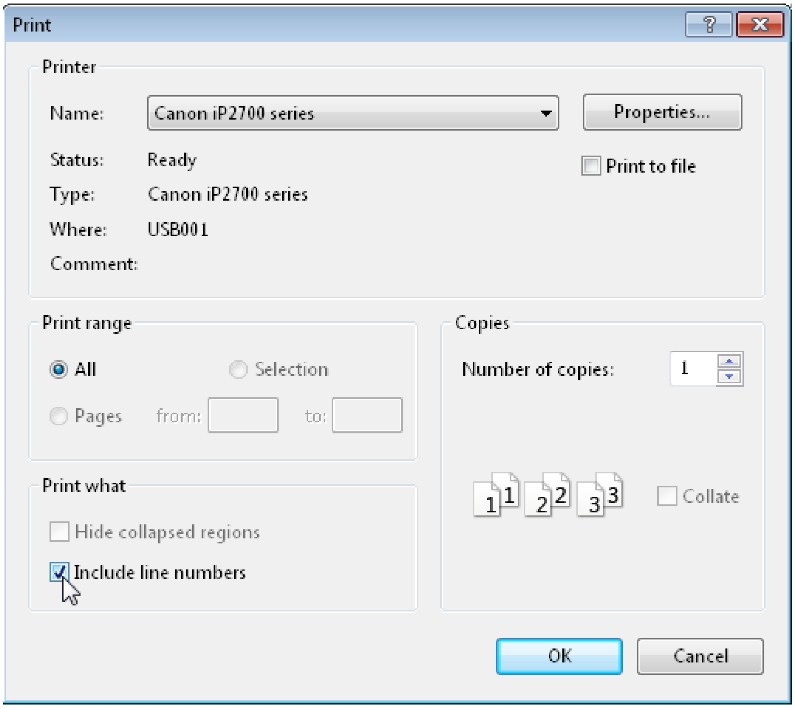
Interestingly, you can find Jeff Atwood's rebuttal to John's argument at *http://www.coding-horror.com/blog/2006/02/revisiting-edit-and-continue.html*.

The bottom line: Make your own informed decision about using Edit And Continue, but even if you do use it, be fully aware of the implications of changes you make when using it.

AX.143 Print with Line Numbers

DEFAULT	Ctrl+P
VISUAL BASIC 6	Ctrl+P
VISUAL C# 2005	Ctrl+P
VISUAL C++ 2	Ctrl+P; Ctrl+Shift+F12; Ctrl+Shift+Alt+F2
VISUAL C++ 6	Ctrl+P
VISUAL STUDIO 6	Ctrl+P
WINDOWS	Alt,F, P
MENU	File | Print
COMMAND	File.Print
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0006

Want to print your line numbers with your code? Don't worry! You can do it by just checking the Include Line Numbers option in the Print dialog box:

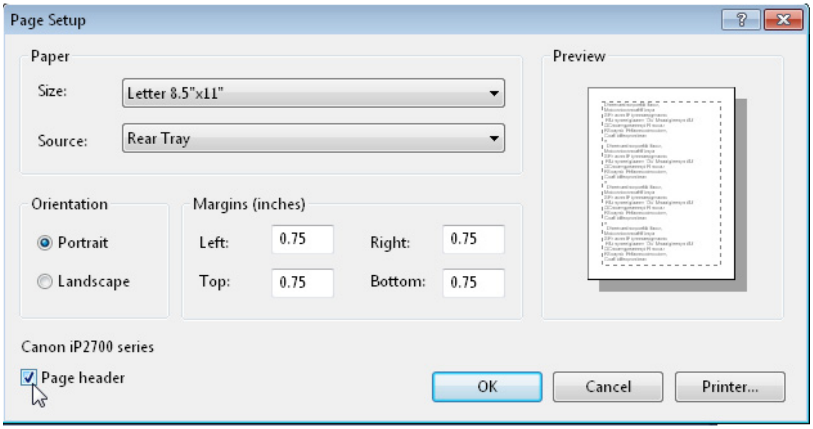


Name: Canon iP2700 series
Status: Ready
Type: Canon iP2700 series
Where: USB001
Comment:
☐ Print to file
● All ○ Selection
○ Pages from: to:
Number of copies: 1
☐ Collate
☐ Hide collapsed regions
☑ Include line numbers

AX.144 Printing the File Path in the Page Header

WINDOWS	Alt,F, U
MENU	File | Page Setup
COMMAND	File.PageSetup
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0008

This feature is on by default in Visual Studio 2010, but just in case you accidentally turn it off or maybe you don't want the file path in the page header, just go to File | Page Setup and notice the Page Header check box:



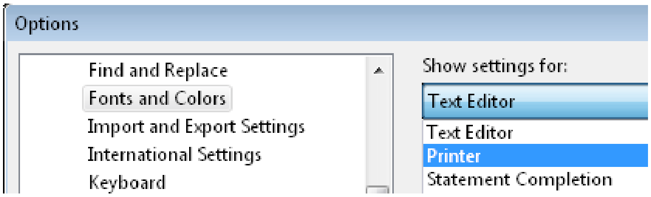
Size: Letter 8.5"x11"
Source: Rear Tray
● Portrait
○ Landscape
Left: 0.75
Right: 0.75
Top: 0.75
Bottom: 0.75
Canon iP2700 series
☑ Page header

The Page Header setting toggles the printing of the file path in the page header.

AX.145 Printing in Different Fonts and Colors

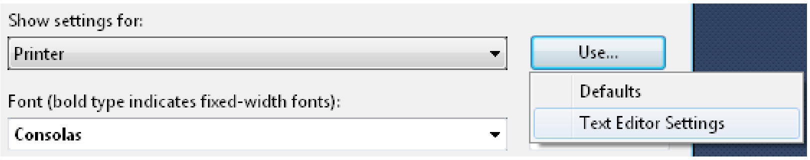
WINDOWS	Alt,T, O
MENU	Tools | Options
COMMAND	Tools.Options
VERSIONS	2005, 2008
CODE	vstipEnv0007

Ever change your fonts or colors in the editor only to be frustrated because the fonts and colors do not print correctly or as expected? Just go to Tools | Options | Environment | Fonts And Colors on your menu bar, and then in the Show Settings For drop-down box, select Printer. Now you can change how your printed output looks:



Show settings for: Text Editor

To use your editor colors when you print, just click Use and select Text Editor Settings:



Show settings for: Printer

Font (bold type indicates fixed-width fonts): Consolas

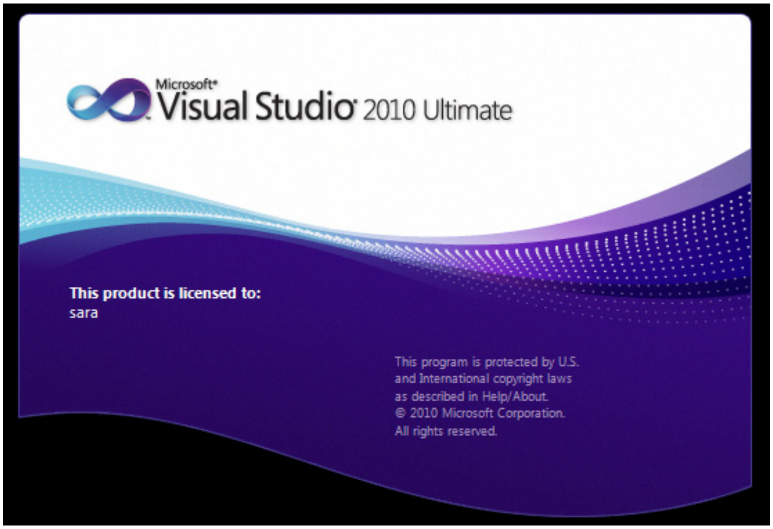
The color and font settings from the text editor will now be applied to your printed output.

Note This is a copy operation, so if you change the text editor settings, you must redo this step to copy the new settings over.

AX.146 Get Rid of the Splash Screen

VERSIONS	2005, 2008, 2010
CODE	vstipEnv0046

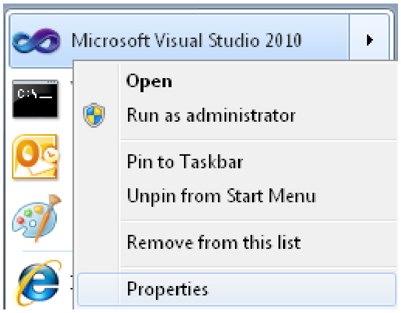
When you start Visual Studio, the splash screen is often the first thing you see:



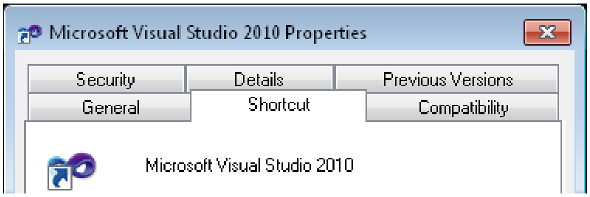
This product is licensed to:
sara

This program is protected by U.S.
and International copyright laws
as described in Help/About.
© 2010 Microsoft Corporation.
All rights reserved.

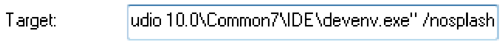
Did you know that you can suppress it? Just go to the properties of your Visual Studio program icon:



Now click the Shortcut tab:


Microsoft Visual Studio 2010 Properties
Security
Details
Previous Versions
General
Shortcut
Compatibility
Microsoft Visual Studio 2010

Add **/nosplash** to the end of the Target area:



Target: udio 10.0\Common7\IDE\devenv.exe" /nosplash

Now when you run Visual Studio, you no longer see the splash screen.

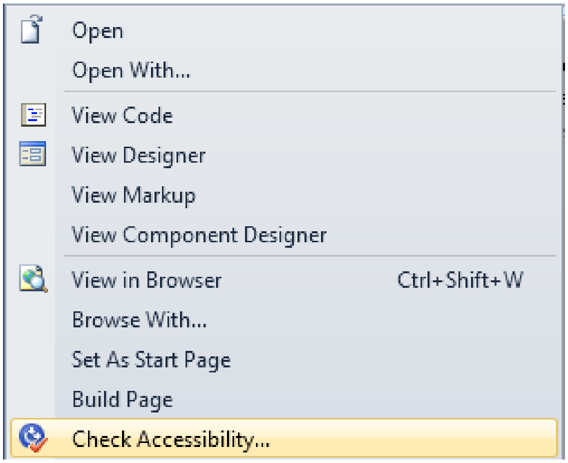
AX.147 Understanding Check Accessibility

WINDOWS	Alt,T, B
MENU	Tools | Check Accessibility; [Context Menu] | Check Accessibility
COMMAND	Tools.CheckAccessibility
VERSIONS	2005, 2008, 2010
CODE	vstipProj0028

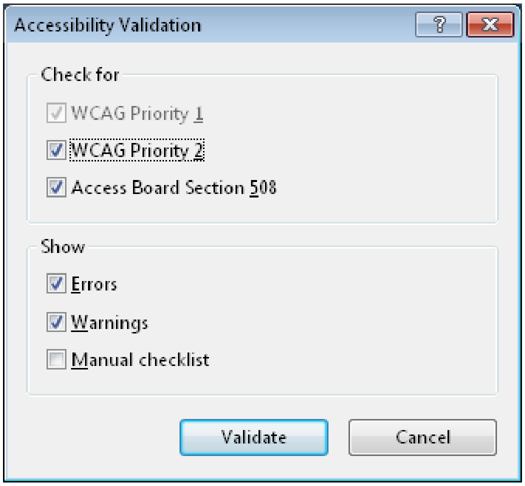
According to the documentation in “Accessibility in Visual Studio and ASP.NET,” at *http://msdn.microsoft.com/en-us/library/ms228004(v=VS.100).aspx*, accessibility is an important consideration in your development work:

“Accessibility standards enable you to build Web pages that can be used by people who have disabilities. [You can] configure ASP.NET Web server controls to make sure that they generate accessible HTML.”

It is always better to make your websites accessible to people with disabilities, and in some cases, it's required. Did you know that you can easily determine whether a page meets accessibility requirements by right-clicking on any page and choosing Check Accessibility?



This opens the Accessibility Validation dialog box:



Check for

☑ WCAG Priority 1
☑ WCAG Priority 2
☑ Access Board Section 508

Show

☑ Errors
☑ Warnings
☐ Manual checklist

Following are descriptions of each option on the Accessibility Validation dialog box.

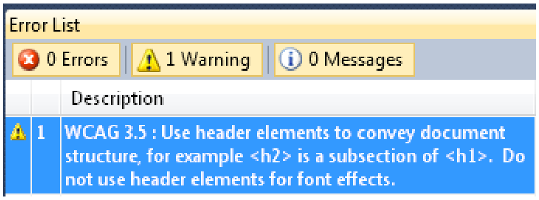
Check For

WCAG Priority 1 & 2—checks for compliance with Web Content Accessibility Guidelines (*http://www.w3.org/WAI/intro/wcag.php*).

Access Board Section 508—checks accessibility by using the standards that were defined by the United States government in Section 508 of the Rehabilitation Act, which are based on the WCAG (*http://www.section508.gov*).

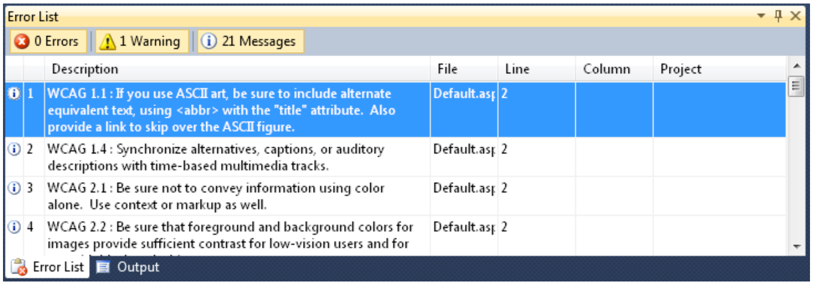
Show

- **Errors and Warnings** Shows relevant items that violate the rules selected in the Check For section:



		Description
⚠	1	WCAG 3.5 : Use header elements to convey document structure, for example <h2> is a subsection of <h1>. Do not use header elements for font effects.

- **Manual Checklist** Generates informational messages that can be used as guides while the errors and warnings are addressed:

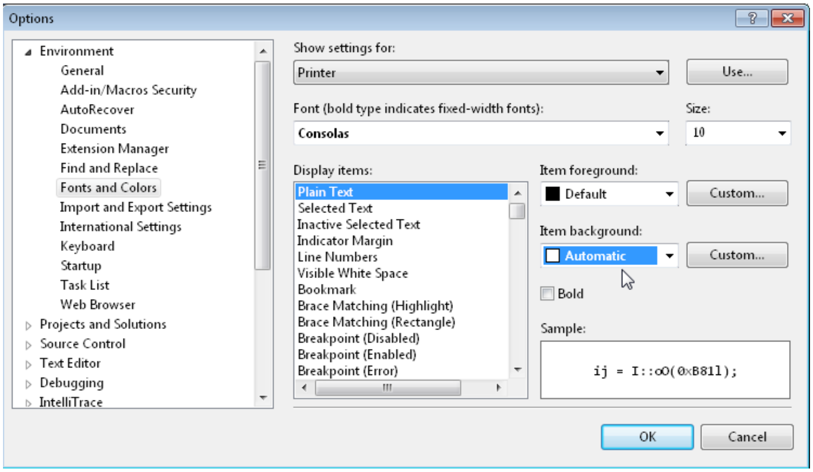


		Description	File	Line	Column	Project
ⓘ	1	WCAG 1.1 : If you use ASCII art, be sure to include alternate equivalent text, using <abbr> with the "title" attribute. Also provide a link to skip over the ASCII figure.	Default.asp	2		
ⓘ	2	WCAG 1.4 : Synchronize alternatives, captions, or auditory descriptions with time-based multimedia tracks.	Default.asp	2		
ⓘ	3	WCAG 2.1 : Be sure not to convey information using color alone. Use context or markup as well.	Default.asp	2		
ⓘ	4	WCAG 2.2 : Be sure that foreground and background colors for images provide sufficient contrast for low-vision users and for	Default.asp	2		

AX.148 Automatic vs. Default in Fonts and Colors

WINDOWS	Alt,T, O
MENU	Tools | Options
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEnv0009

If you've ever wondered what the difference is between the Default and Automatic options in the fonts and colors found at Tools | Options | Environment | Fonts And Colors, see the next illustration:



Show settings for: Printer

Font (bold type indicates fixed-width fonts): Consolas

Size: 10

Item foreground: Default

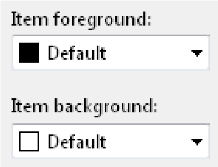
Item background: Automatic

☐ Bold

Sample:

```
ij = I::oO(0xB81l);
```

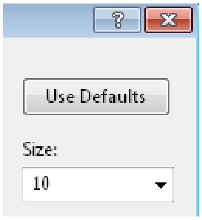
Default



Item foreground: Default

Item background: Default

The Default setting is pretty straightforward. It uses the default colors set up by Visual Studio for the item selected. The colors are what Visual Studio defines as the default. You might call these the "normal" colors, for want of a better term. The good news is that you can always restore these settings by just clicking Use Defaults in the upper-right corner of the dialog box:



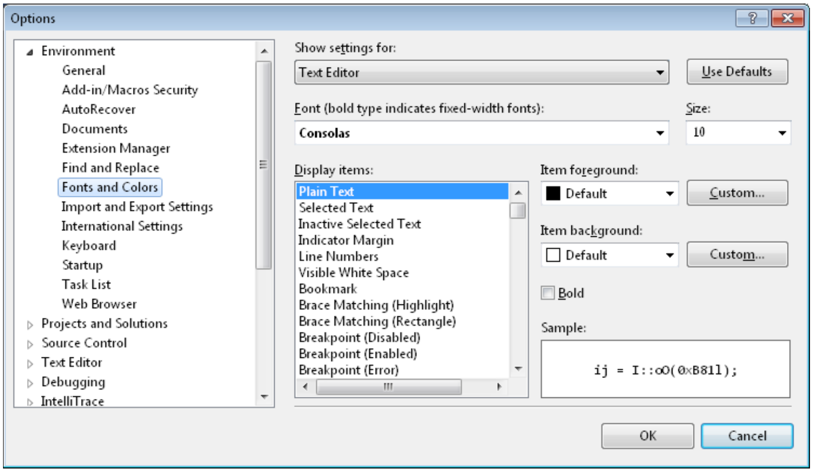
Size: 10

Automatic

The Automatic setting is a lot more interesting. Here is the definition of Automatic from the documentation (the bold emphasis is mine):

> *"Items can **inherit** the [...] color from other display items such as Plain Text. Using this option, when you change the color of an inherited display item, the color of the related display items also change automatically. For example, if you selected the Automatic value for Compiler Error and later changed the color of Plain Text to Red, Compiler Error would also automatically inherit the color Red."*

OK, so let's show you what this means. Following are the current settings for Plain Text:



Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): Consolas

Size: 10

Item foreground: Default

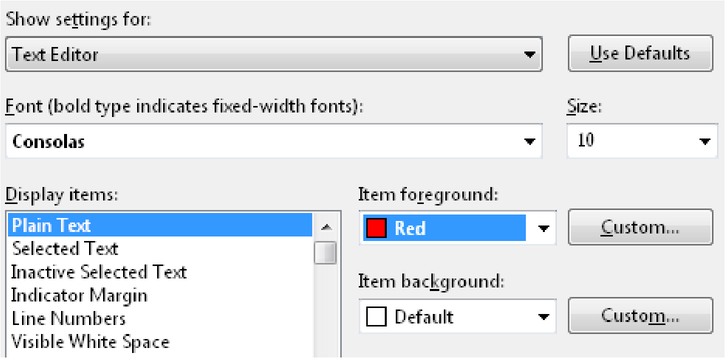
Item background: Default

☐ Bold

Sample:

```
ij = I::oO(0xB81l);
```

Notice, among other things, that the Item foreground is black. Now let's change the foreground color to red:



Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): **Consolas**

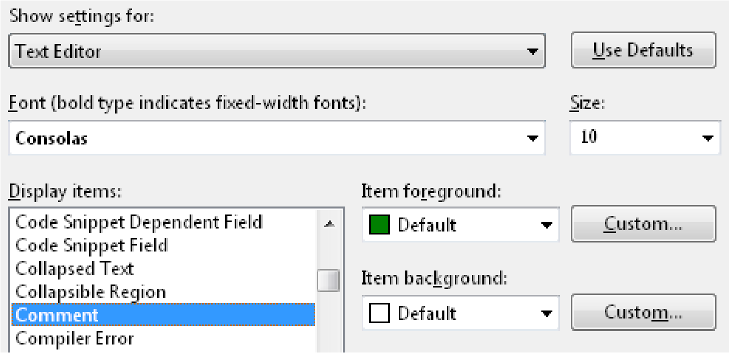
Size: 10

Display items: Plain Text

Item foreground: 🟥 Red

Item background: ☐ Default

Now let's change the display item to Comment and see what the settings are there:



Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): **Consolas**

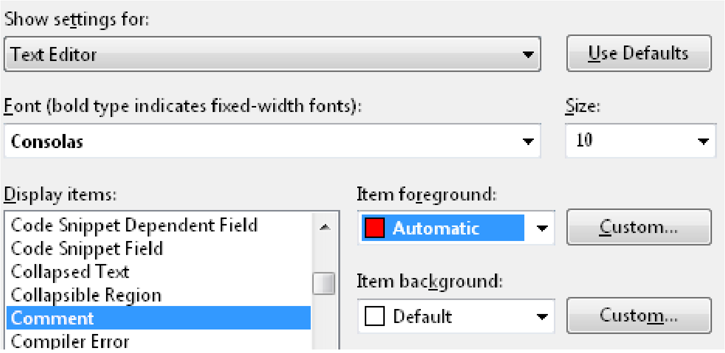
Size: 10

Display items: Comment

Item foreground: 🟩 Default

Item background: ☐ Default

Notice that the Item foreground is green and is set to Default. If we change the Item foreground to Automatic, we get the following:



Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): **Consolas**

Size: 10

Display items: Comment

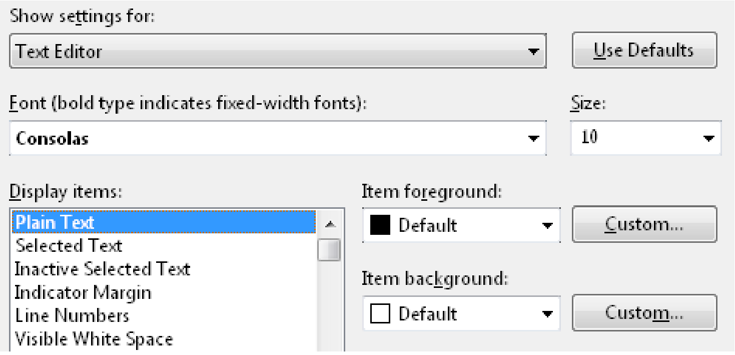
Item foreground: 🟥 Automatic

Item background: ☐ Default

Notice that Comment inherited the Item Foreground setting from the Plain Text setting we just changed to red. So now we clearly see the relationship between the Plain Text setting and some of the other display items. You need to be aware of the following factors:

- Not all items inherit from plain text.
- Plain text inherits from the Windows System.

Let's address the second point. With all the colors back to the default settings, let's look at the Plain Text setting again:


Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): Consolas

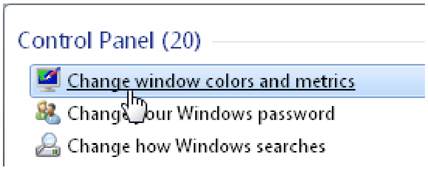
Size: 10

Display items:
Plain Text
Selected Text
Inactive Selected Text
Indicator Margin
Line Numbers
Visible White Space

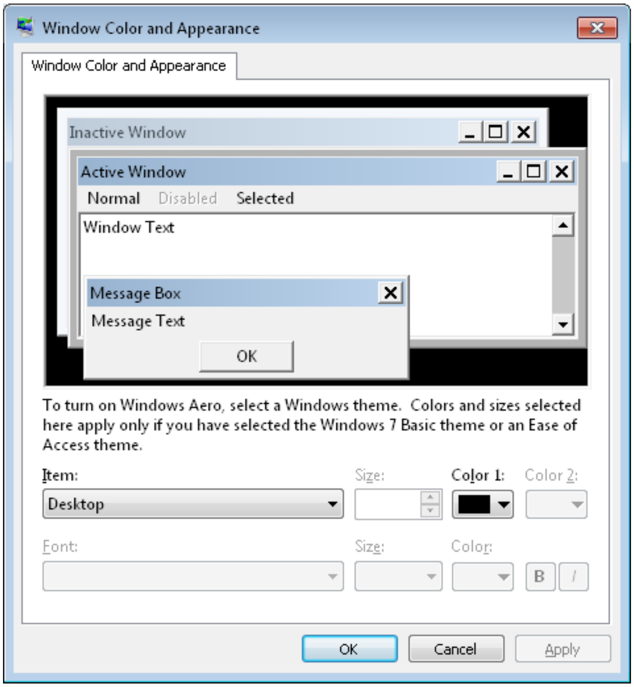
Item foreground: ■ Default

Item background: ☐ Default

Plain Text has a default Item Foreground setting of black again. Where does the plain text default color come from? Let's see what happens if we change the Windows System colors. In this example, I'm using Windows 7, so your settings might be elsewhere, but all versions of Windows have a similar area. In the case of Windows 7, it's called Change Windows Colors And Metrics, as you can see in the following illustration:


Control Panel (20)
Change window colors and metrics
Chang[illegible]our Windows password
Change how Windows searches

When I click this option, it brings me to the Window Color And Appearance dialog box:



To turn on Windows Aero, select a Windows theme. Colors and sizes selected here apply only if you have selected the Windows 7 Basic theme or an Ease of Access theme.

Item: Desktop

Size:

Color 1: ■

Color 2:

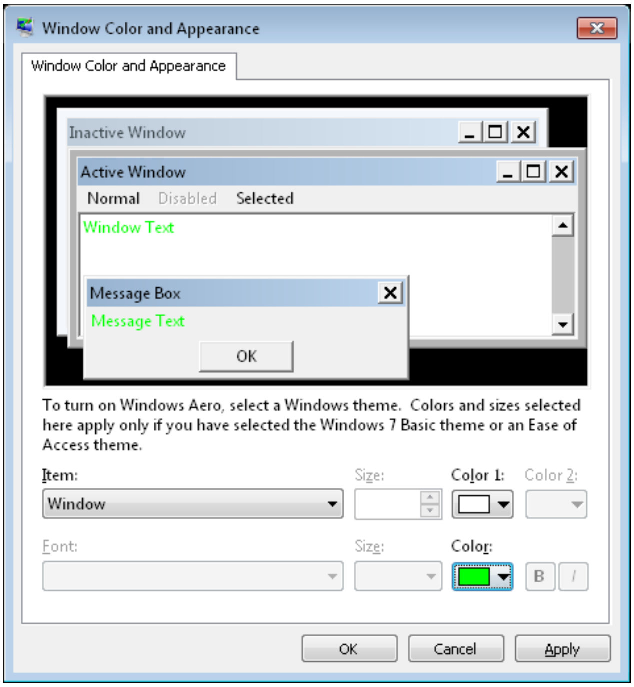
Font:

Size:

Color:

B *I*

Now I change the Item to Window and the font color to lime green:



To turn on Windows Aero, select a Windows theme. Colors and sizes selected here apply only if you have selected the Windows 7 Basic theme or an Ease of Access theme.

Item: Window

Size:

Color 1: □

Color 2:

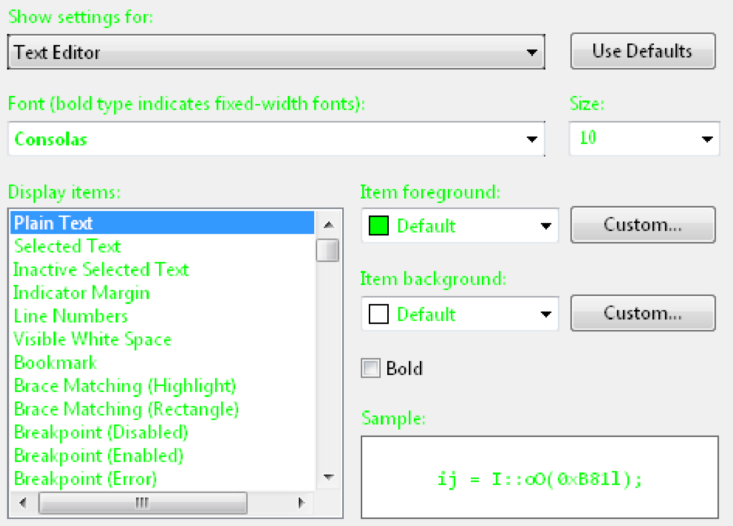
Font:

Size:

Color: 🟩

B *I*

I click Apply and then switch back to Visual Studio and notice that, among other things, the Plain Text setting now has a default color of lime green:



Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): Consolas

Size: 10

Display items: Plain Text

Item foreground: 🟩 Default

Item background: ☐ Default

☐ Bold

Sample:

```
ij = I::oO(0xB81l);
```

So the plain text default color clearly inherits from the Windows System, and now various display items inside Visual Studio can inherit from the Plain Text setting by setting their color to Automatic. Now we finally have a clear picture of the difference between Default and Automatic in the Fonts And Colors dialog box.

At this point, you might want to change all your colors back to what they were before we started this adventure.

AX.149 Visual Studio Permissions Needed on Windows Vista or Later

VERSIONS	2005, 2008, 2010
CODE	vstipEnv0056

A popular misconception is that you need to have Administrator privileges to use Visual Studio. While this is true in some cases, it isn't true in all of them. So when do you need to run Visual Studio as an administrator and when don't you? This tip offers some guidance.

Installing Visual Studio (All Versions)

You need Administrator rights to install Visual Studio.

Running Visual Studio 2005

To run Visual Studio 2005 on Windows Vista or later, you are prompted to run as Administrator when you start the application. This version of Visual Studio requires that you have Administrator rights to use it.

Specific Scenarios for Visual Studio 2008/2010

Web/Internet Information Services

Creating a new local or remote IIS website project—You cannot make changes to the Internet Information Services (IIS) metabase (for example, creating new entries) because it requires administrative privileges. This affects your ability to configure some settings in the Web.config file.

Opening a local or remote IIS website project—You cannot run your website unless you use the ASP.NET Development Server, which is the default web server for filesystem websites. Alternatively, you can set project options to open the browser and point to the website by using IIS.

Debugging a local or remote IIS website project—You cannot attach to a process that is running under the IIS worker process because it requires administrative privileges.

More information can be found here: *http://msdn.microsoft.com/en-us/library/ms178112(v=VS.90).aspx*.

Windows Installer deployment

Windows Installer technology supports software installation on the Windows Vista (or later) operating system. The end user installing applications on Windows Vista should receive prompts only for each component installation that requires elevation, even when the user's computer runs under User Account Control (UAC).

More information can be found here: *http://msdn.microsoft.com/en-us/library/Bb384154(v=VS.100).aspx*.

ClickOnce deployment

Windows Installer deployment requires administrative permissions and allows only limited user installation; ClickOnce deployment enables non-administrative users to install and grants only those Code Access Security permissions necessary for the application.

More information can be found here: *http://msdn.microsoft.com/en-us/library/t71a733d.aspx*.

Code

Some code requires Administrator access to execute. If possible, alternatives to this code should be pursued. Examples of code operations that require Administrator access are as follows:

- Writing to protected areas of the filesystem, such as the Windows or Program Files directories
- Writing to protected areas of the registry, such as HKEY_LOCAL_MACHINE
- Installing assemblies in the Global Assembly Cache (GAC)

Generally, these actions should be limited to application installation programs. This allows users to use administrator status only temporarily.

More information can be found here: *http://msdn.microsoft.com/en-us/library/ms173360(v=VS.100).aspx*.

Debugging

According to the documentation, "you can debug any applications that you launch within Visual Studio (native and unmanaged) as a non-administrator by becoming part of the Debugging Group. This includes the ability to attach to a running application using the Attach to Process command. However, it is necessary to be part of the Administrator Group in order to debug native or managed applications that were launched by a different user."

More information can be found here: *http://msdn.microsoft.com/en-us/library/ms173360(v=VS.100).aspx*.

COM/COM Interop

Classic COM

When you add a classic COM control, such as an .ocx control, to the toolbox, Visual Studio tries to register the control. You must have administrator credentials to register the control.

Add-ins written by using classic COM must be registered to work in Visual Studio. You must have administrator credentials to register the control.

COM Interop

When you build managed components and you have selected Register For COM Interop, the managed assemblies must be registered. You must have administrator credentials to register the assemblies.

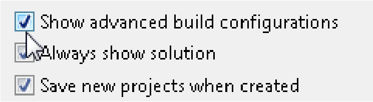
More information can be found here: *http://msdn.microsoft.com/en-us/library/ms165100(v=VS.90).aspx*.

AX.150 Show Advanced Build Configurations

WINDOWS	Alt,T, O
MENU	Tools | Options | Projects and Solutions | General
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipProj0016

This one is interesting, and the main reason I'm showing it to you is so that you know how to turn this back on if it ever gets turned off.

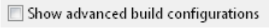
If you go to Tools | Options | Projects And Solutions | General, you can see many available options. Locate the Show Advanced Build Configurations option:



☑ Show advanced build configurations
☑ Always show solution
☑ Save new projects when created

Simplified Build Configurations

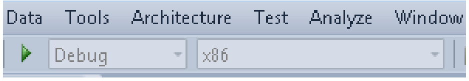
So what does this option do? Well, when it is turned off, it uses simplified build configurations, which involve several changes:



☐ Show advanced build configurations

Configuration Manager

In simplified builds, the Configuration Manager is not available. It's just disabled (or removed completely) from all areas:



The practical implications of this are that you can't do any custom build configurations, which makes sense with a simplified build configuration option. So when you don't see the Configuration Manager anymore or it's disabled, that is usually a clear sign that you have turned off Show Advanced Build Configurations.

Debug Build

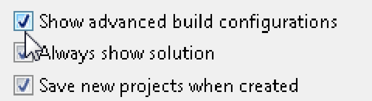
In a simplified build scenario, each time you press F5 or go to Debug | Start Debugging, a debug build of your application is created. This is the expected behavior.

Release Build

The real shocker is that you have no obvious way to create a release build. With simplified builds, you create a release build by going to Build | Build Solution (Ctrl+Shift+B).

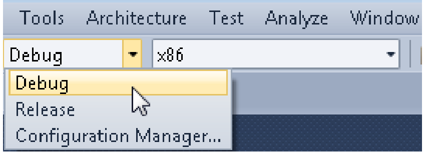
Advanced Build Configurations

I could do an entire series on advanced builds, but I'll just keep it to the basics for now. Essentially, you first select Show Advanced Build Configurations:



☑ Show advanced build configurations
☑ Always show solution
☑ Save new projects when created

This option gives you access to the Configuration Manager, so you can actively switch between build types without having to remember some arcane steps to do it:



As an added bonus, it gives you the ability to make custom builds with your own special configuration options specified for the build.

In short, you have many good reasons to show the advanced build configurations options and few good reasons to turn it off.

AX.151 Emacs Emulation

WINDOWS	Alt,T, O
MENU	Tools | Options | Environment | Keyboard
COMMAND	Tools.Options
VERSIONS	2005, 2008, 2010
CODE	vstipEdit0079

For those not familiar with the term, according to Wikipedia (*http://en.wikipedia.org/wiki/Emacs*), Emacs can be defined as follows:

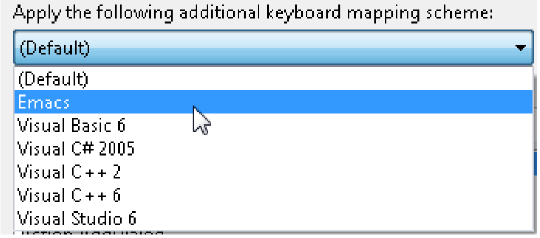
> *"[...] a class of feature-rich text editors, usually characterized by their extensibility. Emacs has, perhaps, more editing commands than other editors, numbering over 1,000 commands. It also allows the user to combine these commands into macros to automate work.*

"Development began in the mid-1970s and continues actively as of 2010. Emacs text editors are most popular with technically proficient computer users and computer programmers. The most popular version of Emacs is GNU Emacs, a part of the GNU project, which is commonly referred to simply as 'Emacs'."

Did you know that Visual Studio supports Emacs emulation? If you are using Visual Studio 2010, you need to install the Emacs extension that can be found here:

http://visualstudiogallery.msdn.microsoft.com/en-us/09dc58c4-6f47-413a-9176-742be7463f92

Then, for Visual Studio 2008, 2005, and 2010, go to Tools | Options | Environment | Keyboard and choose Emacs from the Apply The Following Additional Keyboard Mapping Scheme drop-down list:



Apply the following additional keyboard mapping scheme: (Default)

AX.152 ViM Emulation

VERSIONS	2005, 2008, 2010
CODE	vstipEdit0080

Many people like to use ViM—there is no denying it. Unfortunately, Visual Studio does not support ViM emulation out of the box. However, as of the time of this writing, you have two solutions in the Visual Studio Gallery, depending on your version.

Visual Studio 2010

VsVim by Jared Parsons is very well reviewed, plus it is free. You can find it at *http://visualstudiogallery.msdn.microsoft.com/en-us/59ca71b3-a4a3-46ca-8fe1-0e90e3f79329*.

Visual Studio 2008 and Prior

ViEmu by NGEDIT Software is available for Visual Studio versions prior to Visual Studio 2010. It has good reviews, and the trial version is available from the gallery. It is available at *http://visualstudiogallery.msdn.microsoft.com/en-us/C9055830-39AB-4B39-A19E-4D60F195E7FC*.

So, if you need ViM emulation, you have a couple options. These might not be the only options, but they are readily available in the Visual Studio Gallery.