

# Visual Studio.NET Code Editing Tips and Tricks

## 25 Ways to Improve Productivity

Venkat Subramaniam  
venkats@durasoftcorp.com

Those who know me also know that I love telling stories! So, here is one to start this. This happened several years ago. I was working for a large engineering company and I got called by the administrative assistant of the department to help her with some problem on her system. She had difficulty printing a proposal. I started using a word processor on her system when she slapped on my wrist and said don't do that (those who are familiar of this person would say that that was very mild of her – generally she was much more violent!). Puzzled, I said "what?" She said "do not use the mouse if you can do it with the keyboard. You are not productive if you keep running to the mouse for simple things." When I got back to my desk, it got me thinking, "I spend significant part of my time awake on the system. What she said makes absolute sense." I started to look at common tasks that I do each day; things that I do several times a day and started looking for shortcuts to these operations. When ever I start using a tool extensively, I start looking for short cuts or ways to improve productivity. When I find some one using a tool not to its potential, I share this story, especially with those who take my class.

In this article we present 25 things you can do in the VS.NET to improve your productivity. Go ahead open your copy of VS.NET and practice these. That will help you remember these longer than merely reading.

### 1. Comment – uncomment block of code.

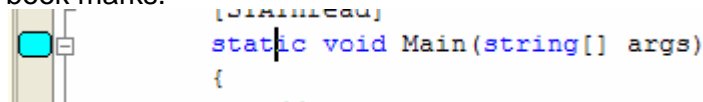
Select the block of code, press **Ctrl+K Ctrl+C** to comment.

To uncomment, press **Ctrl+K Ctrl+U**.

### 2. Switch between code editor windows.

**Ctrl+Tab**

### 3. Use book marks.

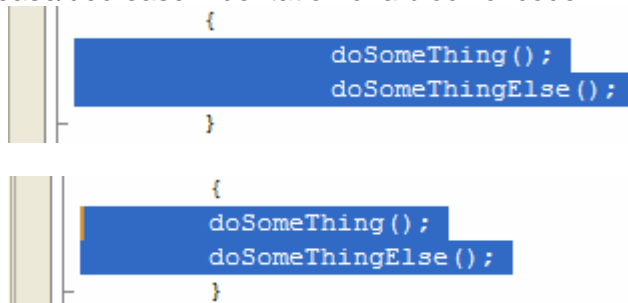


To place (or remove) a book mark, press **Ctrl+K Ctrl+K**.

To go to next book mark, press **Ctrl+K Ctrl+N**.

To go to previous book mark, press **Ctrl+K Ctrl+P**.

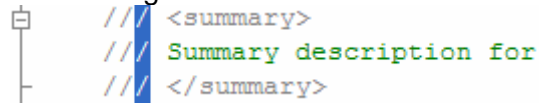
### 4. Increase/decrease Indentation of a block of code.



Select the block of code, press **Tab** (or **Shift-Tab**) to alter indentation.

To select a block of code, press **Shift+down arrow**.

5. Select a rectangular area of text.

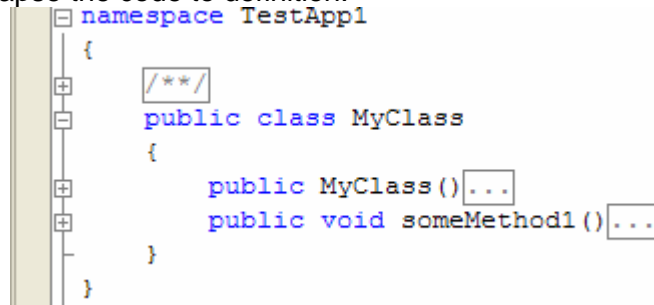


```
/// <summary>
/// Summary description for
/// </summary>
```

The image shows a code editor with a vertical selection bar on the left side. The selection covers three lines of code: a summary tag opening, a summary description, and the summary tag closing. The text is highlighted in blue.

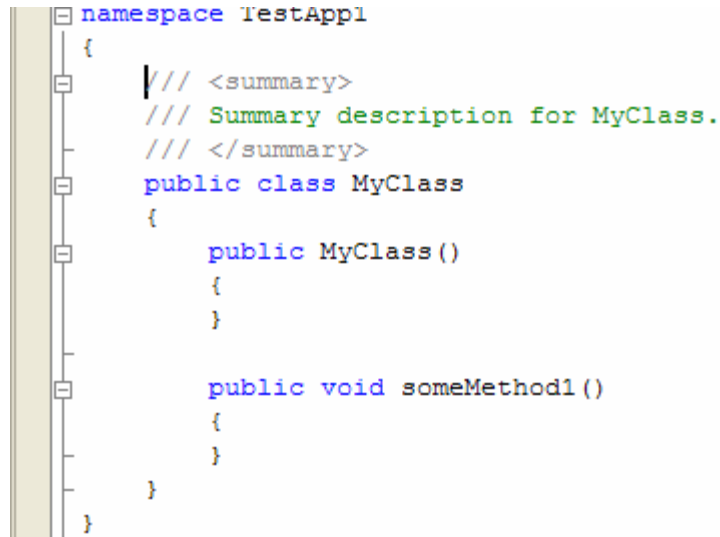
Hold **Alt** down and drag mouse over text area you like to select.

6. Collapse the code to definition.



```
namespace TestApp1
{
    /**/
    public class MyClass
    {
        public MyClass ()...
        public void someMethod1 ()...
    }
}
```

The image shows a code editor with a vertical selection bar on the left side. The code is expanded to definition, showing the full class and method definitions. The code is color-coded: namespace (blue), class (blue), and methods (blue). Ellipses (...) are used to indicate that the code is truncated for brevity.



```
namespace TestApp1
{
    /// <summary>
    /// Summary description for MyClass.
    /// </summary>
    public class MyClass
    {
        public MyClass ()
        {
        }

        public void someMethod1 ()
        {
        }
    }
}
```

The image shows the same code editor as above, but the code is now collapsed to definition. The summary block is shown as a single line of text, and the class and method definitions are shown as single lines of text. The code is color-coded: namespace (blue), class (blue), and methods (blue). Ellipses (...) are used to indicate that the code is truncated for brevity.

To collapse the code to definition, press **Ctrl+M Ctrl+O**. To expand, press **Ctrl+M Ctrl+L**.

7. Navigate through all the code you have visited recently in your edit session

To see what all code you visited or edited in the current edit session, press **Ctrl+hyphen**.

To navigate forward, press **Ctrl+Shift+hyphen**.

8. Search current selected text though out the document.

If you want to search for a selected text in the rest of the document, press **Ctrl+F3**. To continue to search, press **F3**.

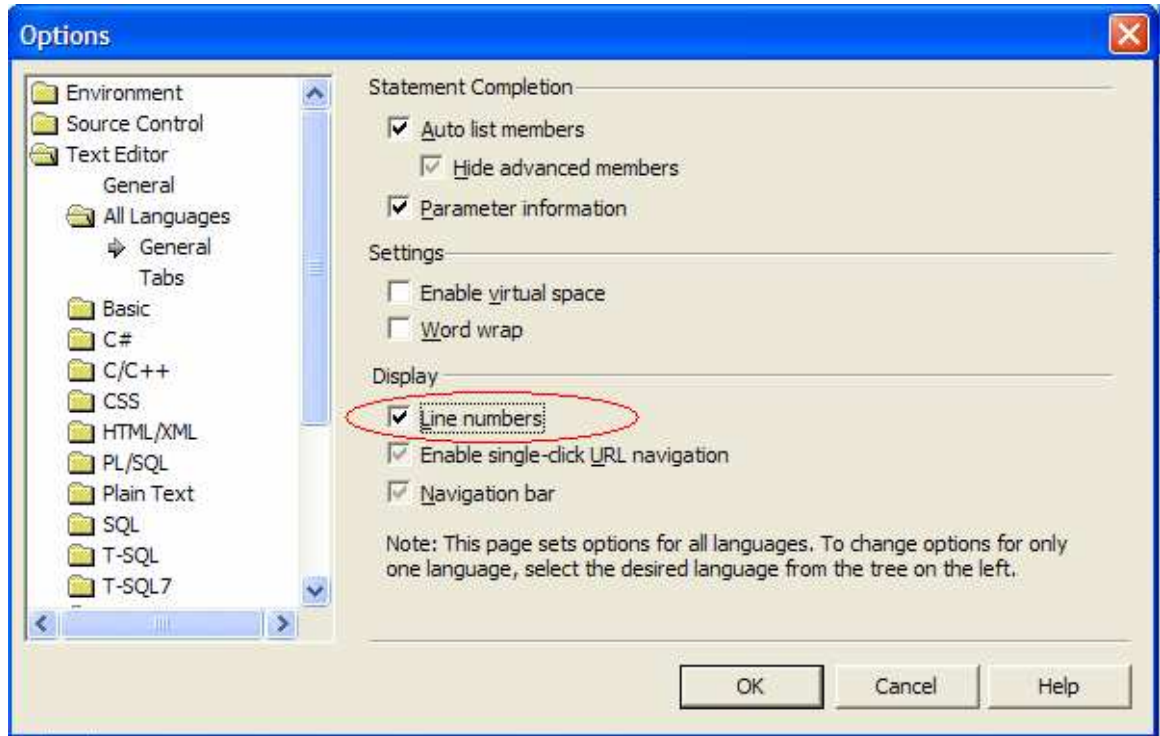
9. Incremental search for arbitrary text.

If you want to start searching for a text incrementally as you type, press **Ctrl+i** and start typing the text you like to search. You can hit backspace to correct

errors in your typing during the search process! You can stop the search by hitting return.

10. Use line numbering on your source code editor.

Using line numbers largely improves productivity, especially when four eyes are looking at a code and discussing it. Turn on line numbering by going to the menu Tools | Options. Look for Text Editor | All Languages | General and check the "Line numbers" checkbox.



11. Use Word Wrap to see the hidden text.

```
10  /// <summary>
11  /// The main entry point for the appli
12  /// </summary>
```

```
10  /// <summary>
11  /// The main entry point for the
    application.
12  /// </summary>
```

If the text you want to see is hidden on the right, you may scroll right to view the text. Or you could use the word wrap to see it. To toggle word wrap, press **Ctrl+R**.

12. Find matching braces.

If you want to find a matching brace for a curly bracket ({}), or the parenthesis (()), place cursor next to it and press **Ctrl+J**.

13. Use automatic word completion.

If your field or method name is fairly long, then you can use the auto word complete feature to avoid typing it. Type a few characters and press **Ctrl+space**. It will either be completed automatically or you will be given a list to select from. You can press Ctrl+space any time to bring up the IntelliSense list members (you may also press Ctrl+J).

14. Invoke the parameter info.

```

public MyClass ()
{
    myQueue = new Queue (|
}

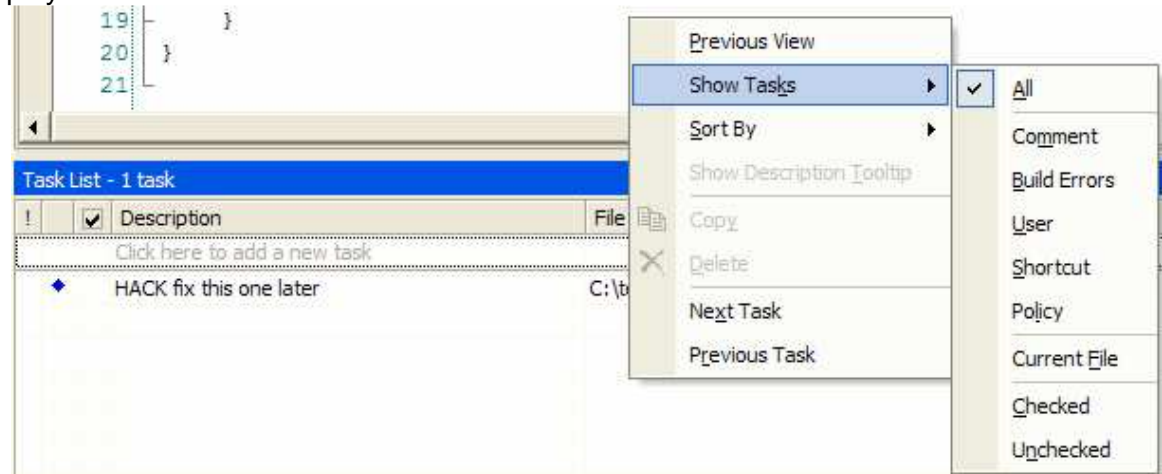
public MyClass ()
{
    myQueue = new Queue (|
}

```

▲ 1 of 4 Queue.Queue (System.Collections.ICollection col)  
 col: The System.Collections.ICollection to copy elements from.

At times when you are writing a call to a method that takes a few arguments, you have to switch back and forth between windows or code. The IntelliSense disappears at this point. You can quickly invoke the parameter info by pressing **Ctrl+Shift+space**.

15. Display Annotations in the Task List.



If Task List is not visible, click on View | Other windows | Task List. Right click on the Task List window and select All to display the tasks in addition to the errors.

16. Create Task List shortcut.

```

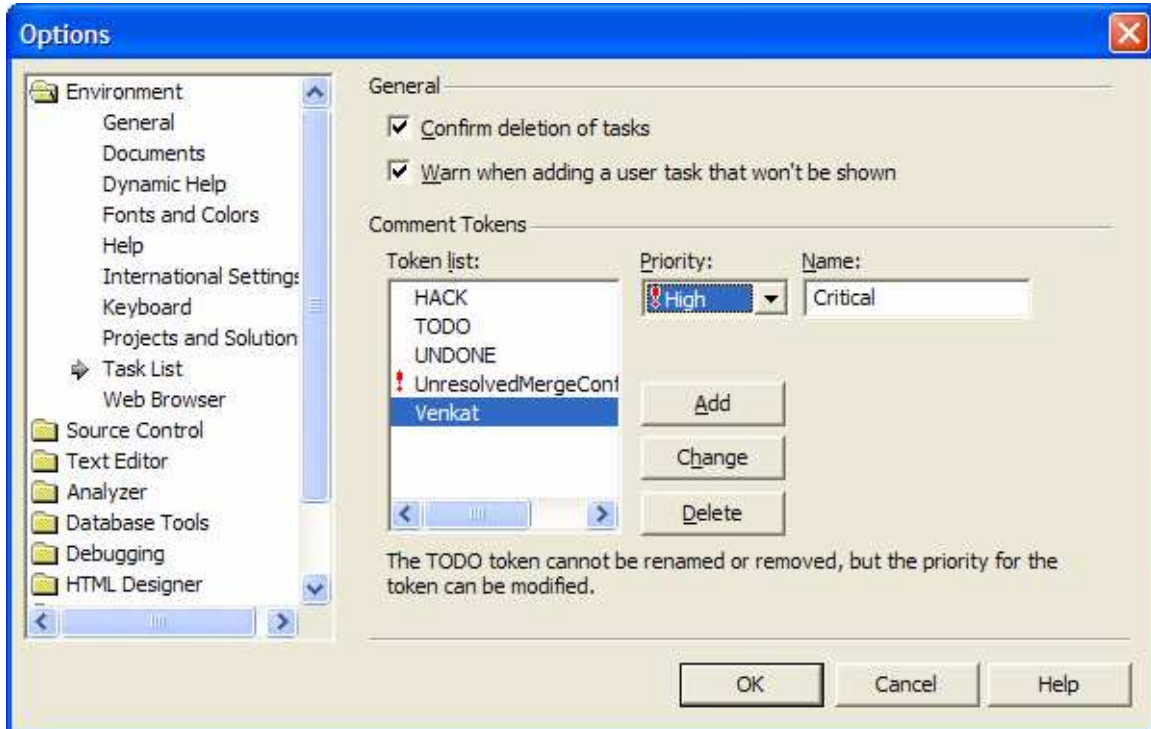
16 public void someMethod1()

```

Task List - 2 tasks		
!	Description	File
	Click here to add a new task	
◆	HACK fix this one later	C:\
▶ □	public void someMethod1()	C:\

You can create an item to appear on the Task List by right pressing **Ctrl+K** **Ctrl+H** (or by clicking on the code and selecting Add Task List Shortcut).

### 17. Create Custom Task List entries.



If you want to leave specific note to some one or represent critical tasks to be done later, you may create custom entries. Go to Tools | Options. Go to the Task List under Environment. Enter a Name and priority and click Add. Now you can annotate your code with these customized tasks.

```

16:         {
17:             // Critical: Must be fixed by demo next week
18:             // Venkat: Please review this for correctness
19:             // The following is Critical.
20:         }

```

Task List - 2 tasks		
!	Description	File
	Click here to add a new task	
! ◆	Critical: Must be fixed by demo next week	C:\
◆	Venkat: Please review this for correctness	C:\

### 18. Navigate through build errors.

Typically when you have more than one build error, you may want to navigate through each one of them. Having to use the mouse to click on the error in the Task List is inconvenient. Instead press the **F8** key to navigate through the errors.

19. Rebuild your solution using shortcut keys

Press **Ctrl+Shift+B** or **Alt+B Alt+B** to compile your code. Do not bother to save your code; it is saved automatically when you start compiling.

20. Place a break point in your code.

Easily place (or remove) break point in your code by pressing **F9**.

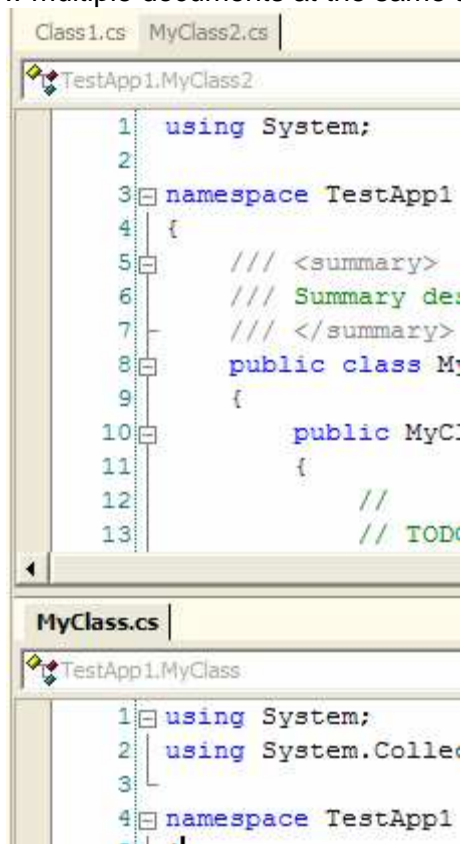
21. Execute or Debug your program using shortcut.

Press **F5** to start debugging. Press **Ctrl+F5** to start executing. Do not bother to save or compile your code. It is done for you automatically.

22. Quickly save a document.

To save a document (like the code you just edited), press **Ctrl+S**.

23. View multiple documents at the same time.



If you want to take a look at two source documents at the same time, you can do so by selecting the document you want in a different window and selecting Window | New Horizontal Tab Group. You can revert back to the grouping, by selecting Window | Move To Previous Tab Group.

24. View different parts of the same document.

```
Class1.cs | MyClass2.cs | MyClass.cs |
TestApp1.MyClass | MyClass()
1 using System;
2 using System.Collections;
```

```
Class1.cs | MyClass2.cs | MyClass.cs |
TestApp1.MyClass | MyClass()
16 public void someMethod1()
17 {
18     // Critical: Must be fixed by demo next week
using System;
using System.Collections;
```

If you want to view different parts of the same document, you can do so by dragging down the splitter window. When done, simply drag back the splitter to collapse the window.

#### 25. Quickly go to a method in your class.

```
Class1.cs | MyClass2.cs | MyClass.cs* |
TestApp1.MyClass | someMethod2()
1 using System;
2 using System.Collections;
```

If you want to go to a method in your class, you may scroll up and down looking for it. However, using the Members dropdown list, you can quickly go to method of interest in your class.

Got more tips? Please email [techlet@durasoftcorp.com](mailto:techlet@durasoftcorp.com) and we will add it here with acknowledgement to your name.