Is the #region directive really useful in .NET?

Asked 16 years, 2 months ago Modified 6 years, 2 months ago

Viewed 8k times

18

After maintaining lots of code littered with #region (in both C# and VB.NET), it seems to me that this construct is just a bunch of "make work" for the programmer. It's work to PUT the dang things into the code, and then they make searching and reading code very annoying.

What are the benefits? Why do coders go to the extra trouble to put this in their code.

Make me a believer!

.net coding-style

Share

Improve this question

Follow

edited Oct 4, 2008 at 0:44

Jeff Yates

62.3k ● 20 ● 142 ● 192

asked Oct 3, 2008 at 23:16

jm.

23.7k ● 23 ● 81 ● 93

17 Answers Sorted by: Highest score (default)

22

A similar question has already been asked.

but...

I would say **not anymore**. It was originally intended to hide *generated code* from WinForms in early versions of .NET. With partial classes the need seems to go away. IMHO it gets way overused now as an organizational construct and has no compiler value whatsoever. It's all for the IDE.

Share Improve this answer

Follow

edited May 23, 2017 at 12:17

Community Bot

1 ● 1

answered Oct 3, 2008 at 23:22

Scott Saad

18.3k ● 11 ● 66 ● 85

Oh. should i delete my question? – jm. Oct 3, 2008 at 23:23

In .NET whitespace has not compiler value either ;) Sure you can use partial classs to do what regions did, but how do you define where to split. – Aaron Powell Oct 3, 2008 at 23:24

1 @jm Don't think there is any need to delete the question. As long as we link back to the original. Plus this is a subjective matter anyway and kind of fun to talk about. – Scott Saad Oct 3, 2008 at 23:42

@Slace You're absolutely right... not sure where to draw the line there. I guess a start is the fact that the IDE separates

the generated code out into a file you really don't have to look at. – Scott Saad Oct 3, 2008 at 23:44

I've found it to be very useful from a pragmatic point of view - it very nicely organizes related methods or properties together, so the whole section can be easily ignored or inspected, as Russell Myers comment below suggests. – Mike Feb 23, 2009 at 21:52

17

Often times, both partials and #regions are used as a crutch for bad design (e.g. class is too big or tries to do too many things).

The *best* use I've had for #regions so far is the grouping of functionality that is seen in many different classes. For example, value objects that have getters, setters, constructors and supporting fields. I might very well group those ideas into regions. Its a matter of opinion, however, as to whether that makes code cleaner or harder to read.

Share Improve this answer

Follow

answered Oct 3, 2008 at 23:19

Russell Myers

2,017 ● 1 ● 15 ● 29

14

http://www.rauchy.net/regionerate/ - Automatically regionised your code ;)

I'm a fan of regions for grouping sections of large classes, say all the properties together, all constances, etc. I'm someone who's constantly collapsing code I don't need to see at that time so I love regions for that.

Also I find regions really useful when implementing interfaces, particularly multiple interfaces. I can group each interfaces methods, properties, events, etc so it's easier at a glance to see what method belongs to what interface.

Share Improve this answer

Follow

answered Oct 3, 2008 at 23:22

Aaron Powell

25.1k ● 18 ● 101 ● 150

> Now that we have partial classes, it's sometimes nicer to split each interface into it's own file. MyClass.cs and MyClass.IMyInterface.cs which also makes it visible when you're searching around the filesystem – Robert Paulson Oct 4, 2008 at 0:45

6

I use them all the time. Again, like anything else, they can be used for both evil and good, and can certainly be the hallmark of bad design, but they can be used to help organize code very well.

```
#region Properties

#region Update Section

#region Accessors
```

Certainly you should avoid Jeff's example of

```
#Sweep under carpet
```

What I find odd about them, as Jeff pointed out, is that they are a compiler preprocessor command for ui purposes. I'm sure VS team could have done something just as useful in another way.

Share Improve this answer

Follow

answered Oct 4, 2008 at 1:07

johnc

40.2k ● 37 ● 103 ● 140

1 I imagine now that VS 2010 will be using the totally revamped WPF-based editor we will start seeing all kinds of new improvements in that area. For example, I always thought I'd be awesome to have a "table" or inline RTF for doc comments. (But the saved text would be just plain text.) Now with the new editor it's doable. – Josh Jun 2, 2009 at 7:57

4

Our Business Objects all have Regions - and we love them.

We have;

- Business Properties and Methods
- Shared Methods
- Constructors
- Authorization
- Data Access
- Events

We have a few others depending on the type of Business Object we are dealing with (Subscriber etc)

For many classes regions just get in the way - but for our standard business objects they save us a ton of time. These Business Objects are code gen'd, so they are very consistent. Can get to where I want to be way faster than the clutter if they aren't, and the consistency makes it easy to find each other's stuff.

Share Improve this answer

Follow

answered Nov 17, 2008 at 19:05

aSkywalker

1,381 ● 1 ● 13 ● 24

3

I hate the over-use of these. The only think I find them useful for is hiding away things you probably never want to see again. Then again, those things should probably be off in a library somewhere.

Share Improve this answer

Follow

answered Nov 17, 2008 at 19:14

GeekyMonkey

12.9k ● 6 ● 35 ● 40

3

I don't generally use code regions, except in one specific case - dependency properties. Although dependecy properties are a pleasure to work with in most respects, their declaraions are an eyesore and they quickly clutter your code. (As if managing GUI code was not already enough of a challenge...)

I like to give the region the same exact name as the CLR property declaration (copy/paste it in there). That way you

can see the scope, type and name when it's collapsed - which is really all you care about 95% of the time.

```
   #region public int ObjectDepthThreshold

    public int ObjectDepthThreshold
    {
        get { return (int)GetValue(ObjectDepthThreshol
        set { SetValue(ObjectDepthThresholdProperty, v
    }

    public static readonly DependencyProperty ObjectDe
DependencyProperty.Register(
        &amp;quot;ObjectDepthThreshold&amp;quot;,
        typeof(int),
        typeof(GotoXControls),
        new
FrameworkPropertyMetadata((int)GotoXServiceState.OBJEC
            FrameworkPropertyMetadataOptions.AffectsRe
            new PropertyChangedCallback(OnControlValue
        )
    );

    #endregion
```

When it's collapsed you just see

```
public int ObjectDepthThreshold
```

If I have more than one dependency property, I like to start the next #region on the very next line. That way you end up with all of them grouped together in your class, and the code is compact and readable.

BTW if you just want to peek at the declaration, mouse hover over it.

answered Jul 30, 2012 at 21:48
Paul Williams
3,347 41 35

There are times when your methods HAVE to be long, especially with web development. In those cases (such as when I've got a gridview with a large, complex object bound to it) I've found it useful to use regions:

```
#region Declaring variables for fields and object prop

#region Getting the fields in scope

#region Getting the properties of the object

#region Setting Fields
```

These are discrete sections of the method that COULD be broken out, but it would be difficult (I'd have to use variables with larger scope than I like or pass a LOT of variables as 'out'), and it is basic plumbing.

In this case, regions are perfectly acceptable. In others, they are needless.

I will also use regions to group methods into logical groups. I reject partial classes for this purpose, as I tend to have a lot of pages open when I'm debugging, and the fewer partial classes there are in an object (or page, or dialog), the more of them I can have on my tab list (which I limit to one line so I can see more code).

Regions are only a problem when used as a crutch, or when they cover poor code (for instance, if you are nesting regions inside of each other within the same scope, it's a bad sign).

Share Improve this answer

Follow

edited Aug 17, 2017 at 13:29

C8H10N4O2

19k ● 9 ● 103 ● 142

answered Nov 13, 2008 at 16:16

Jeff

2,861 ● 3 ● 44 ● 70

2

Going on with what has been previously said by Russell Myers, if you learn how to refactor your code properly (a skill proficient developers must learn), there really isn't too much of a need for regions.

A couple of weeks ago I thought regions were great because they allowed me to hide my fat code, but after exercising my code skills I was able to make it slimmer and now I fit into a size 7 class (someone should SO make that a measurement for refactoring in the future! :P)

Share Improve this answer

Follow

answered Oct 3, 2008 at 23:27

RodgerB

8,658 ● 9 ● 38 ● 48

I find that they obfuscate the code in all but the simplest of uses. The only use we advocate in our projects are the

2

ones the IDE uses (interface implementations and designer code).

The right tools should be used for the right purpose. Code should be written to show intent and function rather than arbitrarily grouping things. Organizing things into access modifier grouping or some other grouping just seems to be illogical. I find the code should be organized in a manner that makes sense for the particular class; after all, there are other tools for viewing class members by access modifier. This is also the case for almost every other use of regions; there is a better way.

For example, grouping properties, events, constants or otherwise together doesn't really make sense either as code is generally more maintainable if the things are grouped together by function (as in, a property that uses a constant should be near that constant, not near other unrelated properties just because it's a property).

Share Improve this answer

Follow

answered Oct 4, 2008 at 0:41

Jeff Yates

62.3k • 20 • 142 • 192

2

I often use them *instead of comments* to order groups of functionality in the body of a class, e.g. "Configuration public interface", "Status public interface", "internal processing" and "internal worker thread management".

Using the keyboard shortcuts to "collapse to definitions" and "expand current block", I can easily navigate even

larger classes.

Unfortunately, Regions are broken for C++, and MS doesn't think it needs to be fixed.

Share Improve this answer

Follow

answered Nov 13, 2008 at 16:39

peterchen

41.1k ● 22 ● 108 ● 193

1

They can be overused, but I like them for separating private methods, public methods, properties, and instance variables.

Share Improve this answer

Follow

answered Oct 3, 2008 at 23:34

user25030

11 ● 1

1

Like any language feature, regions have the potential to be misused and abused but they also have their benefits.

They are great for creating "folding" groups around:

- methods, especially if you have a lot of overloaded methods
- interface implementations
- operator overloads

You can also use it to group properties, public/private methods, events, class-wide variables, etc.

I use regions in my code to help create a consistent structure in my code so I always know where things are at a glance. Yes, it makes things a bit harder during refactoring or adding new functions (especially when autogenerated by Visual Studio) but I feel it's a small price to pay to keep things consistent and structured.

Share Improve this answer

Follow

answered Oct 4, 2008 at 1:01

Scott Dorman

42.5k ● 12 ● 81 ● 112

1

Nice answers, I agree with them that say it sometimes reflects bad coding and design but #region actually is usefull if you're creating documentation (MSDN style) with the SandCastle. Lets say you have a public API and there is some base class that you want to give an example of usage for. Then you would properly document your public methods and add an example region where you could copy and paste some code. Problem with this is that when/if your base class changes you're supposed to change the example eventually. Better solution is to include a sample code project in your solution and build it all together, so everytime you build your solution if the sample code is not up to date it will not compile. So what does that have to do with regions you will be asking your self by now. Well look at this sample:

```
/// <example>
    /// The following code sample is an implementation
LoadPublishedVersion() for XmlPageProvider.
```

```
    /// <code
source="../CodeSamples/EPiServerNET/PageProvider/XmlPa
region="LoadPublishedVersion" lang="cs"/>
    /// </example>
```

Notice there is a link to the source code sample file and region for the method that you want to expose as a sample in your documentation. See here the result. That method needs to be in a proper region and will be automatically included in your documentation. That's why I wouldn't throw away #region yet.

Share Improve this answer Follow

edited Nov 17, 2008 at 18:43

answered Nov 13, 2008 at 16:37

Enes

3,961 ● 3 ● 27 ● 23

1

I love regions because it helps me focus on just what I am working on. I use them even if the class just has a method.

I use code snippets with regions already pre-populated, which is less typing. I feel the class is more organized and does what Code Complete talks about make it nicer for other people to read. The compiler just ignores them, they are now to make code more readable.

Share Improve this answer Follow

answered Feb 23, 2009 at 16:00

David Basarab
73.2k ● 43 ● 130 ● 157

0

There really isn't a benefit. They are a code smell. After using them for awhile, I got sick of them. If you need to break things out by functionality, use a partial class.

Share Improve this answer

Follow

answered Oct 3, 2008 at 23:23

MagicKat
9,811 ● 7 ● 34 ● 43

0

My working day starts with opening files in editor and clicking on "Expand All" to hide all regions. After that I can begin to work.

Share Improve this answer

Follow

answered Mar 24, 2009 at 9:15

User
30.9k ● 22 ● 81 ● 108