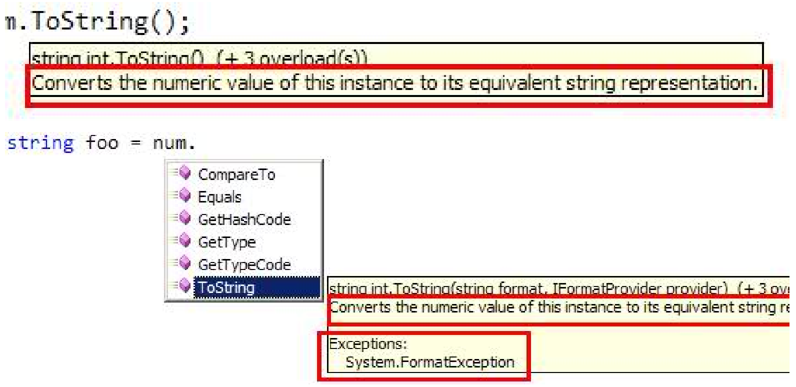
# How to have comments in IntelliSense for function in Visual Studio?

Ask Question

120

19

In Visual Studio and C#, when using a built in function such as ToString(), IntelliSense shows a yellow box explaining what it does.



```
n.ToString();
string int.ToString() (+ 3 overload(s))
Converts the numeric value of this instance to its equivalent string representation.

string foo = num.
```

```
string int.ToString(string format, IFormatProvider provider) (+ 3 ove
Converts the numeric value of this instance to its equivalent string re

Exceptions:
    System.FormatException
```

**How can I have that for functions and properties I write?**

c# vb.net visual-studio visual-studio-2008 xml-comments

edited Feb 28 at 20:03
Glorfindel
16.7k 11 52 73

asked Feb 9 '09 at 20:02
Ali

## 12 Answers

194

To generate an area where you can specify a description for the function and each parameter for the function, type the following on the line before your function and hit Enter:

- **C#:** `///`
- **VB:** `'''`

See Recommended Tags for Documentation Comments (C# Programming Guide) for more info on the structured content you can include in these comments.

edited Feb 17 '17 at 14:59
Kenny Evitt
**6,083** 3 46 62

answered Feb 9 '09 at 20:04
Solmead
**3,149** 2 20 29

1 To emphasize: That is triple-slash in C++/C# (normal comments are double-slash). And in VB, its two single-quotes, not a double-quote. – abelenky Feb 9 '09 at 20:06

1 It's actually three single quotes in vb – Joel Coehoorn Feb 9 '09 at 20:13

1 Actually, in VB, it's 3 single quotes: ''' – hometoast Feb 9 '09 at 20:13

5 us VB-folk are always left out XD – hometoast Feb 9 '09 at 20:13

2 As an alternative, in a VB file you can right click on a function or class and click "Insert Comment". For C# you can use StyleCop which will prompt you to write good documentation

62

What you need is **xml comments** - basically, they follow this syntax (as vaguely described by Solmead):

**C#**

```
///<summary>
///This is a description of my function.
///</summary>
string myFunction() {
     return "blah";
}
```

**VB**

```
'''<summary>
'''This is a description of my function.
'''</summary>
Function myFunction() As String
    Return "blah"
End Function
```

edited Nov 10 '16 at 13:57
jrh
**225** 2 7 22

answered Feb 9 '09 at 20:08
Tomas Aschan
**35.5k** 35 172 325

17

`<c>text</c>` - The text you would like to indicate as code.
The <*c*> tag gives you a way to indicate that text within a description should be marked as code. Use <*code*> to indicate multiple lines as code

`<code>content</code>` - The text you want marked as code.
The <*code*> tag gives you a way to indicate multiple lines as code. Use <*c*> to indicate that text within a description should be marked as code.

`<example>description</example>` - A description of the code sample.
The <*example*> tag lets you specify an example of how to use a method or other library member. This commonly involves using the <*code*> tag.

`<exception cref="member">description</exception>` - A description of the exception.
The <*exception*> tag lets you specify which exceptions can be thrown. This tag can be applied to definitions for methods, properties, events, and indexers.

`<include file='filename' path='tagpath[@name="id"]' />`
The <*include*> tag lets you refer to comments in another file that describe the types and members in your source code. This is an alternative to placing documentation comments directly in your source code file. By putting the documentation in a separate file, you can apply source control to the documentation separately from the source code. One person can have the source code file checked out and someone else can have the documentation file checked out. The <*include*> tag uses the XML XPath syntax. Refer to XPath documentation for ways to customize your <*include*> use.

```
<list type="bullet" | "number" | "table">
    <listheader>
        <term>term</term>
        <description>description</description>
    </listheader>
    <item>
        <term>term</term>
```

The <*listheader*> block is used to define the heading row of either a table or definition list. When defining a table, you only need to supply an entry for term in the heading. Each item in the list is specified with an <*item*> block. When creating a definition list, you will need to specify both term and description. However, for a table, bulleted list, or numbered list, you only need to supply an entry for description. A list or table can have as many <*item*> blocks as needed.

`<para>content</para>`
The <*para*> tag is for use inside a tag, such as <*summary*>, <*remarks*>, or <*returns*>, and lets you add structure to the text.

`<param name="name">description</param>`
The <*param*> tag should be used in the comment for a method declaration to describe one of the parameters for the method. To document multiple parameters, use multiple <*param*> tags.
The text for the <*param*> tag will be displayed in IntelliSense, the Object Browser, and in the Code Comment Web Report.

`<paramref name="name"/>`
The <*paramref*> tag gives you a way to indicate that a word in the code comments, for example in a <*summary*> or <*remarks*> block refers to a parameter. The XML file can be processed to format this word in some distinct way, such as with a bold or italic font.

< `permission cref="member">description</permission>`
The <*permission*> tag lets you document the access of a member. The PermissionSet class lets you specify access to a member.

<*summary*>. This information is displayed in the Object Browser.

`<returns>description</returns>`
The <*returns*> tag should be used in the comment for a method declaration to describe the return value.

`<see cref="member"/>`
The <*see*> tag lets you specify a link from within text. Use <*seealso*> to indicate that text should be placed in a See Also section. Use the cref Attribute to create internal hyperlinks to documentation pages for code elements.

`<seealso cref="member"/>`
The <*seealso*> tag lets you specify the text that you might want to appear in a See Also section. Use <*see*> to specify a link from within text.

`<summary>description</summary>`
The <*summary*> tag should be used to describe a type or a type member. Use <*remarks*> to add supplemental information to a type description. Use the cref Attribute to enable documentation tools such as Sandcastle to create internal hyperlinks to documentation pages for code elements. The text for the <*summary*> tag is the only source of information about the type in IntelliSense, and is also displayed in the Object Browser.

`<typeparam name="name">description</typeparam>`
The <*typeparam*> tag should be used in the comment for a generic type or method declaration to describe a type parameter. Add a tag for each type parameter of the generic type or method. The text for the <*typeparam*> tag will be displayed in IntelliSense, the Object Browser code comment web report.

`<typeparamref name="name"/>`

```
<value>property-description</value>
```

The <*value*> tag lets you describe the value that a property represents. Note that when you add a property via code wizard in the Visual Studio .NET development environment, it will add a <*summary*> tag for the new property. You should then manually add a <*value*> tag to describe the value that the property represents.

answered May 30 '16 at 22:34

Max
**301** 3 14

11

Do XML commenting , like this

```
/// <summary>
/// This does something that is awesome
/// </summary>
public void doesSomethingAwesome() {}
```

edited Feb 9 '09 at 20:13

Joel Coehoorn
**313k** 96 497 736

answered Feb 9 '09 at 20:06

Michael Walts
**408** 2 8

5 For parameters add: `///<param name="paramName">Tralala</param>` – The Oddler Jul 1 '14 at 17:07

10

use /// to begin each line of the comment and have the comment contain the appropriate xml for the meta data reader.

```
///<summary>
/// this method says hello
///</summary>
public void SayHello();
```

Although personally, I believe that these comments are usually misguided, unless you are developing classes where the code cannot be read by its consumers.

edited Nov 24 '12 at 0:17
Robert Oschler
**7,125** 15 54 172

answered Feb 9 '09 at 20:07
DevelopingChris
**20.8k** 26 80 116

2 they're good for shortcuts reminders, or anywhere you have library code where maybe the code is readable but it takes a little extra work to get to it. – Joel Coehoorn Feb 9 '09 at 20:14

1 I agree with you in theory, but if you use that ghostdoc thing, then you are raising the noise/signal ratio to such an extent that the rest of the comments are useless. – DevelopingChris Feb 10 '09 at 15:26

9

Those are called XML Comments. They have been a part of Visual Studio since forever.

You can make your documentation process easier by caret on the method/property you want to document, and press Ctrl-Shift-D.

Here's an example from one of my posts.

Hope that helps :)

edited May 23 '17 at 12:02
Community ♦
1 1

answered Feb 9 '09 at 20:09
Igal Tabachnik
27k 14 77 140

3 Thanks for mentioning GhostDoc. – Ali Feb 9 '09 at 21:42

6

In CSharp, If you create the method/function outline with it's Parms, then when you add the three forward slashes it will auto generate the summary and parms section.

So I put in:

```
public string myMethod(string sImput1, int iInput2)
{
}
```

I then put the three /// before it and Visual Studio's gave me this:

```
/// <summary>
///
/// </summary>
/// <param name="sImput1"></param>
/// <param name="iInput2"></param>
```

```
{
}
```

answered Dec 11 '15 at 22:19

Semperfi89
**61** 1 4

4

read http://msdn.microsoft.com/en-us/library/3260k4x7.aspx Just specifying the comments will not show the help comments in intellisense.

answered May 24 '12 at 19:00

user1180781
**61** 4

They will if you have XML comments enabled. See my answer below. – Suncat2000 Nov 15 '16 at 14:10

3

Define Methods like this and you will get the help you need.

```
/// <summary>
/// Adds two numbers and returns the result
/// </summary>
/// <param name="first">first number to add</param>
/// <param name="second">second number to </param>
/// <returns></returns>
private int Add(int first, int second)
{
    return first + second;
}
```

answered Dec 16 '16 at 11:17

Recev Yildiz
**441** 6 7

---

1

Also the visual studio add-in ghost doc will attempt to create and fill-in the header comments from your function name.

answered Feb 9 '09 at 20:07

Mark Rogers
**42.1k** 17 70 117

---

1

All these others answers make sense, but are incomplete. Visual Studio will process XML comments but you have to turn them on. Here's how to do that:

Intellisense will use XML comments you enter in your source code, but you must have them enabled through Visual Studio Options. Go to `Tools` > `Options` > `Text Editor` . For Visual Basic, enable the `Advanced` > `Generate XML documentation comments for '''` setting. For C#, enable the `Advanced` > `Generate XML documentation comments for ///` setting. Intellisense will use the summary comments when entered. They will be available from other projects after the referenced project is compiled.

To create *external* documentation, you need to generate an XML file through the `Project Settings` > `Build` > `XML documentation file:` path that controls the compiler's `/doc` option. You will need an external tool that will take the XML file as input and generate the documentation in your

Be aware that generating the XML file can noticeably increase your compile time.

answered Nov 15 '16 at 14:15

Suncat2000
**500** 4 11

---

0

Solmead has the correct answer. For more info you can look at XML Comments.

answered Feb 9 '09 at 20:06

Ed S.
**103k** 18 150 225