**Caller Information (C# and Visual Basic)**

**Visual Studio 2013**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2012](http://msdn.microsoft.com/en-us/library/hh534540(d=printer,v=vs.110).aspx)

By using Caller Info attributes, you can obtain information about the caller to a method. You can obtain file path of the source code, the line number in the source code, and the member name of the caller. This information is helpful for tracing, debugging, and creating diagnostic tools.

To obtain this information, you use attributes that are applied to optional parameters, each of which has a default value. The following table lists the Caller Info attributes that are defined in the [System.Runtime.CompilerServices](http://msdn.microsoft.com/en-us/library/system.runtime.compilerservices.aspx) namespace:

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Type |
| [CallerFilePathAttribute](http://msdn.microsoft.com/en-us/library/system.runtime.compilerservices.callerfilepathattribute.aspx) | Full path of the source file that contains the caller. This is the file path at compile time. | String |
| [CallerLineNumberAttribute](http://msdn.microsoft.com/en-us/library/system.runtime.compilerservices.callerlinenumberattribute.aspx) | Line number in the source file at which the method is called. | Integer |
| [CallerMemberNameAttribute](http://msdn.microsoft.com/en-us/library/system.runtime.compilerservices.callermembernameattribute.aspx) | Method or property name of the caller. See [Member Names](http://msdn.microsoft.com/en-us/library/hh534540(d=printer,v=vs.120).aspx#MEMBERNAMES) later in this topic. | String |

[Example](javascript:void(0))

The following example shows how to use Caller Info attributes. On each call to the TraceMessage method, the caller information is substituted as arguments to the optional parameters.

C#

[VB](http://msdn.microsoft.com/en-us/library/hh534540(d=printer,v=vs.120).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

// using System.Runtime.CompilerServices

// using System.Diagnostics;

public void DoProcessing()

{

TraceMessage("Something happened.");

}

public void TraceMessage(string message,

[CallerMemberName] string memberName = "",

[CallerFilePath] string sourceFilePath = "",

[CallerLineNumber] int sourceLineNumber = 0)

{

Trace.WriteLine("message: " + message);

Trace.WriteLine("member name: " + memberName);

Trace.WriteLine("source file path: " + sourceFilePath);

Trace.WriteLine("source line number: " + sourceLineNumber);

}

// Sample Output:

// message: Something happened.

// member name: DoProcessing

// source file path: c:\Users\username\Documents\Visual Studio 2012\Projects\CallerInfoCS\CallerInfoCS\Form1.cs

// source line number: 31

[Remarks](javascript:void(0))

You must specify an explicit default value for each optional parameter. You can't apply Caller Info attributes to parameters that aren't specified as optional.

The Caller Info attributes don't make a parameter optional. Instead, they affect the default value that's passed in when the argument is omitted.

Caller Info values are emitted as literals into the Intermediate Language (IL) at compile time. Unlike the results of the [StackTrace](http://msdn.microsoft.com/en-us/library/system.exception.stacktrace.aspx) property for exceptions, the results aren't affected by obfuscation.

[Member Names](javascript:void(0))

You can use the **CallerMemberName** attribute to avoid specifying the member name as a String argument to the called method. By using this technique, you avoid the problem that Rename Refactoring doesn't change the String values. This benefit is especially useful for the following tasks:

* Using tracing and diagnostic routines.
* Implementing the [INotifyPropertyChanged](http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged.aspx) interface when binding data. This interface allows the property of an object to notify a bound control that the property has changed, so that the control can display the updated information. Without the **CallerMemberName** attribute, you must specify the property name as a literal.

The following chart shows the member names that are returned when you use the **CallerMemberName** attribute.

|  |  |
| --- | --- |
| **Calls occurs within** | **Member name result** |
| Method, property, or event | The name of the method, property, or event from which the call originated. |
| Constructor | The string ".ctor" |
| Static constructor | The string ".cctor" |
| Destructor | The string "Finalize" |
| User-defined operators or conversions | The generated name for the member, for example, "op\_Addition". |
| Attribute constructor | The name of the member to which the attribute is applied. If the attribute is any element within a member (such as a parameter, a return value, or a generic type parameter), this result is the name of the member that's associated with that element. |
| No containing member (for example, assembly-level or attributes that are applied to types) | The default value of the optional parameter. |