**Object.Finalize Method**

**.NET Framework 4.5**

Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.

**Namespace:**  [System](http://msdn.microsoft.com/en-us/library/system(v=vs.110).aspx)  
**Assembly:**  mscorlib (in mscorlib.dll)

[Syntax](javascript:void(0))

C#

protected virtual void Finalize()

[Remarks](javascript:void(0))

The Finalize method is used to perform cleanup operations on unmanaged resources held by the current object before the current object is destroyed. The method is protected and therefore is accessible only through this class or through a derived class.

This method is automatically called after an object becomes inaccessible, unless the object has been exempted from finalization by a call to [GC.SuppressFinalize](http://msdn.microsoft.com/en-us/library/system.gc.suppressfinalize(v=vs.110).aspx). During shutdown of an application domain, Finalize is automatically called on objects that are not exempt from finalization, even those that are still accessible. Finalize is automatically called only once on a given instance, unless the object is re-registered using a mechanism such as [GC.ReRegisterForFinalize](http://msdn.microsoft.com/en-us/library/system.gc.reregisterforfinalize(v=vs.110).aspx) and [GC.SuppressFinalize](http://msdn.microsoft.com/en-us/library/system.gc.suppressfinalize(v=vs.110).aspx) has not been subsequently called.

Every implementation of Finalize in a derived type must call its base type's implementation of Finalize. This is the only case in which application code is allowed to call Finalize.

|  |
| --- |
| **Description: NoteNote** |
| Because the C# compiler does not allow you to directly implement the Finalize method, a C# destructor automatically calls the destructor of its base class. |

Finalize operations have the following limitations:

* The exact time when the finalizer executes during garbage collection is undefined. Resources are not guaranteed to be released at any specific time, unless calling a Close method or a Dispose method.
* The finalizers of two objects are not guaranteed to run in any specific order, even if one object refers to the other. That is, if Object A has a reference to Object B and both have finalizers, Object B might have already finalized when the finalizer of Object A starts.
* The thread on which the finalizer is run is unspecified.

The Finalize method might not run to completion or might not run at all in the following exceptional circumstances:

* Another finalizer blocks indefinitely (goes into an infinite loop, tries to obtain a lock it can never obtain and so on). Because the runtime attempts to run finalizers to completion, other finalizers might not be called if a finalizer blocks indefinitely.
* The process terminates without giving the runtime a chance to clean up. In this case, the runtime's first notification of process termination is a DLL\_PROCESS\_DETACH notification.

The runtime continues to Finalize objects during shutdown only while the number of finalizable objects continues to decrease.

If Finalize or an override of Finalize throws an exception, and the runtime is not hosted by an application that overrides the default policy, the runtime terminates the process and no active try-finally blocks or finalizers are executed. This behavior ensures process integrity if the finalizer cannot free or destroy resources.

Notes to Implementers

Object.Finalize does nothing by default. It must be overridden by a derived class only if necessary, because reclamation during garbage collection tends to take much longer if a Finalize operation must be run.

If an [Object](http://msdn.microsoft.com/en-us/library/system.object(v=vs.110).aspx) holds references to any resources, Finalize must be overridden by a derived class in order to free these resources before the [Object](http://msdn.microsoft.com/en-us/library/system.object(v=vs.110).aspx) is discarded during garbage collection.

A type must implement Finalize when it uses unmanaged resources such as file handles or database connections that must be released when the managed object that uses them is reclaimed. See the [IDisposable](http://msdn.microsoft.com/en-us/library/system.idisposable(v=vs.110).aspx) interface for a complementary and more controllable means of disposing resources.

Finalize can take any action, including resurrecting an object (that is, making the object accessible again) after it has been cleaned up during garbage collection. However, the object can only be resurrected once; Finalize cannot be called on resurrected objects during garbage collection.

Destructors are the C# mechanism for performing cleanup operations. Destructors provide appropriate safeguards, such as automatically calling the base type's destructor. In C# code, Object.Finalize cannot be called or overridden.

[Examples](javascript:void(0))

The following example verifies that the Finalize method is called when an object that overrides Finalize is destroyed. Note that, in a production application, the Finalize method would be overridden to release unmanaged resources held by the object. Also note that the C# example provides a destructor instead of overriding the Finalize method.

C#

using System;

using System.Diagnostics;

public class ExampleClass

{

Stopwatch sw;

public ExampleClass()

{

sw = Stopwatch.StartNew();

Console.WriteLine("Instantiated object");

}

public void ShowDuration()

{

Console.WriteLine("This instance of {0} has been in existence for {1}",

this, sw.Elapsed);

}

~ExampleClass()

{

Console.WriteLine("Finalizing object");

sw.Stop();

Console.WriteLine("This instance of {0} has been in existence for {1}",

this, sw.Elapsed);

}

}

public class Demo

{

public static void Main()

{

ExampleClass ex = new ExampleClass();

ex.ShowDuration();

}

}

// The example displays output like the following:

// Instantiated object

// This instance of ExampleClass has been in existence for 00:00:00.0011060

// Finalizing object

// This instance of ExampleClass has been in existence for 00:00:00.0036294