回收资源

1. 终结器（C# 编程指南）

**https://docs.microsoft.com/zh-cn/dotnet/csharp/programming-guide/classes-and-structs/destructors**

* + - * 1. 说明：不要使用空的析构函数。

如果类包含析构函数，则 Finalize  队列中则会创建一个项。当调用析构函数时，将调用垃圾回收器（GC）来处理该队列。如果析构函数为空，只会导致不必要的性能损失。

程序员无法控制何时调用析构函数，因为这是由垃圾回收器决定的。垃圾回收器检查是否存在应用程序不再使用的对象。如果垃圾回收器认为某个对象符合析构，则调用析构函数（如果有的话），回收该对象的内存。程序退出时也会调用析构函数。

可以通过调用 Collect 强制进行垃圾回收，但大多数情况下应避免这样做，因为会导致性能问题。

通常，.NET Framework 垃圾回收器会隐式地管理对象的内存分配和释放。但当应用程序封装窗口、文件和网络连接这类非托管资源时，应使用析构函数释放这些资源。当对象符合析构时，垃圾回收器将运行对象的 Finalize 方法。

虽然垃圾回收器可以跟踪封装非托管资源的对象的生存期，但它不了解具体如何清理这些资源。常见的非托管源有：ApplicationContext、Brush、Component、ComponentDesigner、Container、Context、Cursor、FileStream、Font、Icon、Image、Matrix、Object、OdbcDataReader、OleDBDataReader、Pen、Regex、Socket、StreamWriter、Timer、Tooltip 等。

1. GC.SuppressFinalize 使对象免除了终结

注意：C# 编译器不允许你直接实现 Finalize 方法，因此 C# 析构函数自动调用其基类的析构函数。

using System;

public class Destroyer

{

public override string ToString() => GetType().Name;

~Destroyer() => Console.WriteLine($"The {ToString()} destructor is executing.");

}

public sealed class Foo : IDisposable

{

private readonly IDisposable \_bar;

public Foo()

{

\_bar = new Bar();

}

public void Dispose()

{

\_bar?.Dispose();

}

}