Armen Melkumyan • 1st
Technical / Solutions Architect
1yr •

From .Net C# interview questions

Question:

What is the purpose of calling GC.SuppressFinalize ?

Answer:

Garbage Collector: The garbage collector (GC) in .NET is responsible for freeing up memory that is no longer in use. Part of this process involves calling finalizers for objects before they are actually removed from memory.

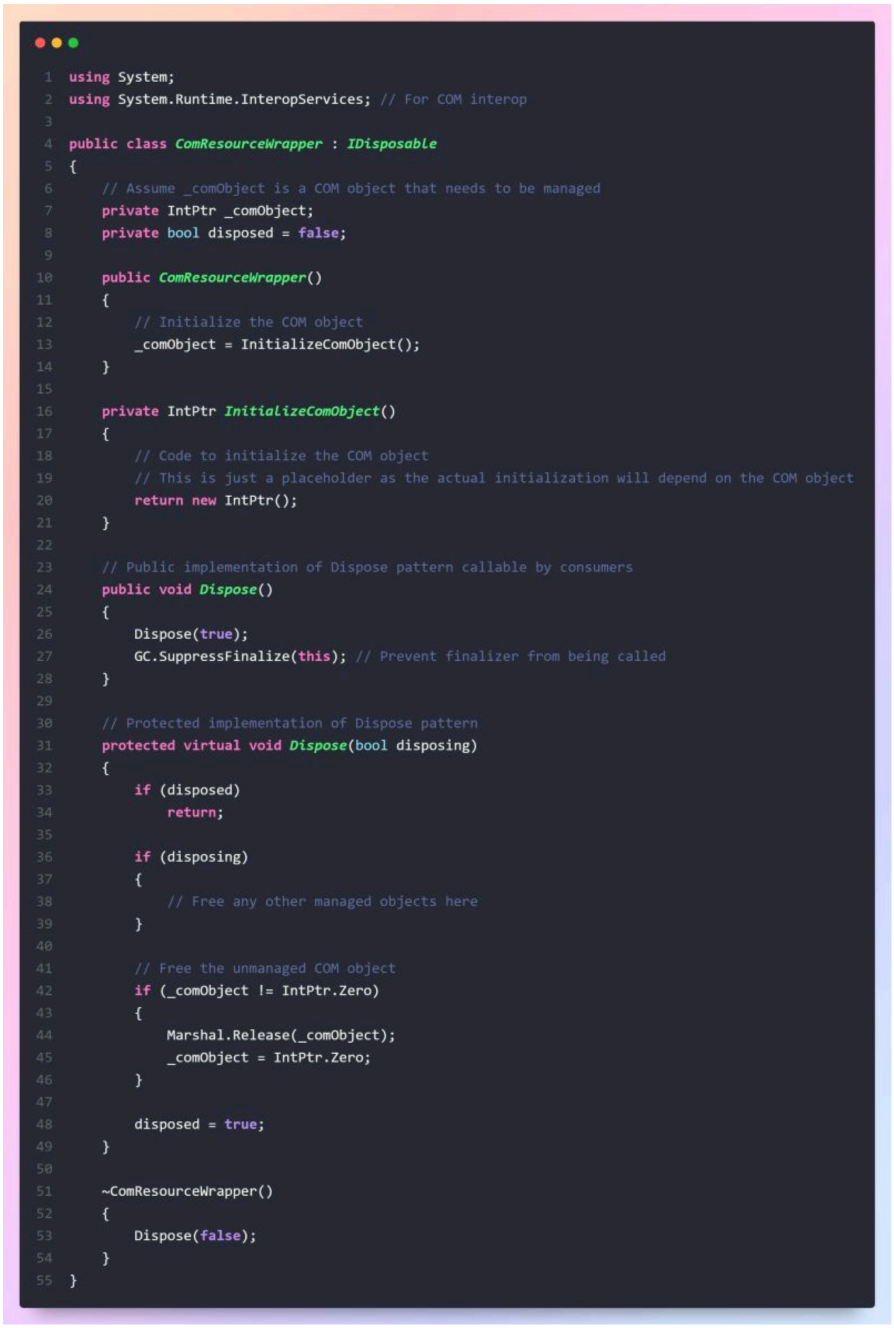
Finalizer: A finalizer is a special method in a class (usually denoted in C# by the ~ClassName syntax) that is designed to release unmanaged resources or perform other cleanup operations just before the object is collected by the GC.
GC.SuppressFinalize Method: When you call GC.SuppressFinalize and pass an object to it, you are instructing the garbage collector that it does not need to execute the finalizer for that particular object.

Why Suppress Finalizers: The main reason to suppress a finalizer is performance. Finalizers add overhead to the garbage collection process because the GC has to make an additional pass to call them. If you have already manually cleaned up the resources (usually in a Dispose method), then the finalizer has nothing left to do. By calling GC.SuppressFinalize, you reduce the workload of the garbage collector, allowing it to be more efficient.

In summary, calling GC.SuppressFinalize for an object is a way to tell the garbage collector, "This object's finalizer does not need to be run because I have already taken care of releasing all the resources it was using." This is typically done after you've manually performed cleanup in a Dispose method, ensuring that all necessary resource releases are handled efficiently.

#csharp #dotnet #gc #csharpdeveloper #interview #interviewquestion #softwareengineering

#GarbageCollection, #DotNetFramework, #SoftwareDevelopment, #CodingInterview, #TechInterview



```csharp
using System;
using System.Runtime.InteropServices; // For COM interop

public class ComResourceWrapper : IDisposable
{
    // Assume _comObject is a COM object that needs to be managed
    private IntPtr _comObject;
    private bool disposed = false;

    public ComResourceWrapper()
    {
        // Initialize the COM object
        _comObject = InitializeComObject();
    }

    private IntPtr InitializeComObject()
    {
        // Code to initialize the COM object
        // This is just a placeholder as the actual initialization will depend on the COM object
        return new IntPtr();
    }

    // Public implementation of Dispose pattern callable by consumers
    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this); // Prevent finalizer from being called
    }

    // Protected implementation of Dispose pattern
    protected virtual void Dispose(bool disposing)
    {
        if (disposed)
            return;

        if (disposing)
        {
            // Free any other managed objects here
        }

        // Free the unmanaged COM object
        if (_comObject != IntPtr.Zero)
        {
            Marshal.Release(_comObject);
            _comObject = IntPtr.Zero;
        }

        disposed = true;
    }

    ~ComResourceWrapper()
    {
        Dispose(false);
    }
}
```

8 1 repost

Like Comment Repost Send