Mastering .NET Memory Management & Garbage Collection

Q1: IS THERE A WAY WE CAN SEE THIS HEAP MEMORY?

Yes, we can analyze GC using performance counters such as GC Heap size, GC0, GC1, GC2, and working set.

Q2: DOES GARBAGE COLLECTOR CLEAN PRIMITIVE TYPES?

No, primitive types are stored on the stack and cleaned up as soon as they go out of scope.

Q3: MANAGED VS UNMANAGED CODE/OBJECTS/RESOURCES?

.NET CLR controls managed resources (pure .NET objects). Unmanaged resources include file handles, COM objects, etc.

Ajay Saini

Q4: CAN GARBAGE COLLECTOR CLEAN UNMANAGED CODE?

No, GC only cleans managed objects.

Q5: EXPLAIN GENERATIONS?

Generations are logical buckets that hold objects by their age (Gen0, Gen1, Gen2).

Q6: WHAT IS GC0, GC1, AND GC2?

GC0: Short-lived (e.g., local variables). GC1: Intermediate (buffers). GC2: Long-lived (e.g., static objects).

Q7: WHY DO WE NEED GENERATIONS?

For performance optimization. GC assumes newer objects are short-lived and older ones are less frequently collected.

Mastering .NET Memory Management & Garbage Collection

Q8: WHICH IS THE BEST PLACE TO CLEAN UNMANAGED OBJECTS?

The destructor is the best place to clean unmanaged objects.

Q9: HOW DOES GC BEHAVE WHEN WE HAVE A DESTRUCTOR?

GC takes more trips to clean them and promotes objects to upper generations, increasing memory pressure.

Q10: WHAT DO YOU THINK ABOUT AN EMPTY DESTRUCTOR?

Empty destructors harm performance by promoting objects unnecessarily.

Q11: EXPLAIN THE DISPOSE PATTERN?

Implement IDisposable and call GC.SuppressFinalize() to avoid finalizer overhead.

Ajay Saini

Q12: FINALIZE VS DESTRUCTOR?

They are the same; the destructor calls the Finalize method.

Q13: WHAT IS THE USE OF THE USING KEYWORD?

Defines a scope at the end of which Dispose() is called automatically.

Q14: CAN YOU FORCE GARBAGE COLLECTOR?

Yes, by calling GC.Collect().

Q15: IS IT A GOOD PRACTICE TO FORCE GC?

No, GC manages memory efficiently on its own and manual interference can degrade performance.

Mastering .NET Memory Management & Garbage Collection

Q16: HOW CAN WE DETECT MEMORY ISSUES?

Use tools like Visual Studio Profiler. Linear memory increase or imbalance in allocation/deallocation indicates issues.

Q17: HOW CAN WE KNOW THE EXACT SOURCE OF MEMORY ISSUES?

Analyze the top memory-consuming objects in the profiler and review related code.

Q18: WHAT IS A MEMORY LEAK?

When memory isn't returned to the OS even after the application ends.

Q19: CAN A .NET APPLICATION HAVE A MEMORY LEAK AS WE HAVE GC?

Yes, GC only manages managed memory. Improper handling of unmanaged resources causes leaks.

Ajay Saini

Q20: HOW TO DETECT MEMORY LEAKS IN .NET APPLICATIONS?

If total memory increases while managed memory remains stable, there's likely an unmanaged memory leak.