Armen Melkumyan • 1st
Technical / Solutions Architect
2mo •

.NET Advanced Interview Questions: JIT Compilation & Optimizations

For senior .NET engineers, understanding JIT compilation internals is crucial when optimizing performance. Let's break down how .NET JIT works and the optimizations it applies.

⚡ JIT (Just-In-Time) Compilation in .NET

.NET uses JIT compilation to convert Intermediate Language (IL) into machine code at runtime. The JIT compiler applies several key optimizations:

✅ Inlining – Small methods are replaced directly at call sites to eliminate method call overhead.

✅ Tiered Compilation – Starts with a quick, unoptimized version and later replaces it with a fully optimized version if frequently executed.

✅ Loop Unrolling – Reduces branch instructions by unrolling loops where applicable.

✅ Constant Folding – Precomputes constant expressions at compile time.

✅ Dead Code Elimination – Removes unreachable code to optimize performance.

✅ Devirtualization – If the compiler determines the method at runtime, it avoids virtual method calls.

✅ Register Allocation – Stores frequently accessed values in CPU registers to minimize memory access.

📝 Example: Forcing Method Inlining

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static int Add(int a, int b) => a + b;
```

Using AggressiveInlining ensures the method is inlined, eliminating the method call overhead.

🏗️ Why It Matters?

Understanding JIT optimizations helps when working on performance-critical applications, reducing execution time and improving efficiency.

#dotnet #performance #jit