

Mohamed Nagy • You
Full Stack Developer | ASP.NET Core
now •

...

ال Boxing & Unboxing

لما ال Design ييجي على حساب ال Performance في Software Design، دايماً بتسعى إن الكود يكون Flexible و Reusable. لكن أحياناً بنقع في فخ استخدام النوع object علشان الميثود تستقبل أي نوع من البيانات. هنا بيظهر مفهوم Boxing & Unboxing.

لما بتصمم Method بتأخذ Parameter من نوع object وأنت ناوي تيعتله Value Types زي int أو double، ال Compiler بيعمل Boxing، يعني بيحول ال Value Type ل Object و يتخط في ال Heap Memory.

الموضوع ده مش بس بيأثر على ال Performance بسبب ال Memory Allocation الإضافي، لكنه كمان بيكسر مبدأ مهم من مبادئ ال Clean Code وهو Type Safety.

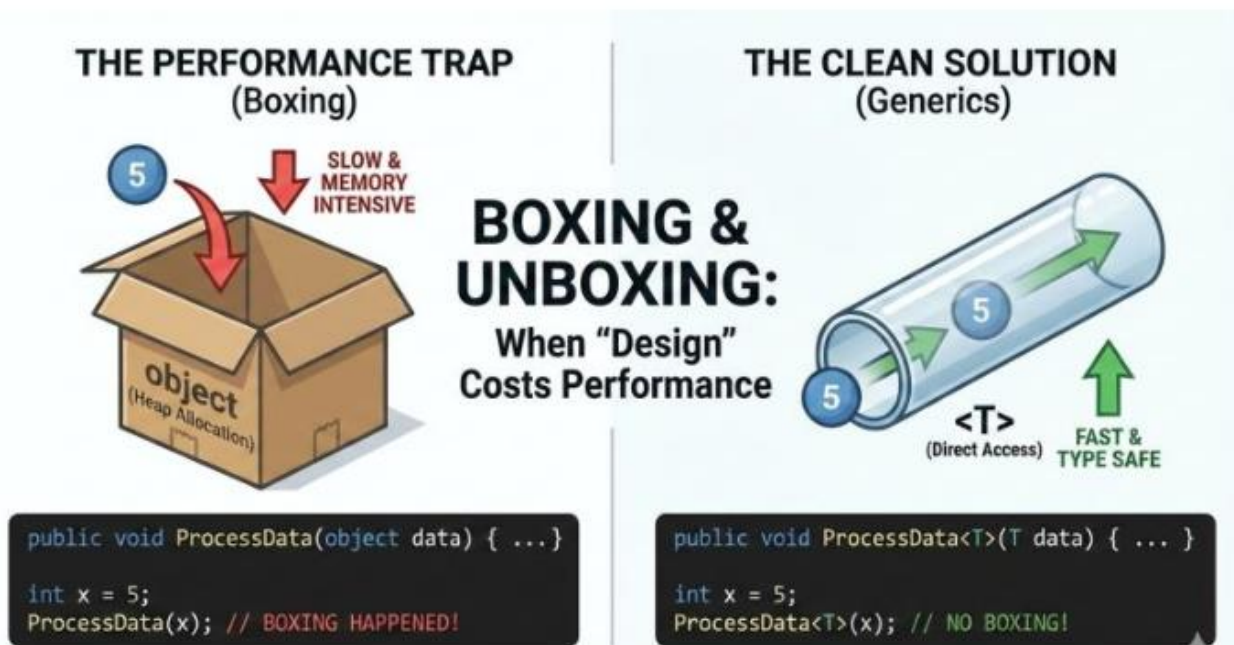
التصميم الأنصف والأصح هو استخدام Generics.

ال Generics بتخليك تحقق المرونة (Reusability) من غير ما تضحي بالسرعة، لأن ال Compiler بيتعامل مع النوع الأصلي مباشرة، وده يمنع ال Boxing ويحافظ على ال Type Safety.

الخلاصة إن SOLID Principles و Clean Architecture مش بس شكل كود حلو وتنظيم ملفات، لكن كمان اختيار صحيح لل Data Types وال Design Patterns اللي بتحافظ على موارد السيرفر وتديك أفضل أداء. بلاش تستخدم object بدافع الكسل، واستخدم Generics علشان كودك يبقى Clean, Safe, and Fast.

[SoftwareArchitecture](#) [CleanCode](#) [DotNet](#) [CSharp](#) [Performance](#) [Boxing](#) [Generics](#) [SoftwareEngineering](#)

1-



2-Search about upcasting and downcasting and its relation to inheritance concept as Design wise?

- a Child object as a Parent (Safe & Automatic). Used for flexibility
- a Parent object as a Child (Manual & Risky). Needs type checking

3-Unmanaged Resources Recovery & I/O Operations?

- Unmanaged resources must be released manually to avoid memory leaks.
- Using statement ensures safe resource cleanup OR IDisposable.

4-Custom Exceptions?

Custom exceptions help separate business errors from system errors and improve maintainability.

5-Global Middleware Try-Catch?

- Global middleware provides a centralized and consistent way to handle errors across the application.