

Part02

1. LinkedIn article about string immutability

Mariam Ehab • You
Web Development Intern @ NTI | HTML, CSS, JavaScript
2h •

زي ما اتكلمنا في البوست إللي فات عن الفرق بين الـ Stack والـ Heap وإزاي إلـ Variables بتتخرّن.
وهي ...more

String Immutability

Why String Don't Change in Place?

The diagram illustrates the concept of string immutability. It shows two memory structures: a Stack and a Heap. In the Stack, there is a variable named "name". In the Heap, there are two strings: "Mariam" and "Marie". A pointer from the "name" variable in the Stack initially points to "Mariam" in the Heap. This pointer is crossed out with a large red "X". Instead, a new pointer is shown, labeled "after", which points to the string "Marie" in the Heap. The word "unreachable" is written in red next to the crossed-out pointer, indicating that the original string object is no longer accessible.

Karin Kheir and 1 other

Like Comment Repost Send

2. What's Enum data type, when is it used? And name three common built-in enums used frequently?

Enum (Enumeration) is a special value type in C# that represents a set of named integer constants. It provides a way to assign meaningful names to integral values, making code more readable and maintainable.

Characteristics

- **Type:** Value type (stored on stack)
- **Underlying Type:** Default is int, but can be byte, short, long and other value types.
- **Values:** Auto incremented (starting from 0) unless explicitly specified

We use it for things that have fixed set of options like

- Days of week
- Months
- Colors
- Status codes

They replace magic numbers/strings with meaningful names, provide compile-time checking, and improve code documentation. The built-in enums like DayOfWeek, ConsoleColor, and FileMode demonstrate practical applications in everyday programming scenarios that we face.

3. what are scenarios to use string Vs StringBuilder?

We use String When:

1) Storing Fixed/Constant Text

- Configuration values (connection strings, file paths)
- Error messages, labels, UI text
- Any text that won't change after creation because Strings are safe, predictable, and memory-efficient for static text.

2) Doing Single Operations

- One concatenation: fullName = firstName + " " + lastName
- One transformation: email.ToLower() or text.Trim()
- Simple formatting or replacement because One operation doesn't need StringBuilder's complexity.

3) Using it as Dictionary Keys

- Keys in Dictionary or HashSet
- Cache keys or lookup identifiers because Strings keep the same hash code, which collections need for finding items.

4) Small, Known Content

- Because the performance difference is too small to matter for tiny text.

And we use **StringBuilder** When:

1) Building Text in Loops

- Any text construction inside a for or foreach loop
- Generating reports with multiple rows
- Because StringBuilder avoids creating a lot of temporary string objects.

2) Creating Large Documents

- Because StringBuilder is much more memory-efficient for big content (no need for GC like the String).

3) Dynamic Query Building

- Constructing SQL queries with variable conditions
- Building search filters dynamically
- Because it handles conditional text building efficiently.

4) Multiple Modifications

- A lot of operations on the same text
- Chain of operations like on the assignment append then insert then replace then remove
- Because each String operation creates a new object but StringBuilder doesn't.

5. Performance-Critical Code

- StringBuilder is designed for fast text building.