Day 45/60 - Understanding yield in C#

The yield keyword in C# simplifies iterators by allowing you to return items one at a time without storing them in memory. This improves performance and reduces memory usage when working with large data collections.

Why Use yield?

Efficient Memory Usage – Items are returned lazily, avoiding unnecessary storage.
Better Performance – No need to create an intermediate list.
Simplifies Iterators – Eliminates manual state management.

Example: Using yield return

```
public static IEnumerable<int> GetNumbers()
{
 for (int i = 1; i <= 5; i++)
 {
 yield return i; // Returns one value at a time
 }
}

static void Main()
{
 foreach (var num in GetNumbers())
 {
 Console.WriteLine(num);
 }
}
```

Output:

```
1
2
3
4
5
```

Here, yield return ensures that numbers are generated on demand instead of preloading them into memory.

Example: Using yield break

Use yield break to stop iteration early.

```
public static IEnumerable<int> GetEvenNumbers()
{
 for (int i = 2; i <= 10; i += 2)
 {
 if (i > 6) yield break; // Stops iteration
 yield return i;
 }
}
```

🔷 Output:

2
4
6

When to Use yield?

Large collections where lazy evaluation saves memory.
When iterating over streams or databases efficiently.
Avoiding unnecessary list creation.

Final Thoughts

The yield keyword makes iteration simpler and more efficient. Instead of returning a full list, you can return values as needed, reducing memory usage.

#dotnet #csharp #yield #performance #lazyloading