Day 42: Getting to Know Async Streams in C#

As .NET developers, we often need to process data as it comes in—without waiting for everything to load first. With IAsyncEnumerable, you can work with asynchronous streams easily using the new await foreach syntax. It's a handy way to write clean, efficient code when dealing with data that arrives over time.

Why Use IAsyncEnumerable?
• Stream Data Efficiently:
Process items one at a time as they're produced, instead of loading an entire collection into memory.
• Keep Your App Responsive:
Ideal for I/O-bound tasks, such as reading large files or streaming API data, since you can handle each item as soon as it's available.
• Optimize Memory Usage:
By handling one element at a time, you avoid unnecessary memory pressure.

A Simple Example

Imagine you need to generate numbers asynchronously with a little delay between each:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

public class AsyncStreamDemo
{
 public async IAsyncEnumerable<int> GetNumbersAsync()
 {
 for (int i = 0; i < 10; i++)
 {
 await Task.Delay(100); // Simulate asynchronous work
 yield return i;
 }
 }

 public async Task RunAsync()
 {
 await foreach (var number in GetNumbersAsync())
 {
 Console.WriteLine($"Received: {number}");
 }
 }
}

class Program
{
 static async Task Main(string[] args)
 {
 var demo = new AsyncStreamDemo();
 await demo.RunAsync();
 }
}
```

In this code, the GetNumbersAsync method generates numbers with a small delay between each one. Using await foreach, you process each number as it's produced—keeping your app responsive and efficient.

Key Takeaways

- IAsyncEnumerable lets you work with data streams asynchronously.
- Use await foreach to handle items as they arrive.
- It's a great tool for situations where data comes in gradually, like file reading or API calls.

Try integrating asynchronous streams into your next project, and see how they can make your code more efficient and responsive.

#dotnet #csharp #asyncstreams #iasyncenumerable #awaitforeach