Day 57/60 - Understanding Source Generators in .NET

Source Generators in .NET allow compile-time code generation, reducing boilerplate and improving performance. They analyze code and generate additional code before compilation finishes.

1. What Are Source Generators?
- Introduced in .NET 5, they run at compile time to generate C# code dynamically.
- Useful for avoiding reflection and repetitive manual coding.
- Commonly used in serialization, logging, and API client generation.

2. Basic Example of a Source Generator

Create a custom generator:

```
[Generator]
public class MyGenerator : ISourceGenerator
{
 public void Execute(GeneratorExecutionContext context)
 {
 string generatedCode = @"
 namespace GeneratedCode
 {
 public static class HelloWorld
 {
 public static string GetMessage() => ""Hello from Source Generator!"";
 }
 }";

 context.AddSource("GeneratedCode.g.cs", generatedCode);
 }

 public void Initialize(GeneratorInitializationContext context) { }
}
```

3. Key Benefits of Source Generators
- Removes runtime reflection overhead (great for JSON serialization, logging, etc.).
- Speeds up application performance by shifting logic to compile time.
- Reduces manual coding efforts by automating repetitive patterns.

4. Common Use Cases
- Auto-generating API clients
- Compiling code at build time to avoid runtime processing
- Reducing reflection in serialization (System.Text.Json uses it)

Why It Matters

Source Generators help write cleaner, more efficient code by reducing manual work and runtime overhead. They are especially useful for library developers and high-performance applications.

Have you used Source Generators yet? Let me know how they've helped your development process.

#dotnet #csharp #sourcegenerators #performance #coding