Gerando código com Source Generators

@giovannibassi

www.lambda3.com.br

Giovanni Bassi

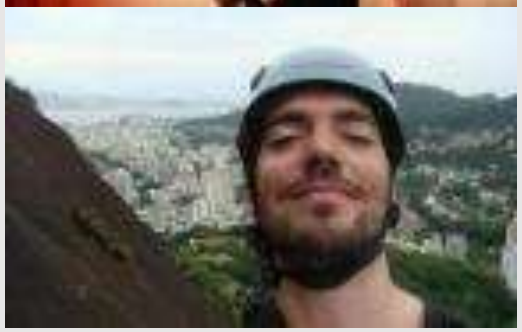
CSA na LAMBDA3

Microsoft Most Valuable Professional há onze anos

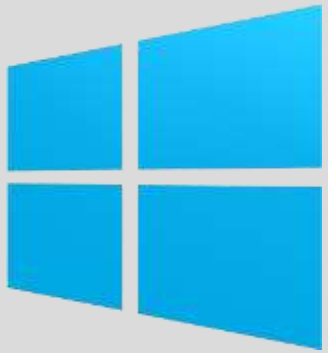
Programador

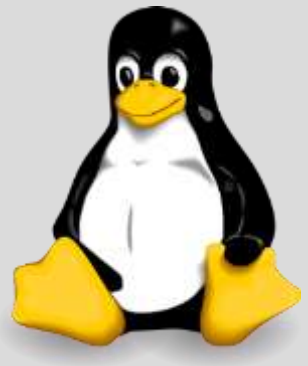
Escalador, ciclista, motociclista, baterista amador, mecânico amador

linkd.in/gbassi
twitter.com/giovannibassi
blog.lambda3.com.br
podcast.lambda3.com.br



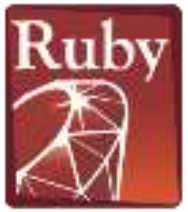

LAMBDA3






Java


Microsoft
.NET

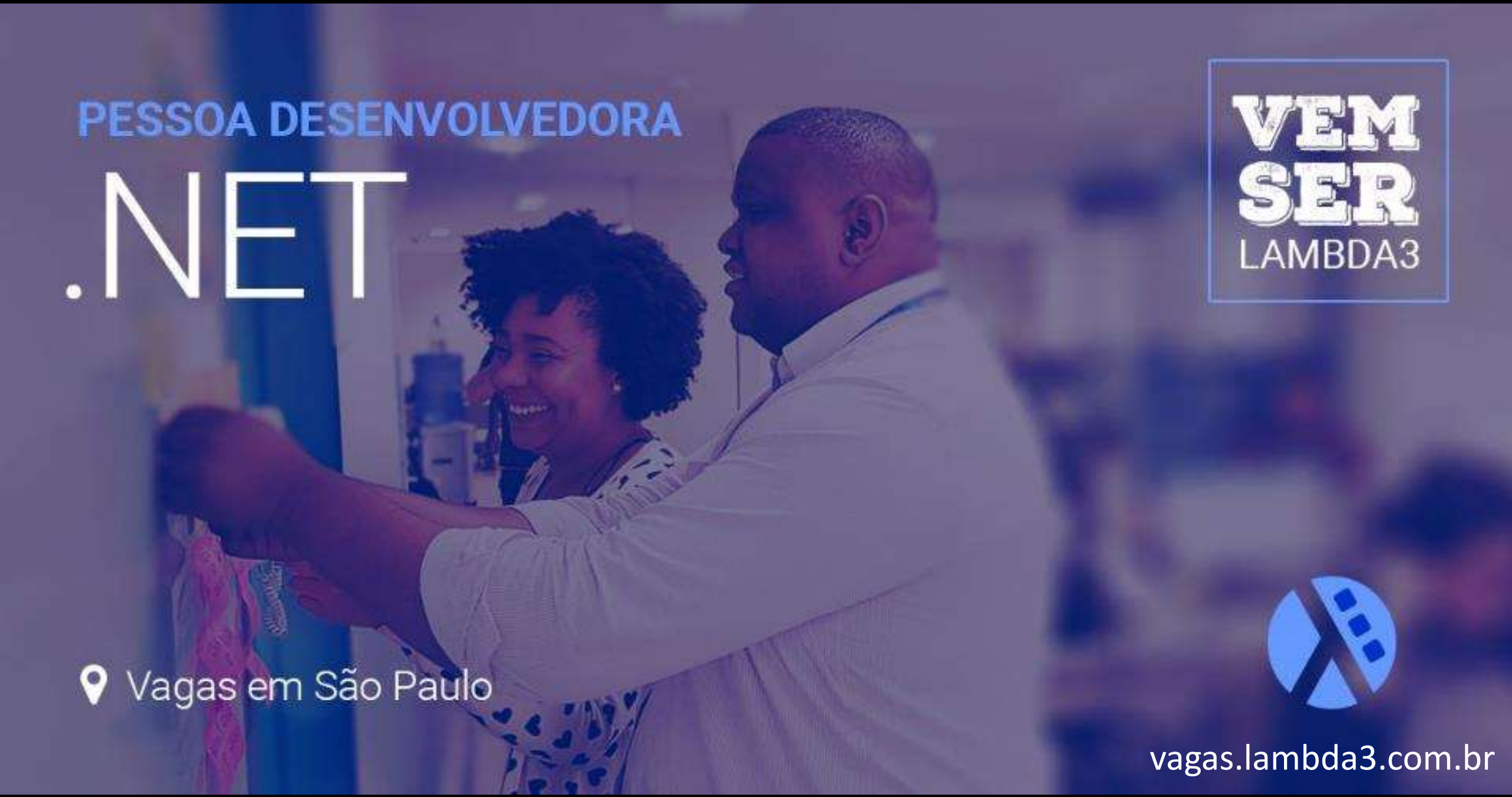

Ruby


node
js


python

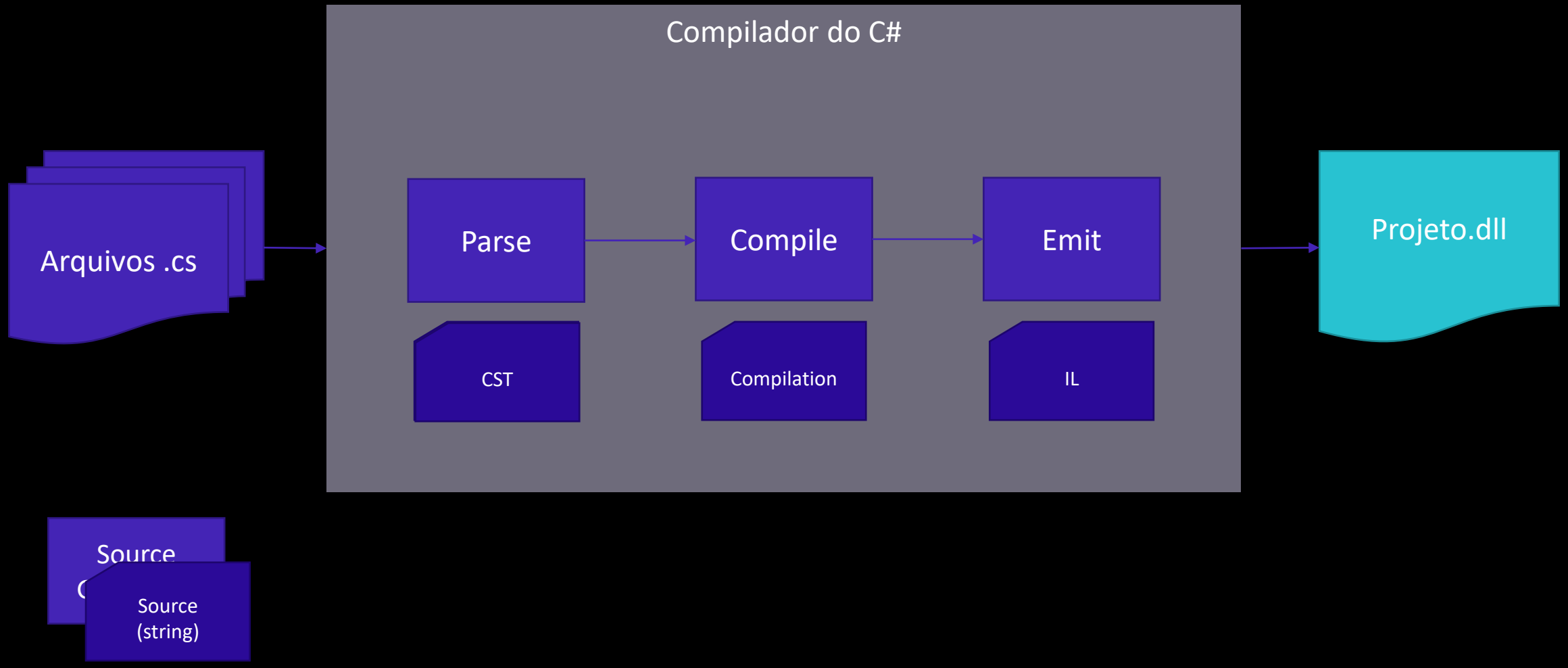

Groovy


LAMBDA3
Podcast
podcast.lambda3.com.br


PESSOA DESENVOLVEDORA
.NET
VEM SER
LAMBDA3
Vagas em São Paulo
vagas.lambda3.com.br

Geradores de código

- Geram código a partir de código existente
- Não alteram código
- Rodam durante a compilação
- Ficaram disponíveis juntos com o C# 9, apesar de não serem considerados algo da linguagem, mas do compilador


Compilador do C#
Arquivos .cs
Parse
Compile
Emit
Projeto.dll
CST
Compilation
IL
Source
Source
(string)

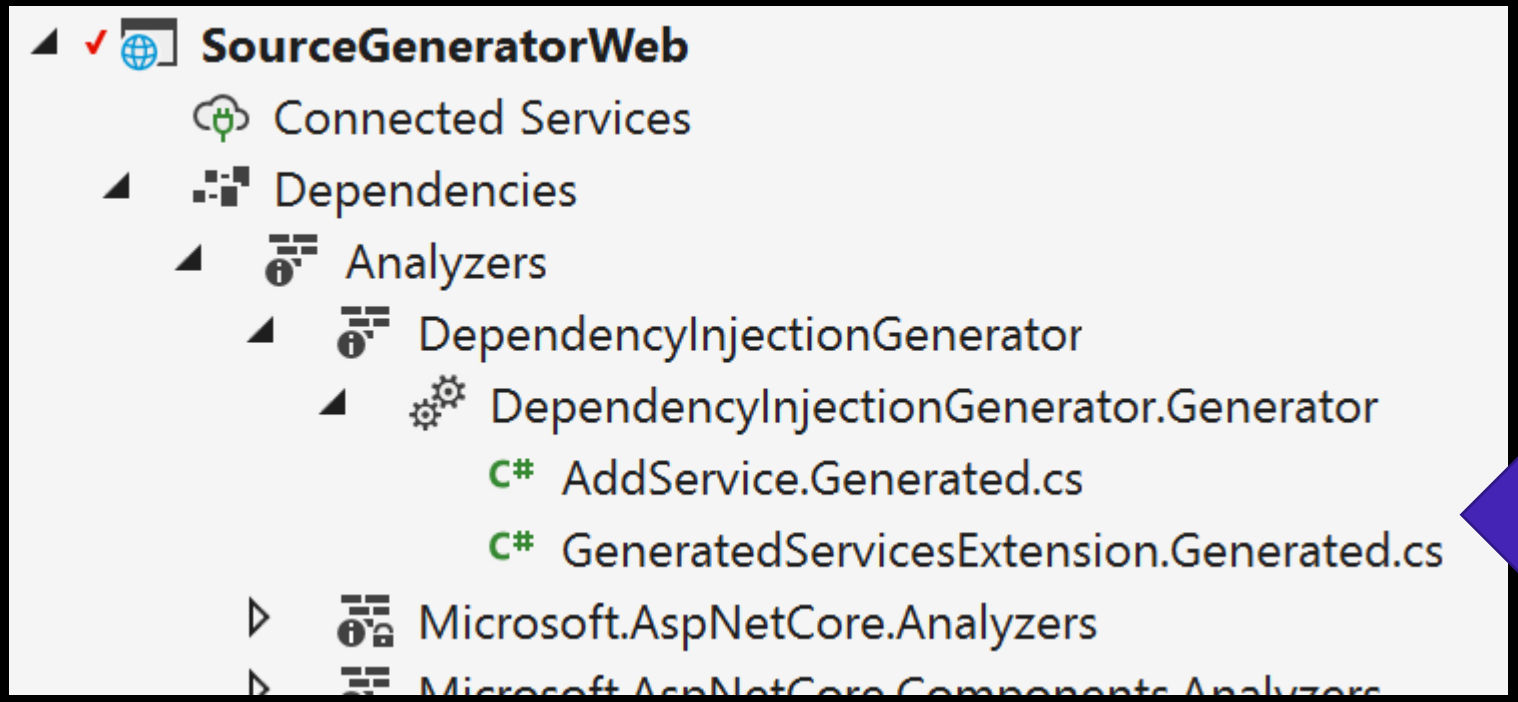
Geradores de código

- O código pode existir no disco usando a propriedade de projeto `EmitCompilerGeneratedFiles`
- A instalação é feita via NuGet (ou referenciando uma dll)
- Um projeto pode utilizar vários geradores
- Geradores podem deixar a compilação lenta

Status

- Funcionam razoavelmente bem no Visual Studio e no Rider, não funcionam no Visual Studio Code
- O desenvolvimento de geradores de código ainda não oferece muita produtividade
- A funcionalidade está bem implementada e funciona, pode ser muito útil
- Esperamos evoluções nas próximas versões do compilador

Visual Studio 16.9



Gerados por código

Ideias de geradores

- Substituir código de Reflection
  - Exemplo de mundo real: criação de um contêiner de injeção de dependência
- Processar arquivos externos
  - Exemplo de mundo real: views com Razor usam SGs no .NET 6
- Automatizar código chato
  - Exemplo de mundo real: `INotifyPropertyChanged`

Gerando código a partir de arquivos externos

```
<ItemGroup>
    <AdditionalFiles Include="settings.xml" />
</ItemGroup>

foreach (var file in context.AdditionalFiles)
{
    //
}
```

Usando classes parciais

- Você deixa uma classe parcial e usa alguma informação dela para gerar código adicional

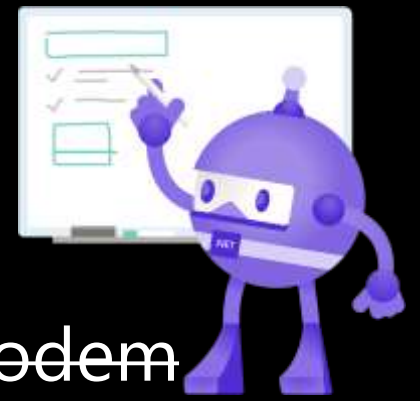
- Problema: métodos parciais devem retornar void, não podem ter parâmetros de saída, e não podem ter modificadores de acesso

Métodos parciais

```cs
// Foo.cs
partial class Foo
{
    void M()
    {
        Metodo();
    }
    partial void Metodo();
}
```

```cs
// Foo.Generated.cs
partial class Foo
{
    partial void Metodo()
    {
        //
    }
}
```

Usando classes parciais



- ~~Problema: métodos parciais devem retornar void, não podem ter parâmetros de saída, e não podem ter modificadores de acesso~~

- Solução: C# 9 permite tudo isso, desde que uma implementação exista: feito exatamente por causa de source generators!

Métodos parciais

```
// Foo.cs
void M()
{
    if (TryConvert("3", out var value))
    {
        //
    }
}
public partial bool TryConvert(
    string text, out int value);
```

```
// Foo.Generated.cs
public partial bool TryConvert(
    string text, out int value)
{
    //
}
```

Escrevendo um gerador

- Muito parecido com a escrita de um analisador de código (Roslyn Analyzers)
- Você atua lendo o resultado da compilação, inspecionando a CST, símbolos etc, e gerando novo código
- Não é algo trivial para quem está começando, mas é um desafio interessante

Depurando um gerador

- Evite tentar depurar: escreva testes de unidade
- Se precisar depurar, veja o projeto Kittitas: github.com/chsienki/kittitas

Demo

Rolsyn Source Generators

Recursos

Post de apresentação de SG

bit.ly/post-sg

Projeto de exemplo

github.com/giggio-samples/SourceGeneratorWeb

Playground de SG

sourcegen.dev

Projeto real de SG para DI

github.com/giggio/sourceinject/

bit.ly/palestrasg

Obrigado!


@giovannibassi

www.lambda3.com.br