.NET Conf China 2023

2023/12/16
09:30 - 18:00

中国・北京

.NET中文社区

中国 · 北京

.NET Conf China 2023

探索 Interceptor 的魔力

李卫涵 WeihanLi
iHerb 后端开发工程师
微软MVP/amazingdotnet 博主

Microsoft® Most Valuable Professional

目录

AGENDA

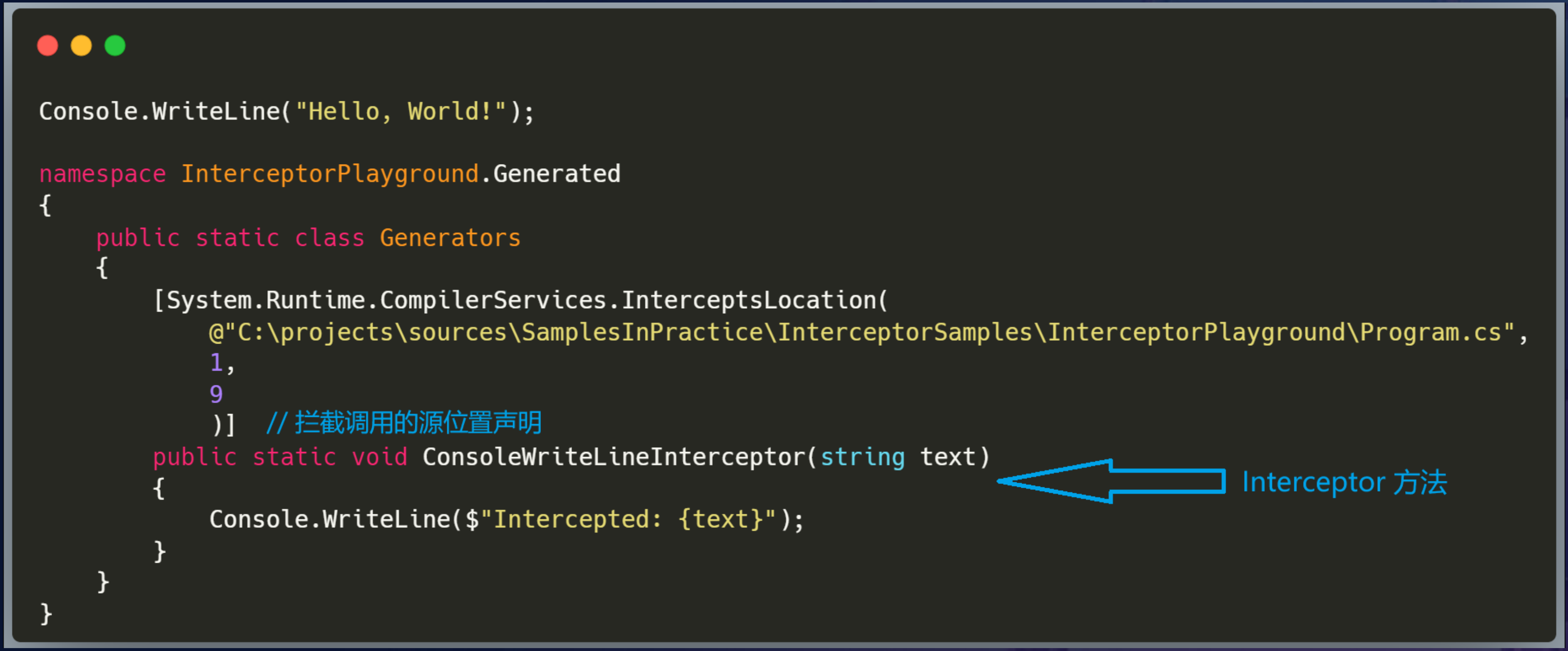
What’ s Interceptor

拦截器 是一种可以在 ***编译时*** 以 ***声明方式*** 用对其自身的调用来替换对可拦截方法的调用的 ***方法***。

An *interceptor* is a method which can declaratively substitute a call to an interceptable method with a call to itself at compile time.

方法	编译时
声明式	调用替换

How it intercepts

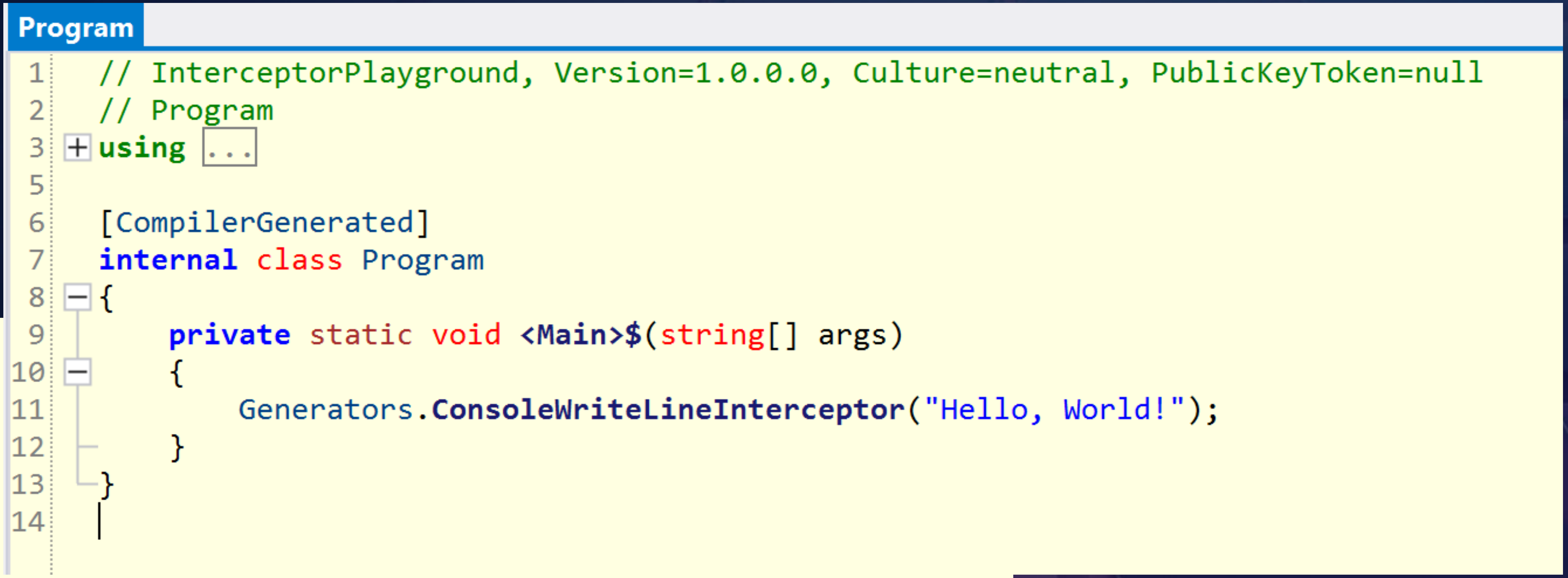


```
Console.WriteLine("Hello, World!");

namespace InterceptorPlayground.Generated
{
    public static class Generators
    {
        [System.Runtime.CompilerServices.InterceptsLocation(
            @"C:\projects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\Program.cs",
            1,
            9
            )]  //拦截调用的源位置声明
        public static void ConsoleWriteLineInterceptor(string text)
        {
            Console.WriteLine($"Intercepted: {text}");
        }
    }
}
```

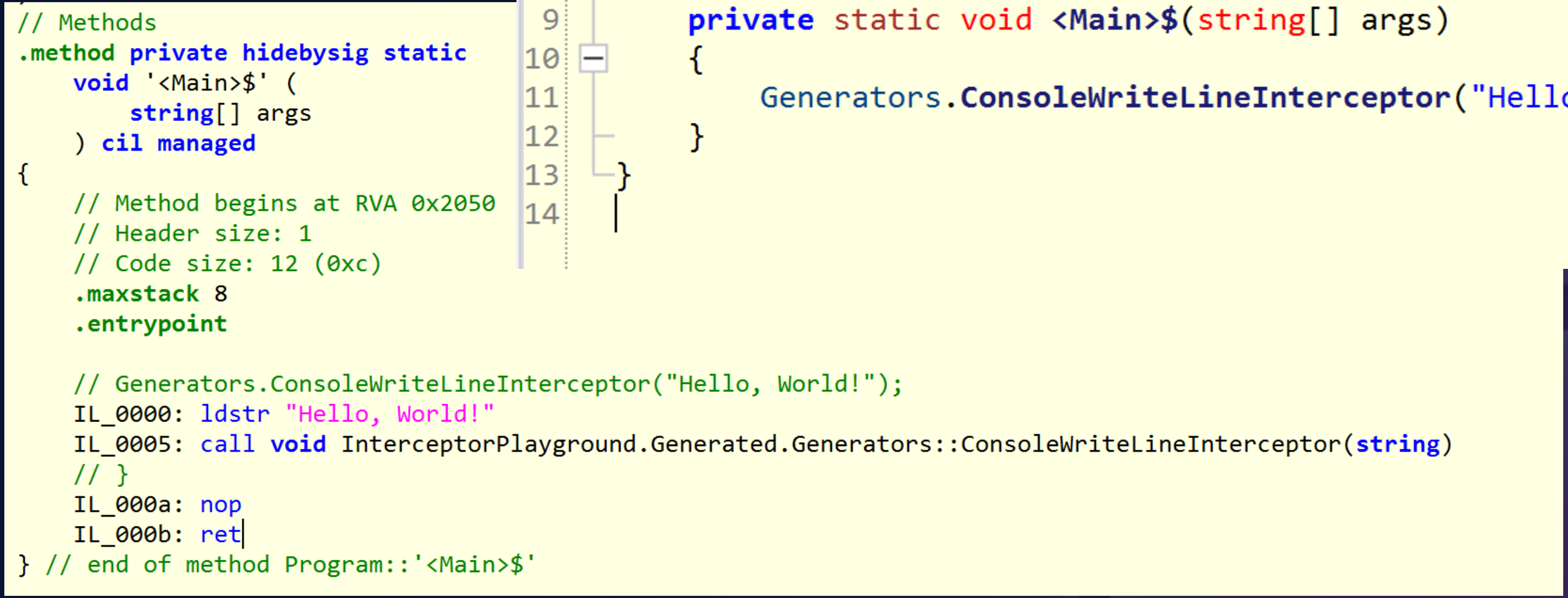
This substitution occurs by having the interceptor declare the source locations of the calls that it intercepts.

这种替换是通过让拦截器声明它拦截的调用的源位置来实现的。

What happens when compile


Program

```
// InterceptorPlayground, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// Program
using ...

[CompilerGenerated]
internal class Program
{
    private static void <Main>$(string[] args)
    {
        Generators.ConsoleWriteLineInterceptor("Hello, World!");
    }
}
```



```
// Methods
.method private hidebysig static
    void '<Main>$' (
        string[] args
    ) cil managed
{
    // Method begins at RVA 0x2050
    // Header size: 1
    // Code size: 12 (0xc)
    .maxstack 8
    .entrypoint

    // Generators.ConsoleWriteLineInterceptor("Hello, World!");
    IL_0000: ldstr "Hello, World!"
    IL_0005: call void InterceptorPlayground.Generated.Generators::ConsoleWriteLineInterceptor(string)
    // }
    IL_000a: nop
    IL_000b: ret
} // end of method Program::'<Main>$'
```

Why Interceptor

性 能

AOT

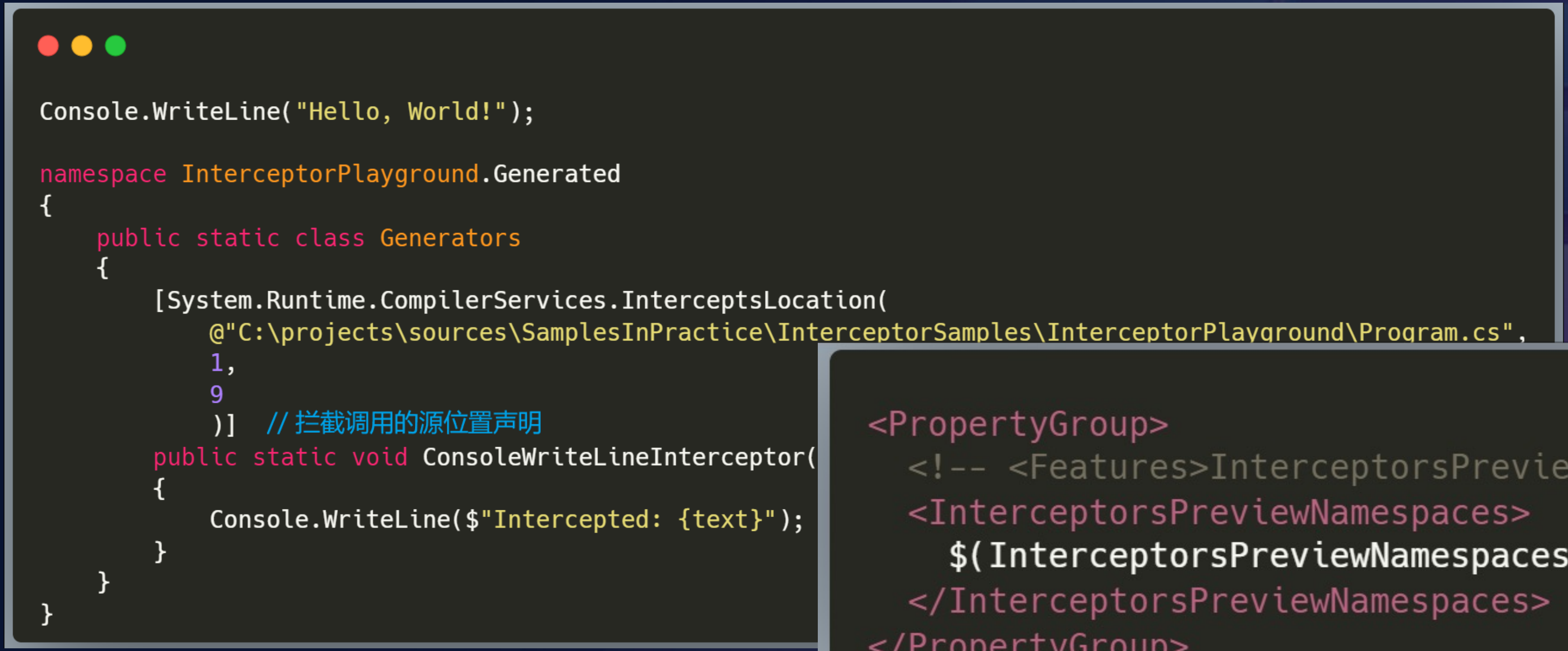
临时测试

AOP

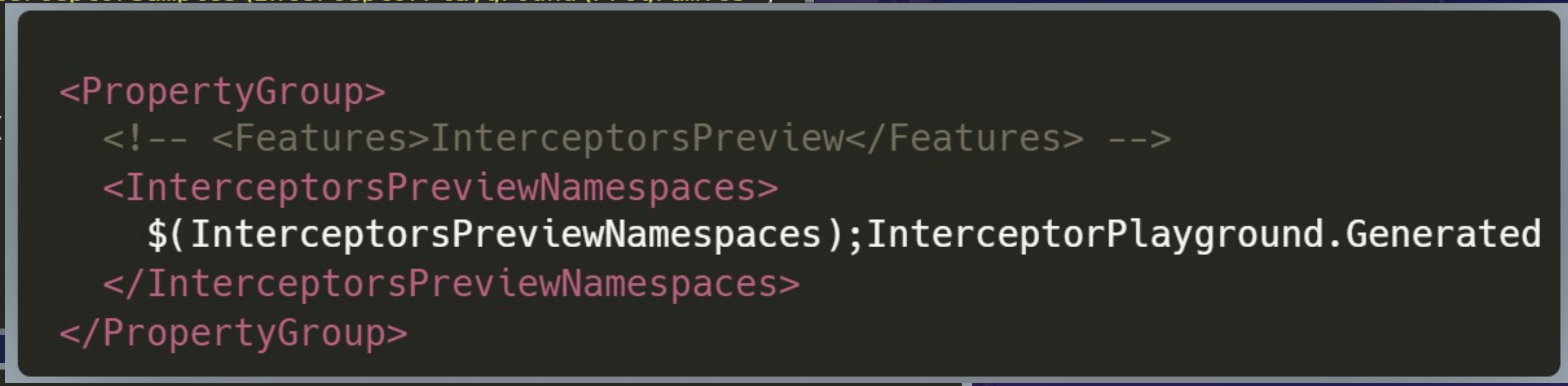
.NET中文社区

.NET Conf China 2023
中国 • 北京

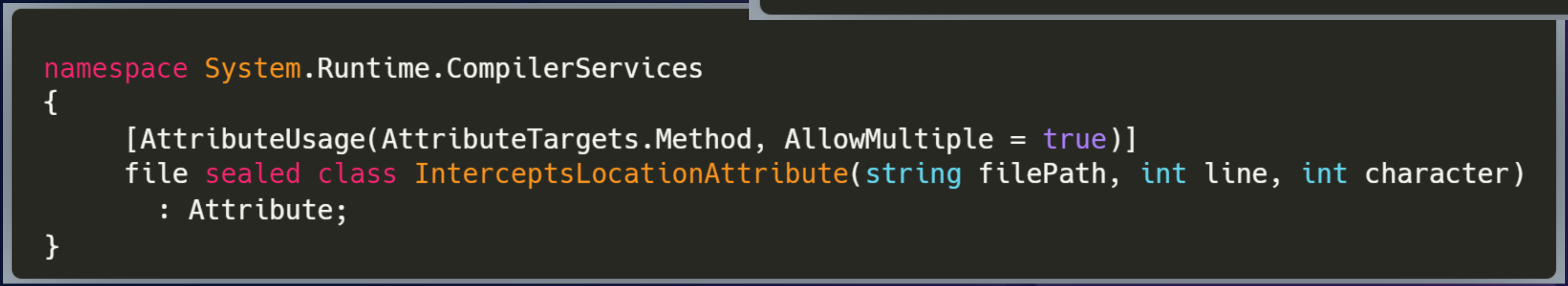
How to implement an interceptor



```
Console.WriteLine("Hello, World!");

namespace InterceptorPlayground.Generated
{
    public static class Generators
    {
        [System.Runtime.CompilerServices.InterceptsLocation(
            @"C:\projects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\Program.cs",
            1,
            9
            )]  // 拦截调用的源位置声明
        public static void ConsoleWriteLineInterceptor(
        {
            Console.WriteLine($"Intercepted: {text}");
        }
    }
}
```



```
<PropertyGroup>
  <!-- <Features>InterceptorsPreview</Features> -->
  <InterceptorsPreviewNamespaces>
    $(InterceptorsPreviewNamespaces);InterceptorPlayground.Generated
  </InterceptorsPreviewNamespaces>
</PropertyGroup>
```



```
namespace System.Runtime.CompilerServices
{
    [AttributeUsage(AttributeTargets.Method, AllowMultiple = true)]
    file sealed class InterceptsLocationAttribute(string filePath, int line, int character)
      : Attribute;
}
```

Must && Limitation

只拦截普通方法调用

只支持本地代码
不支持类库内部调用

方法签名应一致

只支持 C#
不支持 VB

定义在非泛型类型中

Not Supported

Constructor
构造方法

Property
属性

Operator
操作符

Delegate
委托

Local Function
本地函数

Intercepts instance method

```
[System.Runtime.CompilerServices.InterceptsLocation(
    @"C:\projects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\Program.cs",
    4,
    24
    )]
public static string InstanceMethodInterceptor(this Test test, int age)
{
    return $"Intercepted: {test.Hello(age)}";
}
```

```
var test = new Test();
Console.WriteLine(test.Hello(10));

public class Test
{
    public string Hello(int age)
      => $"Hello, I'm {age} years old";
}
```

Intercepts extension method

```
[System.Runtime.CompilerServices.InterceptsLocation(
    @"C:\projects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\Program.cs",
    5,
    24
    )]
public static int ExtensionMethodInterceptor(this Test test, int age)
{
    return ++age;
}
```

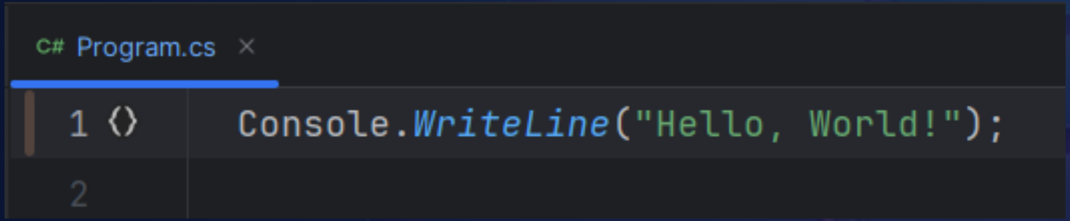
```
var test = new Test();
Console.WriteLine(test.AgePlusPlus(10));

public class Test
{
}
public static class Extensions
{
    public static int AgePlusPlus(this Test test, int age)
      => age++;
}
```

Multiple Interception

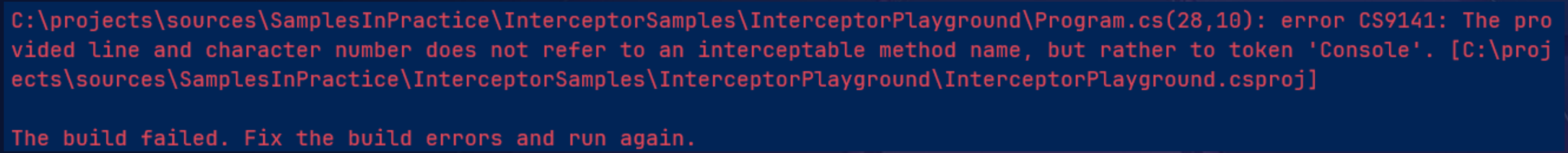
```
Console.WriteLine("Amazing Interceptor");
Console.WriteLine("Amazing .NET Conf China 2023");

[System.Runtime.CompilerServices.InterceptsLocation(
    @"C:\projects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\Program.cs",
    8,
    9
    )]
[System.Runtime.CompilerServices.InterceptsLocation(
    @"C:\projects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\Program.cs",
    7,
    9
    )]
public static void AmazingConsoleWriteLineInterceptor(string? text)
{
    Console.WriteLine($"Amazing .NET, {text}");
}
```

```
Amazing .NET, Amazing Interceptor
Amazing .NET, Amazing .NET Conf China 2023
```

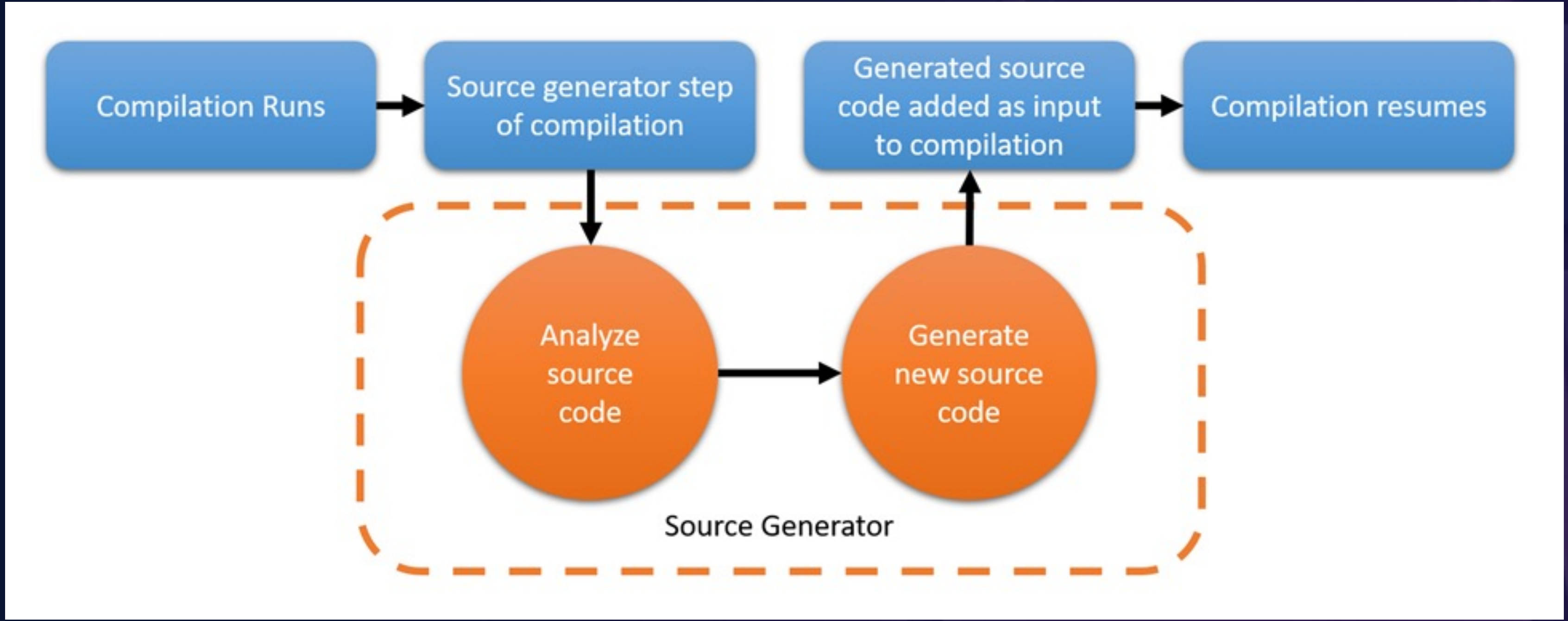
Source Generator Integration



```csharp
Console.WriteLine("Hello, World!");
```



```
C:\projects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\Program.cs(28,10): error CS9141: The pro
vided line and character number does not refer to an interceptable method name, but rather to token 'Console'. [C:\proj
ects\sources\SamplesInPractice\InterceptorSamples\InterceptorPlayground\InterceptorPlayground.csproj]

The build failed. Fix the build errors and run again.
```



Source Generator Integration

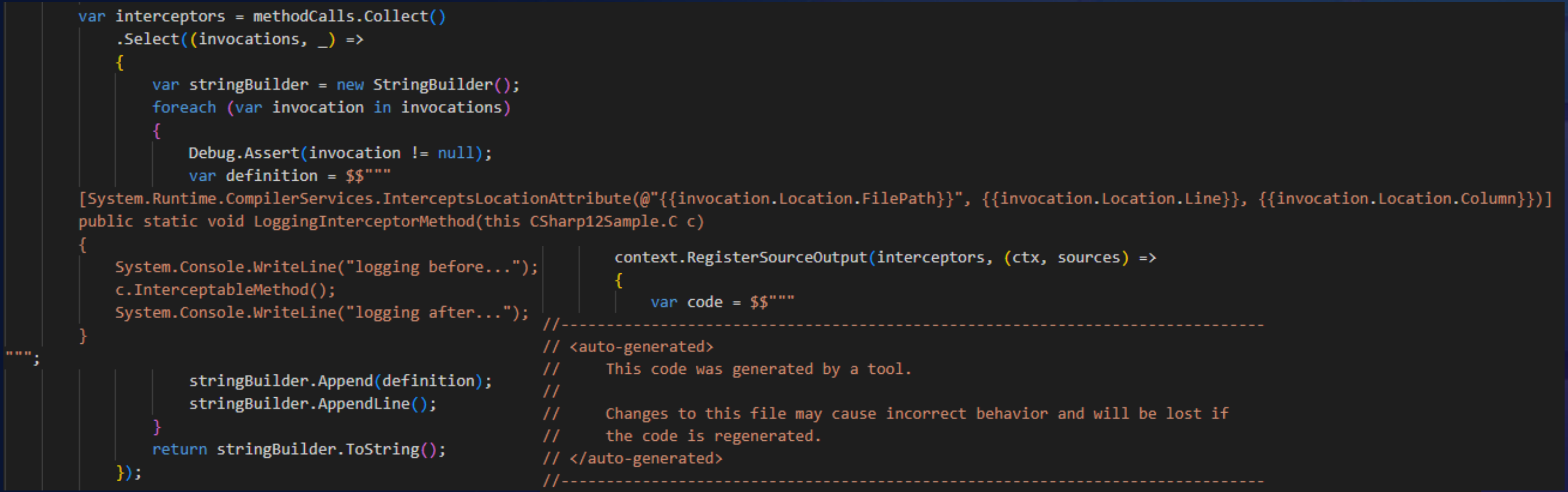
```
private static (string, int, int) GetLocation(IInvocationOperation operation)
{
    // The invocation expression consists of two properties:
    // - Expression: which is a `MemberAccessExpressionSyntax` that represents the method being invoked.
    // - ArgumentList: the list of arguments being invoked.
    // Here, we resolve the `MemberAccessExpressionSyntax` to get the location of the method being invoked.
    var memberAccessorExpression = ((MemberAccessExpressionSyntax)((InvocationExpressionSyntax)operation.Syntax).Expression);
    // The `MemberAccessExpressionSyntax` in turn includes three properties:
    // - Expression: the expression that is being accessed.
    // - OperatorToken: the operator token, typically the dot separate.
    // - Name: the name of the member being accessed, typically `MapGet` or `MapPost`, etc.
    // Here, we resolve the `Name` to extract the location of the method being invoked.
    var invocationNameSpan = memberAccessorExpression.Name.Span;
    // Resolve LineSpan associated with the name span so we can resolve the line and character number.
    var lineSpan = operation.Syntax.SyntaxTree.GetLineSpan(invocationNameSpan);
    // Resolve the filepath of the invocation while accounting for source mapped paths.
    var filePath = operation.Syntax.SyntaxTree.GetInterceptorFilePath(operation.SemanticModel?.Compilation.Options.SourceReferenceResolver);
    // LineSpan.LinePosition is 0-indexed, but we want to display 1-indexed line and character numbers in the interceptor attribute.
    return (filePath, lineSpan.StartLinePosition.Line + 1, lineSpan.StartLinePosition.Character + 1);
}
```

```
file static class Extensions
{
    // https://github.com/dotnet/roslyn/blob/main/docs/features/interceptors.md
    // Utilize the same logic used by the interceptors API for resolving the source mapped
    internal static string GetInterceptorFilePath(this SyntaxTree tree, SourceReferenceResolver? resolver) =>
        resolver?.NormalizePath(tree.FilePath, baseFilePath: null) ?? tree.FilePath;
}
```

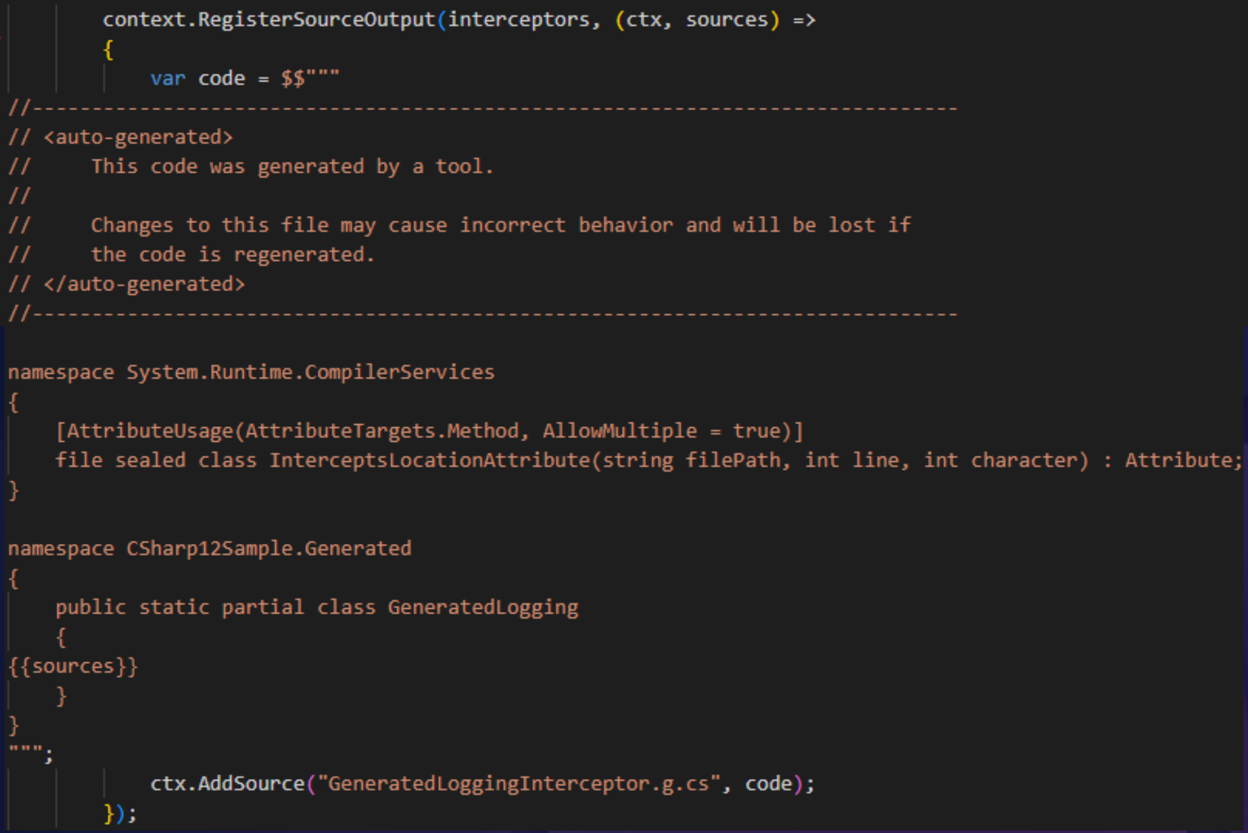
Source Generator Integration

```
[Generator(LanguageNames.CSharp)]
public sealed class LoggingGenerator : IIncrementalGenerator
{
    public void Initialize(IncrementalGeneratorInitializationContext context)
    {
        var methodCalls = context.SyntaxProvider.CreateSyntaxProvider(
            predicate: static (node, _) => node is
                InvocationExpressionSyntax
                {
                    Expression: MemberAccessExpressionSyntax
                    {
                        Name:
                        {
                            Identifier:
                            {
                                ValueText: "InterceptableMethod"
                            }
                        }
                    }
                }
            ,
            transform: static (context, token) =>
            {
                var operation = context.SemanticModel.GetOperation(context.Node, token);
                if (operation is IInvocationOperation targetOperation
                    )
                {
                    return new InterceptInvocation(targetOperation);
                }
                return null;
            })
            .Where(static invocation => invocation != null);
            // ...
    }
}
```

Source Generator Integration



```
        var interceptors = methodCalls.Collect()
            .Select((invocations, _) =>
            {
                var stringBuilder = new StringBuilder();
                foreach (var invocation in invocations)
                {
                    Debug.Assert(invocation != null);
                    var definition = $$"""
        [System.Runtime.CompilerServices.InterceptsLocationAttribute(@"{{invocation.Location.FilePath}}", {{invocation.Location.Line}}, {{invocation.Location.Column}})]
        public static void LoggingInterceptorMethod(this CSharp12Sample.C c)
        {
            System.Console.WriteLine("logging before...");
            c.InterceptableMethod();
            System.Console.WriteLine("logging after...");
        }
""";
                    stringBuilder.Append(definition);
                    stringBuilder.AppendLine();
                }
                return stringBuilder.ToString();
            });
```

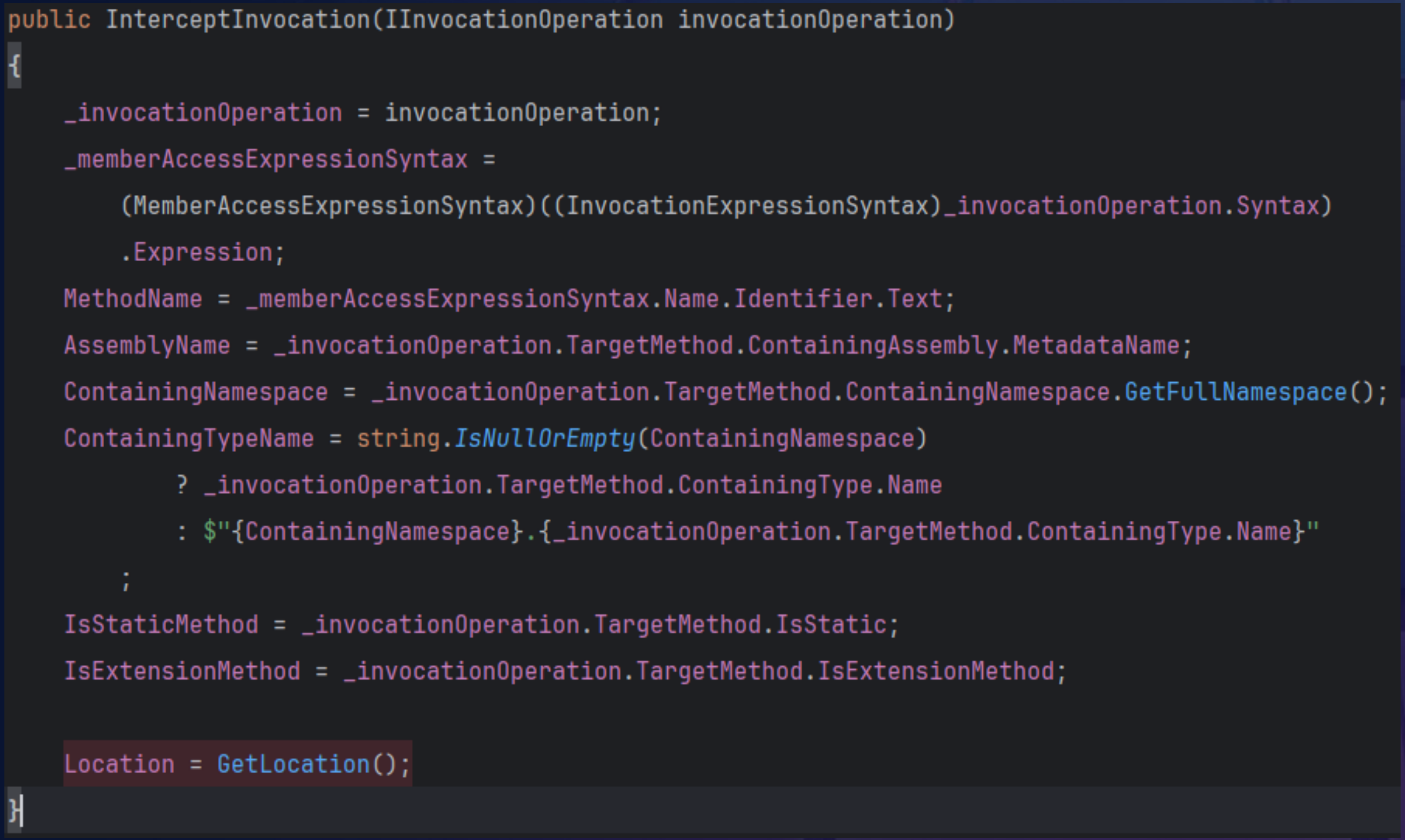


```
        context.RegisterSourceOutput(interceptors, (ctx, sources) =>
        {
            var code = $$"""
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace System.Runtime.CompilerServices
{
    [AttributeUsage(AttributeTargets.Method, AllowMultiple = true)]
    file sealed class InterceptsLocationAttribute(string filePath, int line, int character) : Attribute;
}

namespace CSharp12Sample.Generated
{
    public static partial class GeneratedLogging
    {
{{sources}}
    }
}
""";
            ctx.AddSource("GeneratedLoggingInterceptor.g.cs", code);
        });
```

.NET中文社区

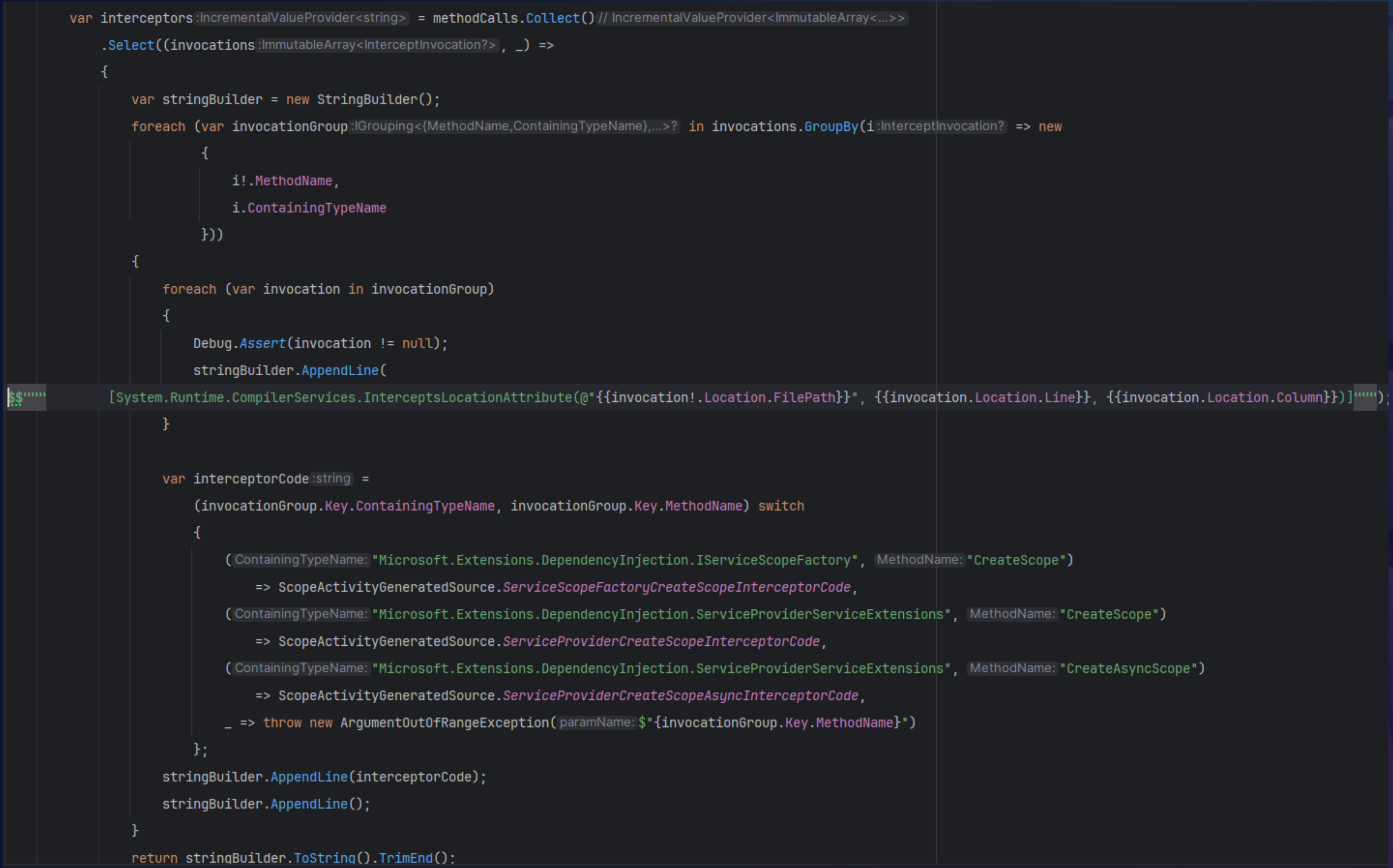
.NET Conf China 2023
中国 • 北京

Source Generator Integration



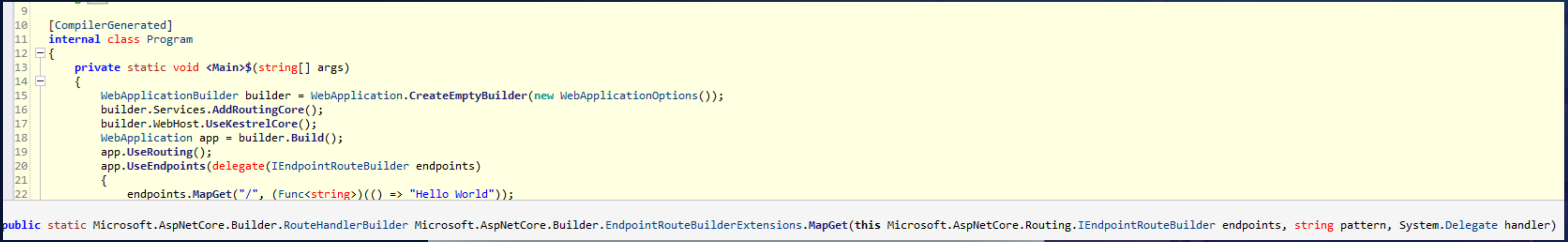
```
public InterceptInvocation(IInvocationOperation invocationOperation)
{
    _invocationOperation = invocationOperation;
    _memberAccessExpressionSyntax =
        (MemberAccessExpressionSyntax)((InvocationExpressionSyntax)_invocationOperation.Syntax)
        .Expression;
    MethodName = _memberAccessExpressionSyntax.Name.Identifier.Text;
    AssemblyName = _invocationOperation.TargetMethod.ContainingAssembly.MetadataName;
    ContainingNamespace = _invocationOperation.TargetMethod.ContainingNamespace.GetFullNamespace();
    ContainingTypeName = string.IsNullOrEmpty(ContainingNamespace)
            ? _invocationOperation.TargetMethod.ContainingType.Name
            : $"{ContainingNamespace}.{_invocationOperation.TargetMethod.ContainingType.Name}"
        ;
    IsStaticMethod = _invocationOperation.TargetMethod.IsStatic;
    IsExtensionMethod = _invocationOperation.TargetMethod.IsExtensionMethod;

    Location = GetLocation();
}
```

Source Generator Integration



```
    var interceptors :IncrementalValueProvider<string> = methodCalls.Collect() // IncrementalValueProvider<ImmutableArray<...>>
        .Select((invocations :ImmutableArray<InterceptInvocation?> , _) =>
        {
            var stringBuilder = new StringBuilder();
            foreach (var invocationGroup :IGrouping<{MethodName,ContainingTypeName},...>? in invocations.GroupBy(i :InterceptInvocation? => new
                     {
                         i!.MethodName,
                         i.ContainingTypeName
                     }))
            {
                foreach (var invocation in invocationGroup)
                {
                    Debug.Assert(invocation != null);
                    stringBuilder.AppendLine(
$$"""        [System.Runtime.CompilerServices.InterceptsLocationAttribute(@"{{invocation!.Location.FilePath}}", {{invocation.Location.Line}}, {{invocation.Location.Column}})]""");
                }

                var interceptorCode :string =
                    (invocationGroup.Key.ContainingTypeName, invocationGroup.Key.MethodName) switch
                    {
                        ( ContainingTypeName: "Microsoft.Extensions.DependencyInjection.IServiceScopeFactory", MethodName: "CreateScope")
                            => ScopeActivityGeneratedSource.ServiceScopeFactoryCreateScopeInterceptorCode,
                        ( ContainingTypeName: "Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions", MethodName: "CreateScope")
                            => ScopeActivityGeneratedSource.ServiceProviderCreateScopeInterceptorCode,
                        ( ContainingTypeName: "Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions", MethodName: "CreateAsyncScope")
                            => ScopeActivityGeneratedSource.ServiceProviderCreateScopeAsyncInterceptorCode,
                        _ => throw new ArgumentOutOfRangeException( paramName: $"{invocationGroup.Key.MethodName}")
                    };
                stringBuilder.AppendLine(interceptorCode);
                stringBuilder.AppendLine();
            }
            return stringBuilder.ToString().TrimEnd();
```

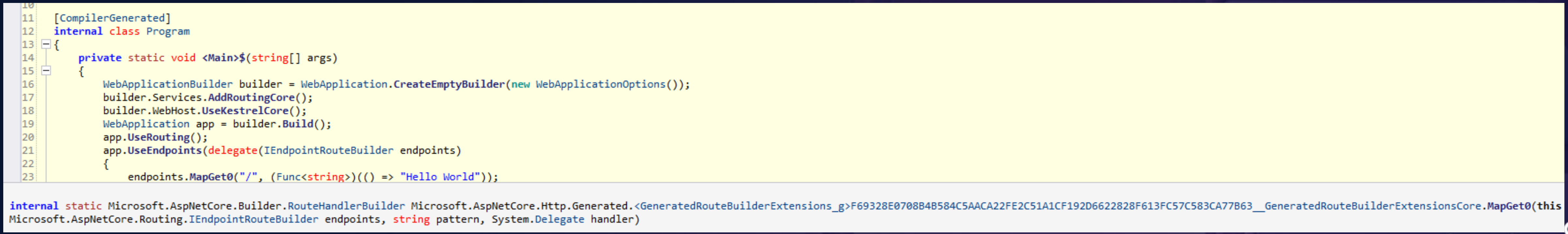
Minimal API AOT



```csharp
[CompilerGenerated]
internal class Program
{
    private static void <Main>$(string[] args)
    {
        WebApplicationBuilder builder = WebApplication.CreateEmptyBuilder(new WebApplicationOptions());
        builder.Services.AddRoutingCore();
        builder.WebHost.UseKestrelCore();
        WebApplication app = builder.Build();
        app.UseRouting();
        app.UseEndpoints(delegate(IEndpointRouteBuilder endpoints)
        {
            endpoints.MapGet("/", (Func<string>)(() => "Hello World"));
```

```
public static Microsoft.AspNetCore.Builder.RouteHandlerBuilder Microsoft.AspNetCore.Builder.EndpointRouteBuilderExtensions.MapGet(this Microsoft.AspNetCore.Routing.IEndpointRouteBuilder endpoints, string pattern, System.Delegate handler)
```

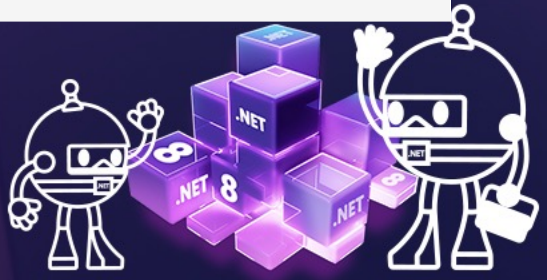
```
dotnet build -p PublishAot=false # without AOT
```



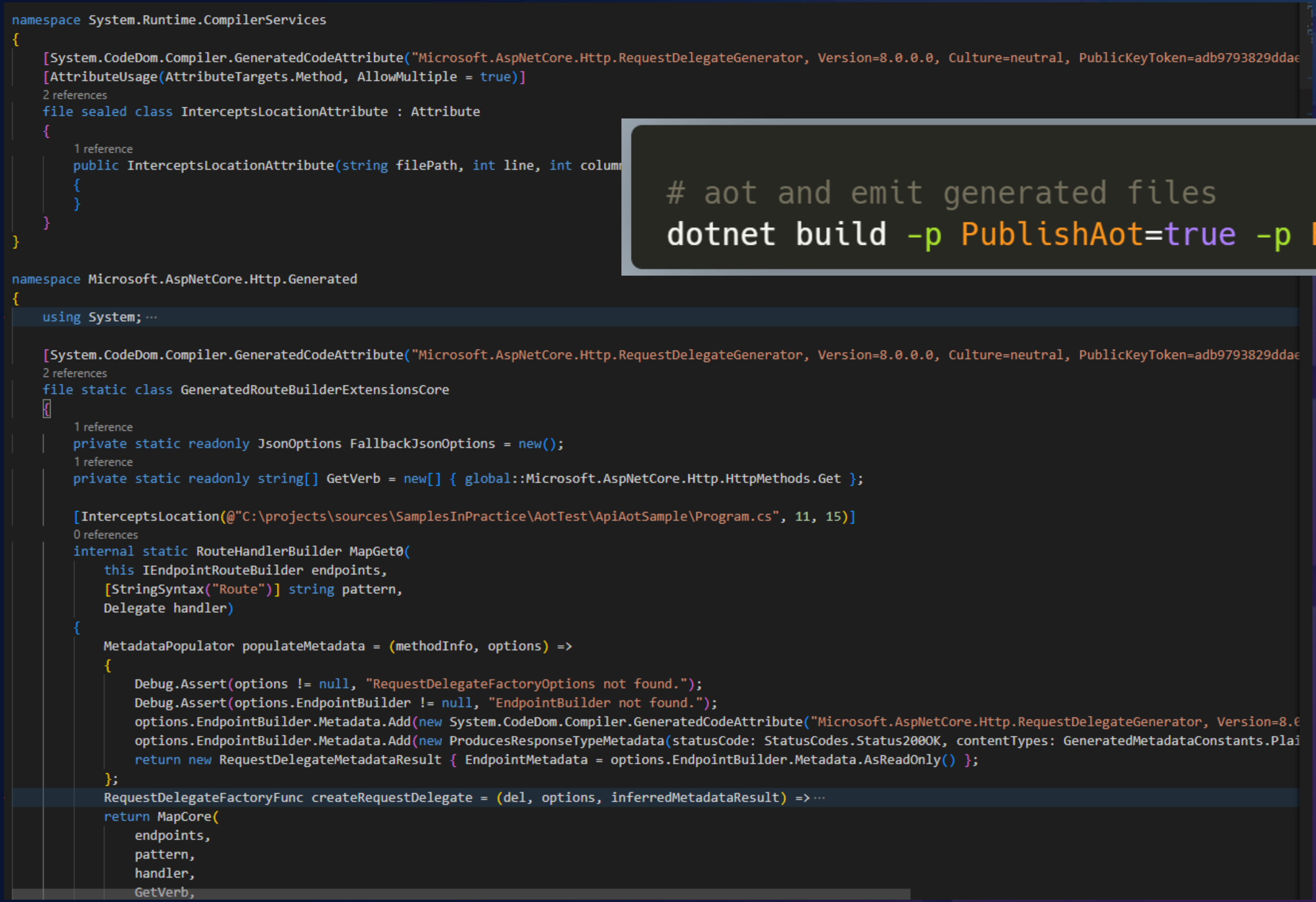
```csharp
[CompilerGenerated]
internal class Program
{
    private static void <Main>$(string[] args)
    {
        WebApplicationBuilder builder = WebApplication.CreateEmptyBuilder(new WebApplicationOptions());
        builder.Services.AddRoutingCore();
        builder.WebHost.UseKestrelCore();
        WebApplication app = builder.Build();
        app.UseRouting();
        app.UseEndpoints(delegate(IEndpointRouteBuilder endpoints)
        {
            endpoints.MapGet0("/", (Func<string>)(() => "Hello World"));
```

```
internal static Microsoft.AspNetCore.Builder.RouteHandlerBuilder Microsoft.AspNetCore.Http.Generated.<GeneratedRouteBuilderExtensions_g>F69328E0708B4B584C5AACA22FE2C51A1CF192D6622828F613FC57C583CA77B63__GeneratedRouteBuilderExtensionsCore.MapGet0(this
Microsoft.AspNetCore.Routing.IEndpointRouteBuilder endpoints, string pattern, System.Delegate handler)
```

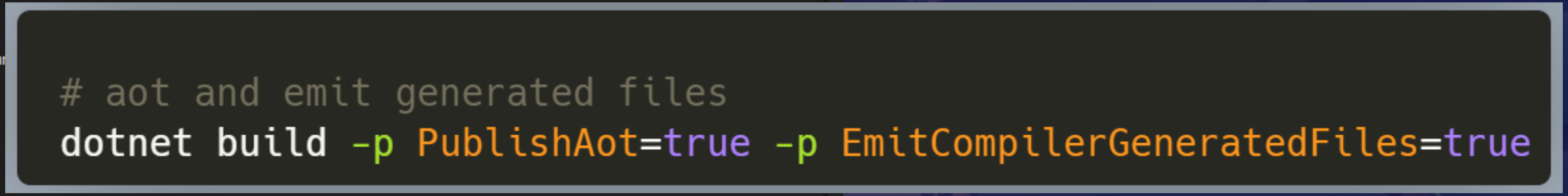
```
dotnet build -p PublishAot=true # with AOT
```



Minimal API AOT

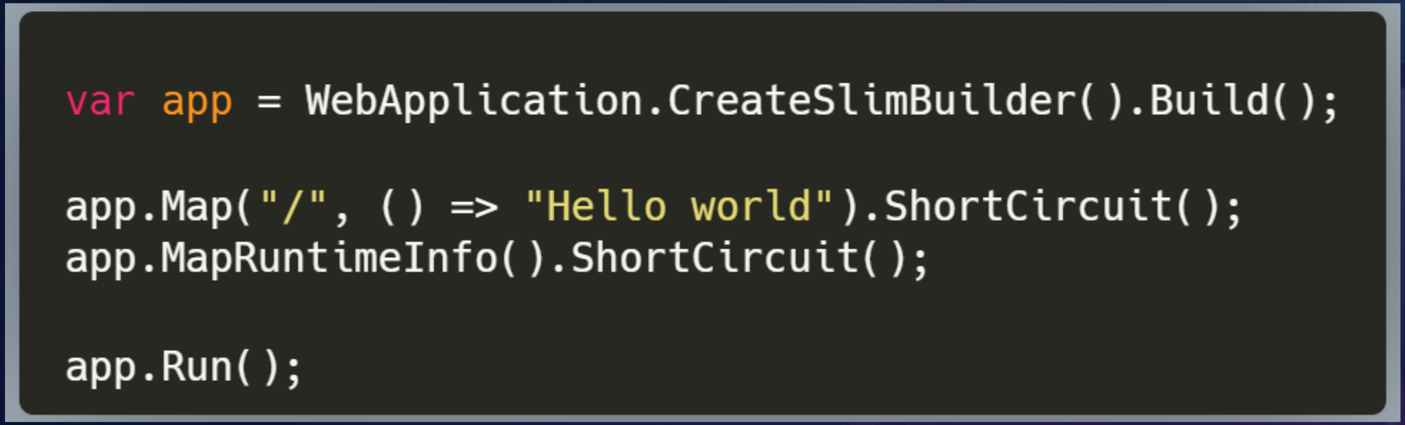


```
namespace System.Runtime.CompilerServices
{
    [System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.AspNetCore.Http.RequestDelegateGenerator, Version=8.0.0.0, Culture=neutral, PublicKeyToken=adb9793829ddae
    [AttributeUsage(AttributeTargets.Method, AllowMultiple = true)]
    2 references
    file sealed class InterceptsLocationAttribute : Attribute
    {
        1 reference
        public InterceptsLocationAttribute(string filePath, int line, int colum
        {
        }
    }
}

namespace Microsoft.AspNetCore.Http.Generated
{
    using System; ···

    [System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.AspNetCore.Http.RequestDelegateGenerator, Version=8.0.0.0, Culture=neutral, PublicKeyToken=adb9793829ddae
    2 references
    file static class GeneratedRouteBuilderExtensionsCore
    {
        1 reference
        private static readonly JsonOptions FallbackJsonOptions = new();
        1 reference
        private static readonly string[] GetVerb = new[] { global::Microsoft.AspNetCore.Http.HttpMethods.Get };

        [InterceptsLocation(@"C:\projects\sources\SamplesInPractice\AotTest\ApiAotSample\Program.cs", 11, 15)]
        0 references
        internal static RouteHandlerBuilder MapGet0(
            this IEndpointRouteBuilder endpoints,
            [StringSyntax("Route")] string pattern,
            Delegate handler)
        {
            MetadataPopulator populateMetadata = (methodInfo, options) =>
            {
                Debug.Assert(options != null, "RequestDelegateFactoryOptions not found.");
                Debug.Assert(options.EndpointBuilder != null, "EndpointBuilder not found.");
                options.EndpointBuilder.Metadata.Add(new System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.AspNetCore.Http.RequestDelegateGenerator, Version=8.0
                options.EndpointBuilder.Metadata.Add(new ProducesResponseTypeMetadata(statusCode: StatusCodes.Status200OK, contentTypes: GeneratedMetadataConstants.Plai
                return new RequestDelegateMetadataResult { EndpointMetadata = options.EndpointBuilder.Metadata.AsReadOnly() };
            };
            RequestDelegateFactoryFunc createRequestDelegate = (del, options, inferredMetadataResult) =>···
            return MapCore(
                endpoints,
                pattern,
                handler,
                GetVerb,
```

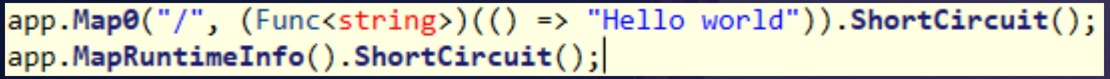


```
# aot and emit generated files
dotnet build -p PublishAot=true -p EmitCompilerGeneratedFiles=true
```

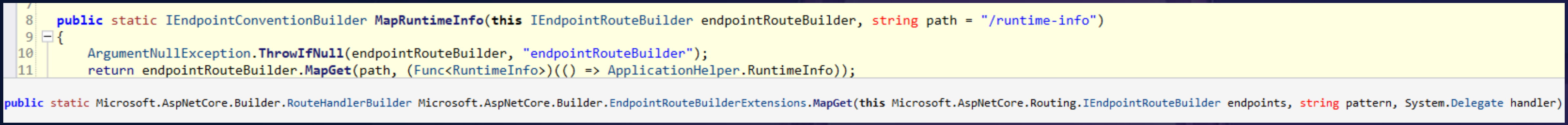
Minimal API AOT



```
var app = WebApplication.CreateSlimBuilder().Build();

app.Map("/", () => "Hello world").ShortCircuit();
app.MapRuntimeInfo().ShortCircuit();

app.Run();
```



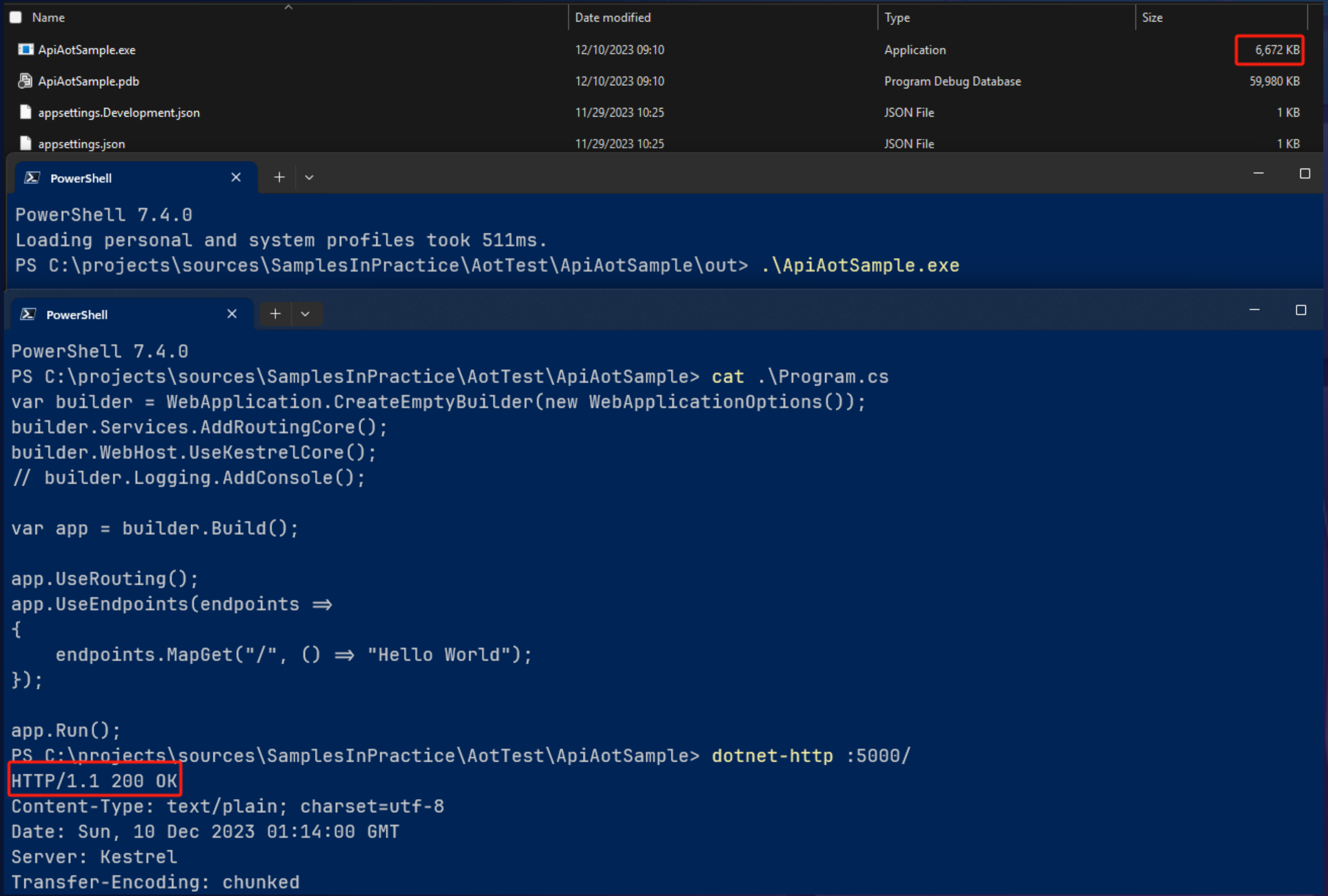
```
app.Map0("/", (Func<string>)(() => "Hello world")).ShortCircuit();
app.MapRuntimeInfo().ShortCircuit();
```



```
public static IEndpointConventionBuilder MapRuntimeInfo(this IEndpointRouteBuilder endpointRouteBuilder, string path = "/runtime-info")
{
    ArgumentNullException.ThrowIfNull(endpointRouteBuilder, "endpointRouteBuilder");
    return endpointRouteBuilder.MapGet(path, (Func<RuntimeInfo>)(() => ApplicationHelper.RuntimeInfo));
```

```
public static Microsoft.AspNetCore.Builder.RouteHandlerBuilder Microsoft.AspNetCore.Builder.EndpointRouteBuilderExtensions.MapGet(this Microsoft.AspNetCore.Routing.IEndpointRouteBuilder endpoints, string pattern, System.Delegate handler)
```

Minimal API AOT



Name	Date modified	Type	Size
ApiAotSample.exe	12/10/2023 09:10	Application	6,672 KB
ApiAotSample.pdb	12/10/2023 09:10	Program Debug Database	59,980 KB
appsettings.Development.json	11/29/2023 10:25	JSON File	1 KB
appsettings.json	11/29/2023 10:25	JSON File	1 KB

```
PowerShell 7.4.0
Loading personal and system profiles took 511ms.
PS C:\projects\sources\SamplesInPractice\AotTest\ApiAotSample\out> .\ApiAotSample.exe
```

```
PowerShell 7.4.0
PS C:\projects\sources\SamplesInPractice\AotTest\ApiAotSample> cat .\Program.cs
var builder = WebApplication.CreateEmptyBuilder(new WebApplicationOptions());
builder.Services.AddRoutingCore();
builder.WebHost.UseKestrelCore();
// builder.Logging.AddConsole();

var app = builder.Build();

app.UseRouting();
app.UseEndpoints(endpoints =>
{
    endpoints.MapGet("/", () => "Hello World");
});

app.Run();
PS C:\projects\sources\SamplesInPractice\AotTest\ApiAotSample> dotnet-http :5000/
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Sun, 10 Dec 2023 01:14:00 GMT
Server: Kestrel
Transfer-Encoding: chunked
```

References

- https://github.com/dotnet/roslyn/blob/main/docs/features/interceptors.md
- https://github.com/dotnet/csharplang/issues/7009
- https://github.com/WeihanLi/SamplesInPractice/blob/main/CSharp12Sample/InterceptorSample.cs
- https://github.com/WeihanLi/SamplesInPractice/blob/main/InterceptorSamples
- https://andrewlock.net/exploring-the-dotnet-8-preview-changing-method-calls-with-interceptors/
- https://github.com/dotnet/aspnetcore/blob/main/src/Http/Http.Extensions/gen/RequestDelegateGenerator.cs
- https://khalidabuhakmeh.com/dotnet-8-interceptors
- https://code-maze.com/how-to-use-interceptors-in-c-12/

Thank You

Keep Coding