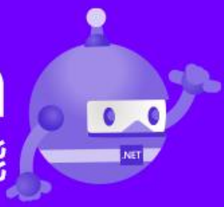
2021.NET Conf China

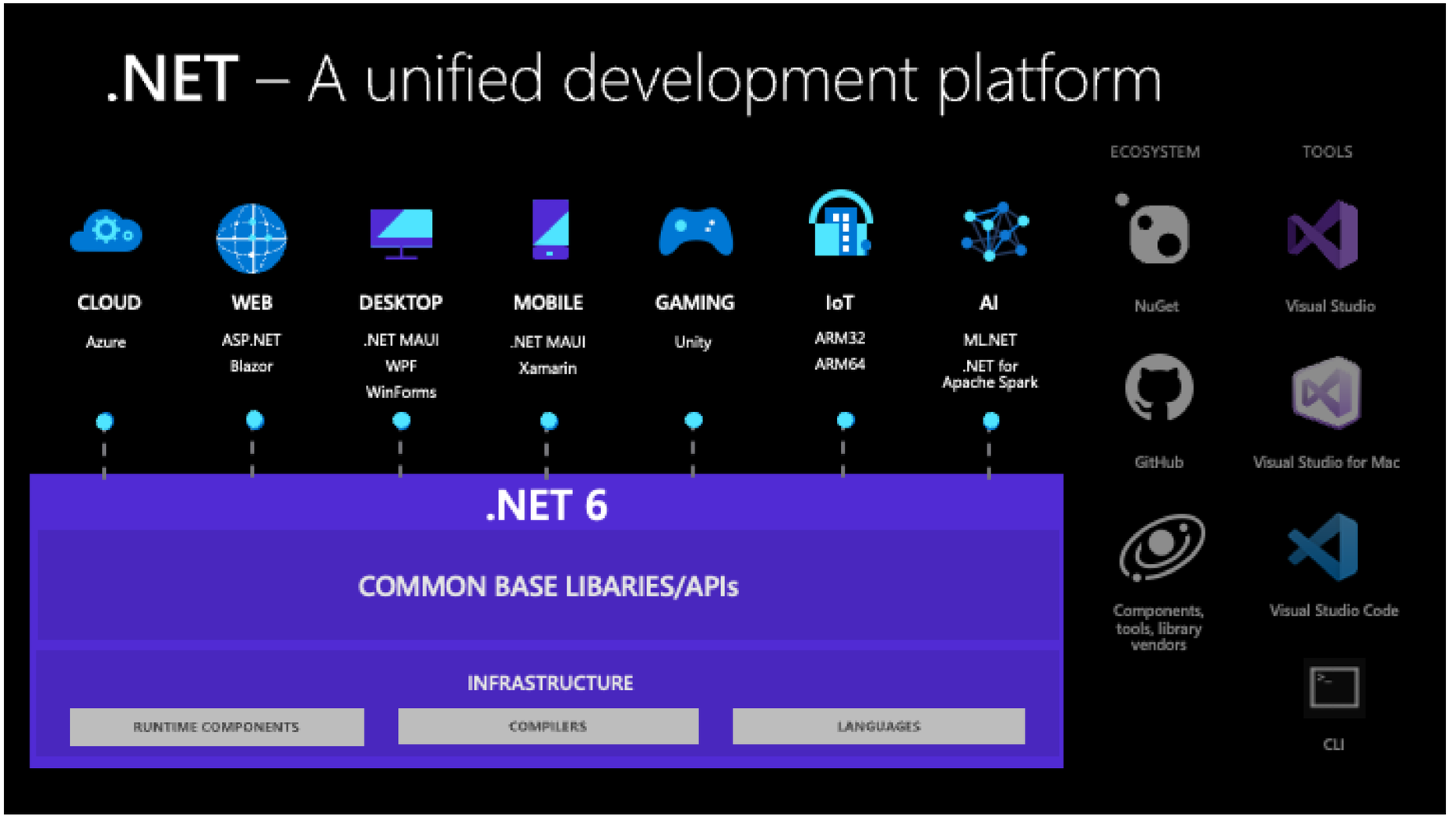
开源共建 | 开放创新 | 开发赋能



使用 .NET 6 新特性优化代码

李卫涵

The Fastest .NET Yet



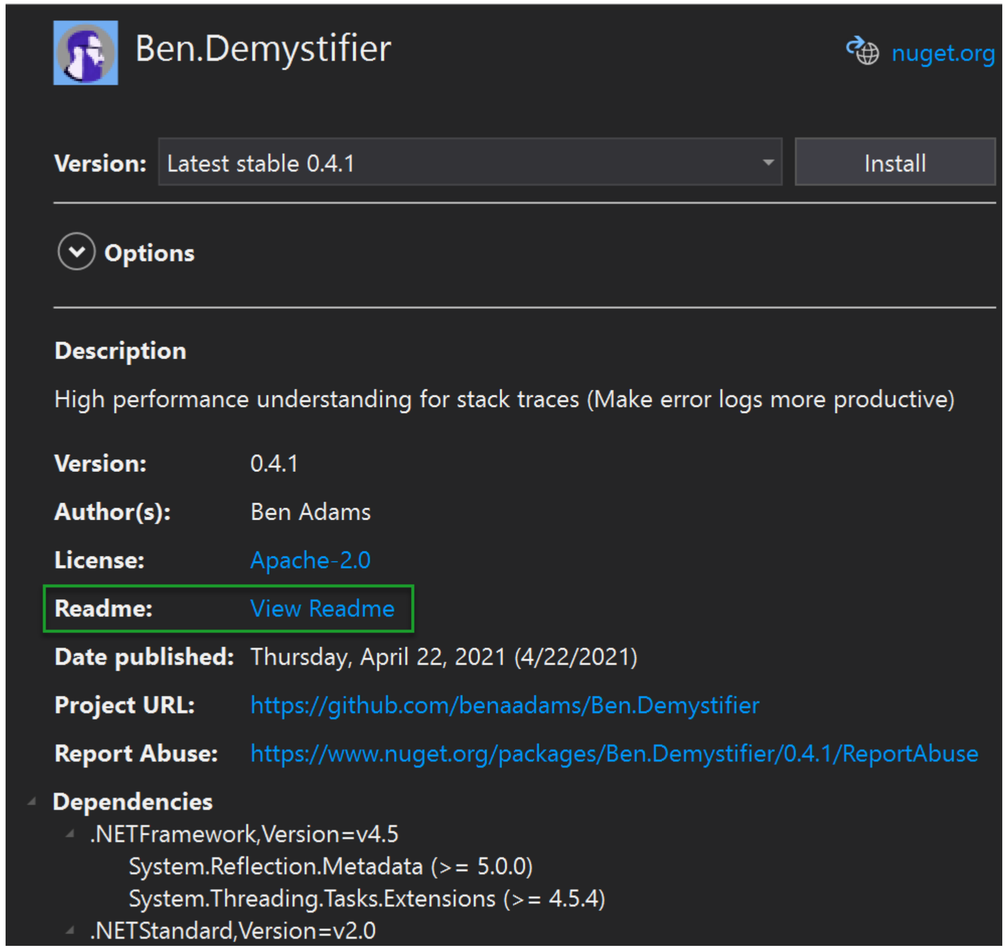
类库应用

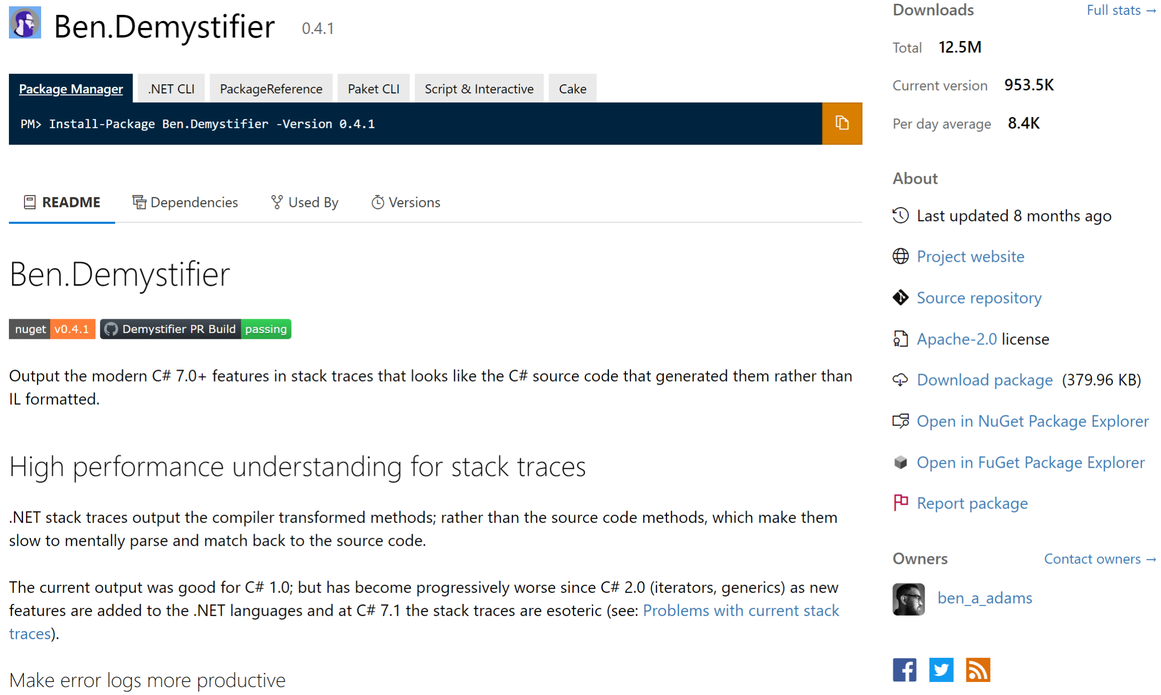
业务应用

类库应用

- NuGet Package ReadMe
- NuGet Package Validation
- Caller Argument Expression
- Logging Source Generator
- Json Source Generator
- Interpolate String Handler
- Namespace Updates

NuGet Package ReadMe





NuGet Package ReadMe


```
  <Project Sdk="Microsoft.NET.Sdk">
+   <PropertyGroup>
+     <PackageReadmeFile>README.md</PackageReadmeFile>
+   </PropertyGroup>
    <Import Project="../../build/logging.log4net.props" />
    <ItemGroup>
      <PackageReference Include="log4net" Version="2.0.12" />
      <None Include="log4net.config" Pack="true" />
      <ProjectReference Include="..\WeihanLi.Common\WeihanLi.Common.csproj" />
    </ItemGroup>
+   <ItemGroup>
+     <None Include="README.md">
+       <Pack>True</Pack>
+       <PackagePath>\</PackagePath>
+     </None>
+   </ItemGroup>
  </Project>
```

WeihanLi.Common.Logging.Log4Net 1.0.47 Prefix Reserved

ⓘ There is a newer prerelease version of this package available.
See the version list below for details.

Package Manager | .NET CLI | PackageReference | Paket CLI | Script & Interactive | Cake

```
PM> Install-Package WeihanLi.Common.Logging.Log4Net -Version 1.0.47
```

README | Dependencies | Used By | Versions | Release Notes

WeihanLi.Common.Logging.Log4Net nuget v1.0.47

Intro

- log4net extensions
  - log4net ElasticSearchAppender
  - Log4NetHelper
- `Log4NetLoggerProvider` for Microsoft.Extensions.Logging
- `Log4NetLogHelperProvider` for WeihanLi.Common.Logging

Use log4net only

NuGet Package Validation

```
    <PackageProjectUrl>https://github.com/WeihanLi/WeihanLi.Common/tree/dev/src/Wei
    ommon</PackageProjectUrl>
10      <NoWarn>$(NoWarn);1591;DE0003;SYSLIB0014;</NoWarn>
11  +   <EnablePackageValidation>true</EnablePackageValidation>
12  +   <PackageValidationBaselineVersion>1.0.47</PackageValidationBaselineVersion>
13    </PropertyGroup>
```

```
Error:
C:\Users\runneradmin\AppData\Local\Microsoft\dotnet\sdk\6.0.101\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Compatibility.Common.targets(32,5): error CP0009: Type 'WeihanLi.Common.NullScope' has the sealed modifier on lib/net6.0/WeihanLi.Common.dll but not on [Baseline] lib/net6.0/WeihanLi.Common.dll [D:\a\WeihanLi.Common\WeihanLi.Common\src\WeihanLi.Common\WeihanLi.Common.csproj]
Error:
C:\Users\runneradmin\AppData\Local\Microsoft\dotnet\sdk\6.0.101\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Compatibility.Common.targets(32,5): error CP0009: Type 'WeihanLi.Common.NullScope' has the sealed modifier on lib/netstandard2.0/WeihanLi.Common.dll but not on [Baseline] lib/netstandard2.0/WeihanLi.Common.dll
[D:\a\WeihanLi.Common\WeihanLi.Common\src\WeihanLi.Common\WeihanLi.Common.csproj]
Error:
C:\Users\runneradmin\AppData\Local\Microsoft\dotnet\sdk\6.0.101\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Compatibility.Common.targets(32,5): error CP0009: Type 'WeihanLi.Common.NullScope' has the sealed modifier on lib/netstandard2.1/WeihanLi.Common.dll but not on [Baseline] lib/netstandard2.1/WeihanLi.Common.dll
[D:\a\WeihanLi.Common\WeihanLi.Common\src\WeihanLi.Common\WeihanLi.Common.csproj]
```

```
15
16-     public Task<IDisposable> LockAsync() => LockAsync(CancellationToken.None
17
18-     public Task<IDisposable> LockAsync(CancellationToken cancellationToken)
19
20      public async Task<IDisposable> LockAsync(TimeSpan timeout, CancellationT
21      {
22          if (timeout <= TimeSpan.Zero)
```

```
15
16+     // public Task<IDisposable> LockAsync() => LockAsync(CancellationToken.N
17
18+     public Task<IDisposable> LockAsync(CancellationToken cancellationToken =
19
20      public async Task<IDisposable> LockAsync(TimeSpan timeout, CancellationT
21      {
22          if (timeout <= TimeSpan.Zero)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                  pwsh

Time Elapsed 00:00:33.31
PS C:\projects\sources\WeihanLi.Common> cd .\src\WeihanLi.Common\
PS C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common> dotnet pack
Microsoft (R) Build Engine version 17.0.0+c9eb9dd64 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  All projects are up-to-date for restore.
  WeihanLi.Common -> C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\bin\Debug\netstandard2.1\WeihanLi.Common.dll
  WeihanLi.Common -> C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\bin\Debug\net6.0\WeihanLi.Common.dll
  WeihanLi.Common -> C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\bin\Debug\netstandard2.0\WeihanLi.Common.dll
  Successfully created package 'C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\bin\Debug\WeihanLi.Common.1.0.48.nupkg'.
  Successfully created package 'C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\bin\Debug\WeihanLi.Common.1.0.48.snupkg'.
C:\Program Files\dotnet\sdk\6.0.101\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Compatibility.Common.targets(32,5): error CP0002: Member 'WeihanLi.Common.AsyncLock.LockA
sync()' exists on [Baseline] lib/net6.0/WeihanLi.Common.dll but not on lib/net6.0/WeihanLi.Common.dll [C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\WeihanLi.Co
mmon.csproj]
C:\Program Files\dotnet\sdk\6.0.101\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Compatibility.Common.targets(32,5): error CP0009: Type 'WeihanLi.Common.NullScope' has th
e sealed modifier on lib/net6.0/WeihanLi.Common.dll but not on [Baseline] lib/net6.0/WeihanLi.Common.dll [C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\WeihanLi
.Common.csproj]
C:\Program Files\dotnet\sdk\6.0.101\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Compatibility.Common.targets(32,5): error CP0002: Member 'WeihanLi.Common.AsyncLock.LockA
sync()' exists on [Baseline] lib/netstandard2.0/WeihanLi.Common.dll but not on lib/netstandard2.0/WeihanLi.Common.dll [C:\projects\sources\WeihanLi.Common\src\WeihanLi.Co
mmon\WeihanLi.Common.csproj]
C:\Program Files\dotnet\sdk\6.0.101\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Compatibility.Common.targets(32,5): error CP0009: Type 'WeihanLi.Common.NullScope' has th
ard2.1/WeihanLi.Common.dll [C:\projects\sources\WeihanLi.Common\src\WeihanLi.Common\WeihanLi.Common.csproj]
```

Validation rules: *https://docs.microsoft.com/zh-cn/dotnet/fundamentals/package-validation/diagnostic-ids*

Caller Argument Expression

Caller info attribute

- CallerFilePath
- CallerLineNumber
- CallerMemberName

```
public static void MainTest()
{
    // 我是谁，我在哪儿
    DumpCallerInfo();
}

private static void DumpCallerInfo(
    [CallerFilePath] string? callerFilePath = null,
    [CallerLineNumber] int? callerLineNumber = null,
    [CallerMemberName] string? callerMemberName = null
)
{
    Console.WriteLine("Caller info:");
    Console.WriteLine($@"CallerFilePath: {callerFilePath}
CallerLineNumber: {callerLineNumber}
CallerMemberName: {callerMemberName}
");
}
```

```
void CheckExpression(bool condition,
    [CallerArgumentExpression("condition")] string? message = null )
{
    Console.WriteLine($"Condition: {message}");
}

var a = 6;
var b = true;
CheckExpression(true);
CheckExpression(b);
CheckExpression(a > 5);

// Output:
// Condition: true
// Condition: b
// Condition: a > 5
```

CallerArgumentExpression(C# 10)

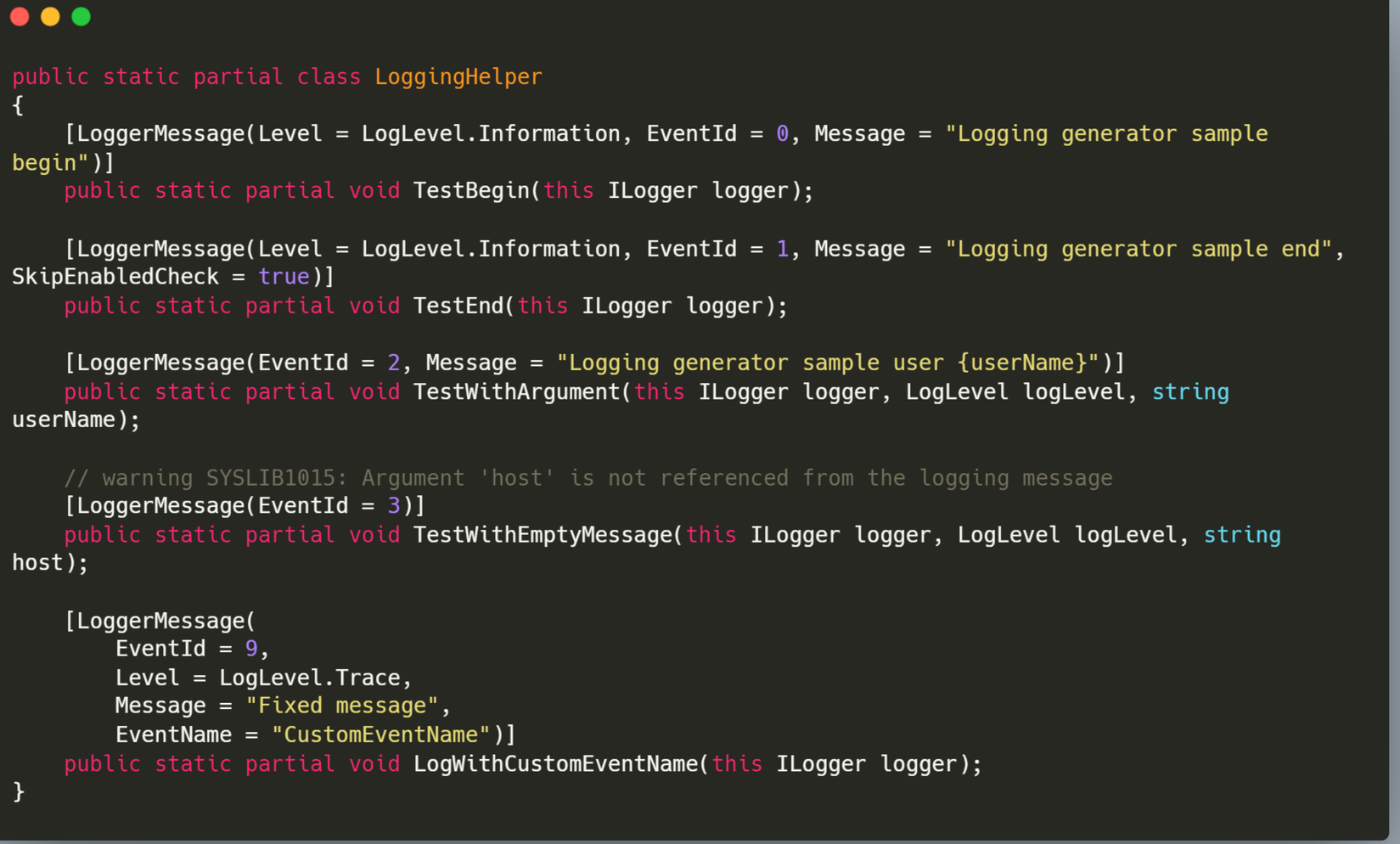
Caller Argument Expression

```
/// <summary>Throws an <see cref="ArgumentNullException"/> if <paramref name="argument"/> is null.
</summary>
/// <param name="argument">The reference type argument to validate as non-null.</param>
/// <param name="paramName">The name of the parameter with which <paramref name="argument"/>
corresponds.</param>
public static void ThrowIfNull([NotNull] object? argument, [CallerArgumentExpression("argument")]
string? paramName = null)
{
    if (argument is null)
    {
        Throw(paramName);
    }
}

[DoesNotReturn]
private static void Throw(string? paramName) =>
    throw new ArgumentNullException(paramName);
```

Caller Argument Expression

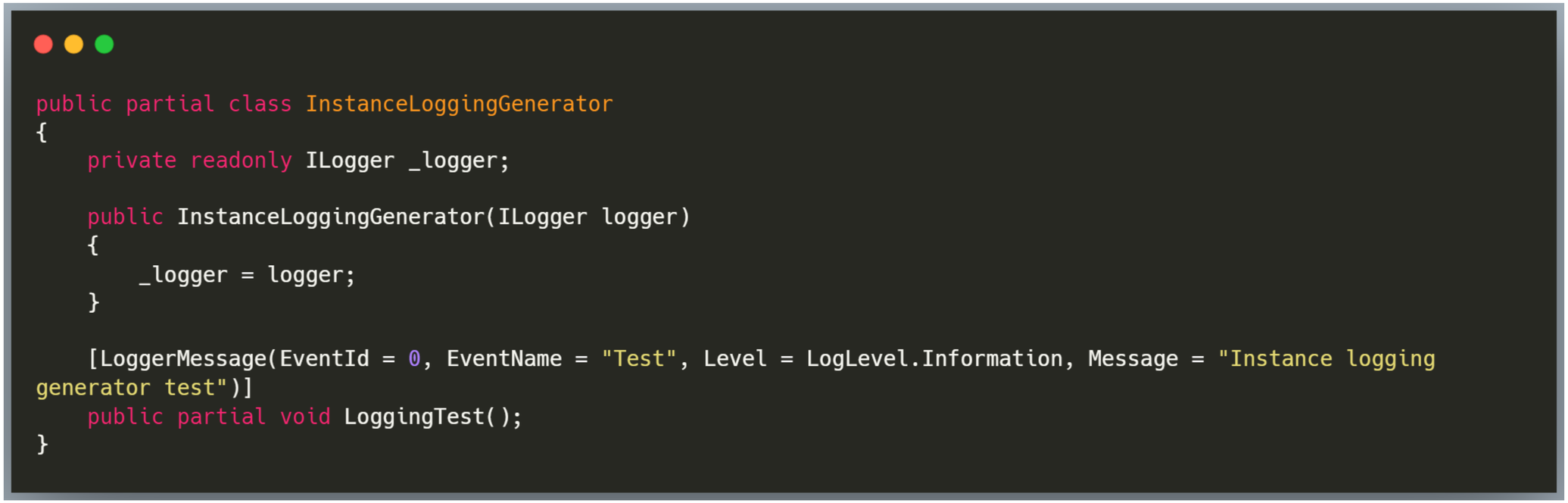

```
public static void Argument(bool condition, string message, [CallerArgumentExpression("condition")]
string? conditionExpression = null)
{
    if (!condition) throw new ArgumentException(message: message, paramName: conditionExpression);
}

public static void NotNullOrEmpty(string argument, [CallerArgumentExpression("argument")] string?
argumentExpression = null)
{
    if (string.IsNullOrEmpty(argument))
    {
        throw new ArgumentException("Can not be null or empty", argumentExpression);
    }
}

public static void InRange(int argument, int low, int high,
    [CallerArgumentExpression("argument")] string? argumentExpression = null,
    [CallerArgumentExpression("low")] string? lowExpression = null,
    [CallerArgumentExpression("high")] string? highExpression = null)
{
    if (argument < low)
    {
        throw new ArgumentOutOfRangeException(paramName: argumentExpression,
            message: $"{argumentExpression} ({argument}) cannot be less than {lowExpression}
({low}).");
    }

    if (argument > high)
    {
        throw new ArgumentOutOfRangeException(paramName: argumentExpression,
            message: $"{argumentExpression} ({argument}) cannot be greater than {highExpression}
({high}).");
    }
}
```

Logging Source Generator



```
public static partial class LoggingHelper
{
    [LoggerMessage(Level = LogLevel.Information, EventId = 0, Message = "Logging generator sample
begin")]
    public static partial void TestBegin(this ILogger logger);

    [LoggerMessage(Level = LogLevel.Information, EventId = 1, Message = "Logging generator sample end",
SkipEnabledCheck = true)]
    public static partial void TestEnd(this ILogger logger);

    [LoggerMessage(EventId = 2, Message = "Logging generator sample user {userName}")]
    public static partial void TestWithArgument(this ILogger logger, LogLevel logLevel, string
userName);

    // warning SYSLIB1015: Argument 'host' is not referenced from the logging message
    [LoggerMessage(EventId = 3)]
    public static partial void TestWithEmptyMessage(this ILogger logger, LogLevel logLevel, string
host);

    [LoggerMessage(
        EventId = 9,
        Level = LogLevel.Trace,
        Message = "Fixed message",
        EventName = "CustomEventName")]
    public static partial void LogWithCustomEventName(this ILogger logger);
}
```

Logging Source Generator

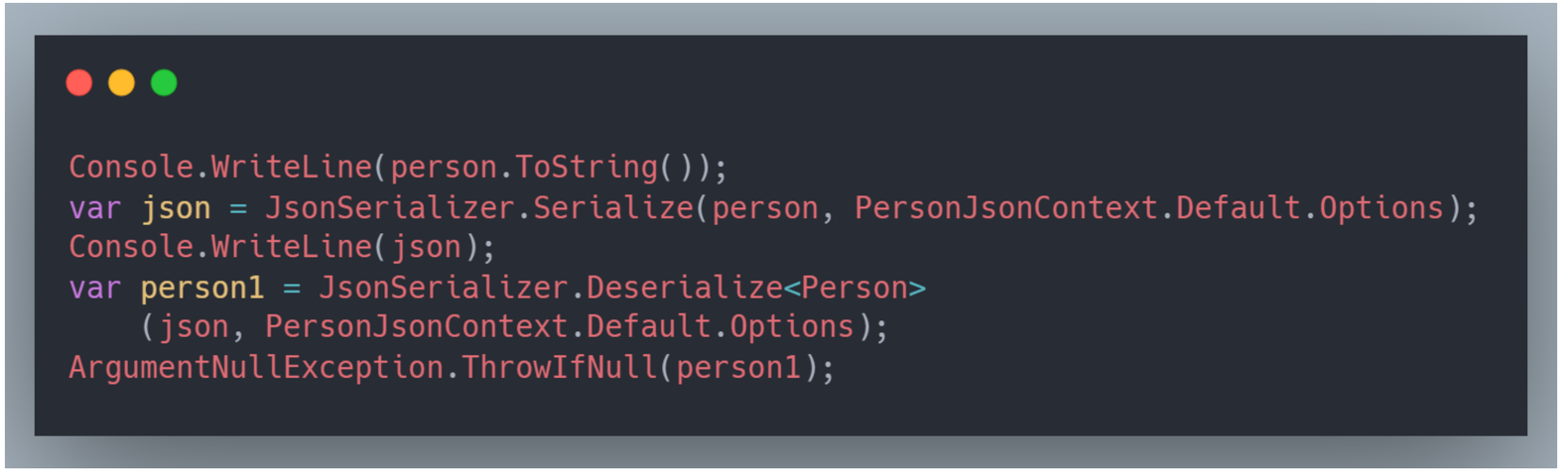


```
public partial class InstanceLoggingGenerator
{
    private readonly ILogger _logger;

    public InstanceLoggingGenerator(ILogger logger)
    {
        _logger = logger;
    }

    [LoggerMessage(EventId = 0, EventName = "Test", Level = LogLevel.Information, Message = "Instance logging
generator test")]
    public partial void LoggingTest();
}
```

Question: 如果有多个 ILogger 字段会怎么样?

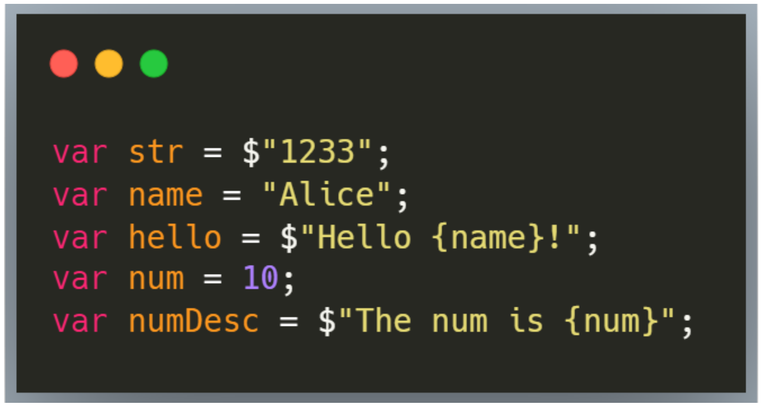
JSON Source Generator

```
private record Person
{
    //init-only properties, deserialization of which is currently not supported in source generation mode

    public string FirstName { get; set; } = string.Empty;

    public string LastName { get; set; } = string.Empty;
}

[JsonSourceGenerationOptions(PropertyNamingPolicy = JsonKnownNamingPolicy.CamelCase)]
[JsonSerializable(typeof(Person))]
private partial class PersonJsonContext : JsonSerializerContext
{
}
```



```
Console.WriteLine(person.ToString());
var json = JsonSerializer.Serialize(person, PersonJsonContext.Default.Options);
Console.WriteLine(json);
var person1 = JsonSerializer.Deserialize<Person>
    (json, PersonJsonContext.Default.Options);
ArgumentNullException.ThrowIfNull(person1);
```

Interpolated String Handler



```
var str = $"1233";
var name = "Alice";
var hello = $"Hello {name}!";
var num = 10;
var numDesc = $"The num is {num}";
```

```
IL_0000  nop
IL_0001  ldstr     "1233"
IL_0006  stloc.0     // str
IL_0007  ldstr     "Alice"
IL_000C  stloc.1     // name
IL_000D  ldstr     "Hello "
IL_0012  ldloc.1     // name
IL_0013  ldstr     "!"
IL_0018  call      String.Concat (String, String, String)
IL_001D  stloc.2     // hello
IL_001E  ldc.i4.s  0A  // 10
IL_0020  stloc.3     // num
IL_0021  ldstr     "The num is {0}"
IL_0026  ldloc.3     // num
IL_0027  box       Int32
IL_002C  call      String.Format (String, Object)
IL_0031  stloc.s   04    // numDesc
IL_0033  ret
```

Main ()

```
IL_0006  stloc.0     // str
IL_0007  ldstr     "Alice"
IL_000C  stloc.1     // name
IL_000D  ldstr     "Hello "
IL_0012  ldloc.1     // name
IL_0013  ldstr     "!"
IL_0018  call      String.Concat (String, String, String)
IL_001D  stloc.2     // hello
IL_001E  ldc.i4.s  0A  // 10
IL_0020  stloc.3     // num
IL_0021  ldloca.s  05
IL_0023  ldc.i4.s  0B  // 11
IL_0025  ldc.i4.1
IL_0026  call      DefaultInterpolatedStringHandler..ctor
IL_002B  ldloca.s  05
IL_002D  ldstr     "The num is "
IL_0032  call      DefaultInterpolatedStringHandler.AppendLiteral (String)
IL_0037  nop
IL_0038  ldloca.s  05
IL_003A  ldloc.3     // num
IL_003B  call      DefaultInterpolatedStringHandler.AppendFormatted<Int32> (Int32)
IL_0040  nop
IL_0041  ldloca.s  05
IL_0043  call      DefaultInterpolatedStringHandler.ToStringAndClear ()
IL_0048  stloc.s   04    // numDesc
IL_004A  ret
```

Interpolated String Handler

```
    [InterpolatedStringHandler]
    public ref struct DefaultInterpolatedStringHandler
    {
        public DefaultInterpolatedStringHandler(int literalLength, int formattedCount);
        public DefaultInterpolatedStringHandler(int literalLength, int formattedCount, System.IFormatProvider?
provider);
        public DefaultInterpolatedStringHandler(int literalLength, int formattedCount, System.IFormatProvider?
provider, System.Span<char> initialBuffer);

        public void AppendLiteral(string value);

        public void AppendFormatted<T>(T value);
        public void AppendFormatted<T>(T value, string? format);
        public void AppendFormatted<T>(T value, int alignment);
        public void AppendFormatted<T>(T value, int alignment, string? format);

        public void AppendFormatted(ReadOnlySpan<char> value);
        public void AppendFormatted(ReadOnlySpan<char> value, int alignment = 0, string? format = null);

        public void AppendFormatted(string? value);
        public void AppendFormatted(string? value, int alignment = 0, string? format = null);
        public void AppendFormatted(object? value, int alignment = 0, string? format = null);

        public string ToStringAndClear();
    }
```

Interpolated String Handler


```
[InterpolatedStringHandler]
public struct CustomInterpolatedStringHandler
{
    private readonly StringBuilder builder;

    public CustomInterpolatedStringHandler(int literalLength, int formattedCount)
    {
        builder = new StringBuilder(literalLength);
        Console.WriteLine($"\tliteral length: {literalLength}, formattedCount: {formattedCount}");
    }

    // Required
    public void AppendLiteral(string s)
    {
        Console.WriteLine($"\tAppendLiteral called: {{{s}}}");
        builder.Append(s);
        Console.WriteLine($"\tAppended the literal string");
    }

    // Required
    public void AppendFormatted<T>(T t)
    {
        Console.WriteLine($"\tAppendFormatted called: {{{t}}} is of type {typeof(T)}");
        builder.Append(t?.ToString());
        Console.WriteLine($"\tAppended the formatted object");
    }

    public override string ToString()
    {
        return builder.ToString();
    }
}
```

```
private static void LogInterpolatedString(string str)
{
    Console.WriteLine(nameof(LogInterpolatedString));
    Console.WriteLine(str);
}

private static void LogInterpolatedString(CustomInterpolatedStringHandler stringHandler)
{
    Console.WriteLine(nameof(LogInterpolatedString));
    Console.WriteLine(nameof(CustomInterpolatedStringHandler));
    Console.WriteLine(stringHandler.ToString());
}

LogInterpolatedString("The num is 10");
LogInterpolatedString($"The num is {num}");
```

```
public const string Name = "Xiao Ming";
public const string Hello = $"Hello {Name}";
```

Namespace updates

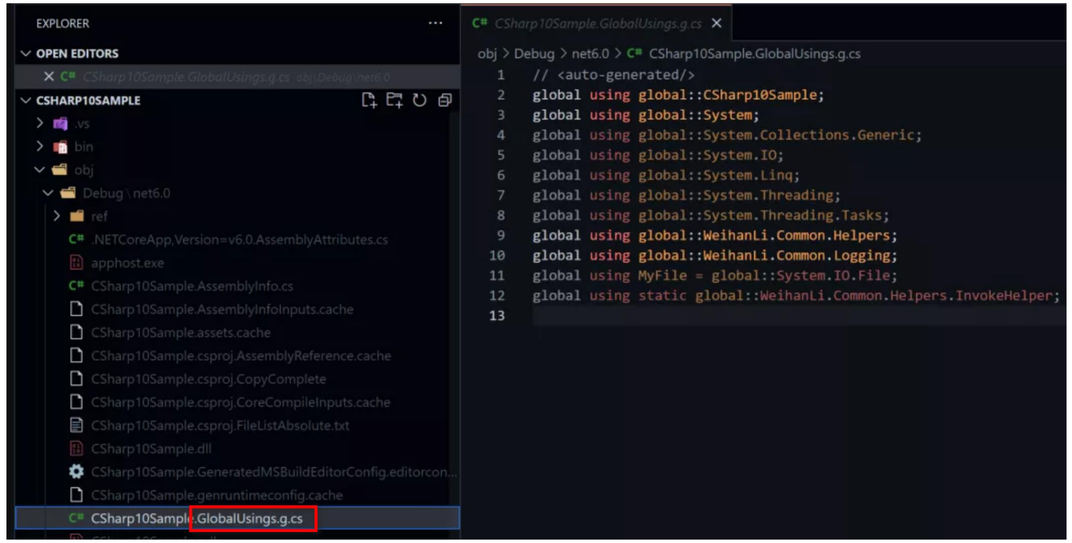
Global Usings

```
global using System;
global using static System.Console; // static using
global using MyFile = System.IO.File; // alias
```

```
<ItemGroup>
  <Using Include="$(RootNamespace)"/>
  <Using Include="WeihanLi.Common.Helpers"/>
  <Using Include="WeihanLi.Common.Helpers.InvokeHelper" Static="true"/>
  <Using Include="WeihanLi.Common.Logging"/>
  <Using Include="System.IO.File" Alias="MyFile"/>
  <Using Remove="System.Net.Http" />
</ItemGroup>
```

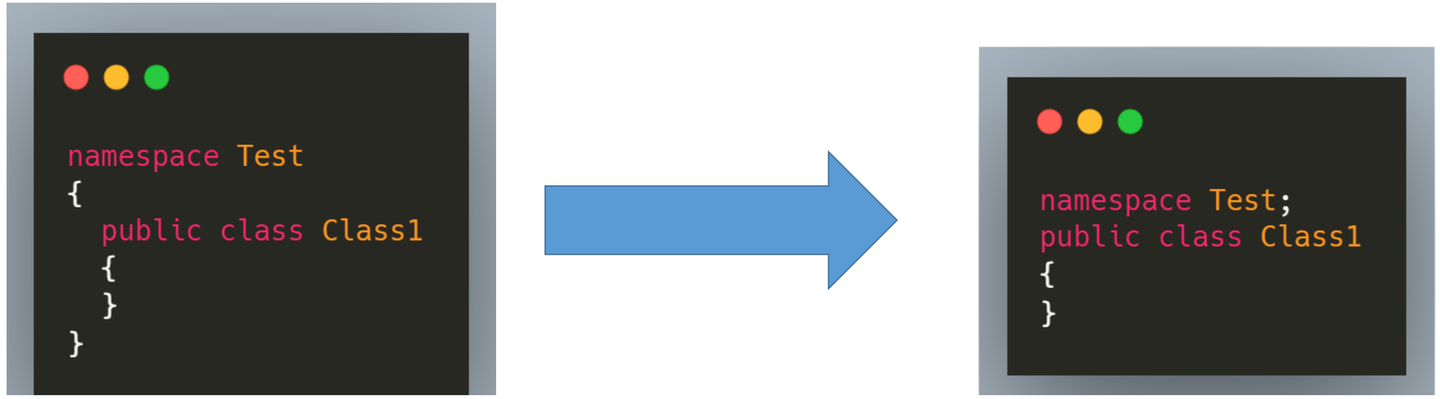
Implicit Usings

```
<PropertyGroup>
    <LangVersion>latest</LangVersion>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <PackageIcon>icon.png</PackageIcon>
```



Namespace updates

File-scoped namespace



限制

- 一个文件中只能声明一个 namespace，如果一个文件中要有两个不同的命名空间不能使用 file-scoped namespace
- 不能和之前命名空间用法一起使用

小技巧

- Editorconfig(*csharp_style_namespace_declarations=file_scoped:suggestion*)
- dotnet format
- Combine Directory.Build.props/Directory.Build.targets with implicit usings

业务应用

- Parallel.ForEachAsync
- PriorityQueue
- PeriodicTimer
- Random.Shared/WaitAsync
- JSON Writable DOM
- More Linq
- With expression support
- Socks proxy support/DateOnly/TimeOnly
- Minimal API
- Http logging middleware

Parallel.ForEachAsync

```csharp
using var semaphore = new SemaphoreSlim(10, 10);
await Task.WhenAll(Enumerable.Range(1, 100)
.Select(async _ =>
{
    try
    {
        await semaphore.WaitAsync();
        await Task.Delay(1000);
    }
    finally
    {
        semaphore.Release();
    }
}));
```

```csharp
public static Task ForEachAsync<TSource>
(IEnumerable<TSource> source, Func<TSource, CancellationToken, ValueTask> body);

public static Task ForEachAsync<TSource>(IEnumerable<TSource> source, CancellationToken cancellationToken,
Func<TSource, CancellationToken, ValueTask> body);

public static System.Threading.Tasks.Task ForEachAsync<TSource>(IEnumerable<TSource> source, ParallelOptions
parallelOptions, Func<TSource, CancellationToken, ValueTask> body);

public static Task ForEachAsync<TSource>(IAsyncEnumerable<TSource> source, Func<TSource, CancellationToken,
ValueTask> body);

public static Task ForEachAsync<TSource>(AsyncEnumerable<TSource> source, CancellationToken cancellationToken,
Func<TSource, CancellationToken, ValueTask> body);

public static Task ForEachAsync<TSource>(IAsyncEnumerable<TSource> source, ParallelOptions parallelOptions,
Func<TSource, CancellationToken, ValueTask> body);
```

```csharp
await Parallel.ForEachAsync(Enumerable.Range(1, 100),
new ParallelOptions()
{
    MaxDegreeOfParallelism = 10
}, async (_, _) => await Task.Delay(1000));
```

```csharp
public class ParallelOptions
{
    public TaskScheduler? TaskScheduler { get; set; }
    public int MaxDegreeOfParallelism { get; set; }
    public CancellationToken CancellationToken { get; set; }
}
```

PriorityQueue

```
var random = new Random();

var queue = new PriorityQueue<string, int>();
for (var i = 1; i <= 10; i++)
{
    queue.Enqueue($"Message_{i}", random.Next(10, 100));
}

while (queue.TryDequeue(out var message, out _))
{
    Console.WriteLine(message);
}
```

```
public class High2LowComparer : IComparer<int>
{
    public int Compare(int x, int y)
    {
        return y.CompareTo(x);
    }
}
```

```
var list = new List<(string name, int score)>
{
    ("Mike", 99),
    ("Ming", 100),
    ("Jane", 96),
    ("Yi", 94),
    ("Harry", 90),
};

priorityQueue = new PriorityQueue<string, int>(new High2LowComparer());
priorityQueue.EnqueueRange(list);
var rank = 0;
while (rank++ < 3 && priorityQueue.TryDequeue(out var name, out var
score))
    Console.WriteLine($"Rank({rank}):Name:{name}, score:{score}");
}
```

PeriodicTimer

```
using var timer = new PeriodicTimer(TimeSpan.FromSeconds(3));
try
{
    while (await timer.WaitForNextTickAsync(cts.Token))
    {
        Console.WriteLine($"Timed event triggered({DateTime.Now:HH:mm:ss})");
    }
}
catch (OperationCanceledException)
{
    Console.WriteLine("Operation cancelled");
}
```

```
var timer1 = new PeriodicTimer(TimeSpan.FromSeconds(2));
timer1.Dispose();
if (await timer1.WaitForNextTickAsync())
{
    Console.WriteLine("Timer1 event triggered");
}
```

Random.Shared/WaitAsync

Random.Shared

```
Console.WriteLine(Random.Shared.Next(1, 100));

Parallel.For(0, Environment.ProcessorCount, _ =>
{
    var thread = new Thread(() =>
    {
        var arr = new int[10];
        for (var i = 0; i < arr.Length; i++)
        {
            arr[i] = Random.Shared.Next(1, 100);
        }
        Console.WriteLine(arr.Average());
    });
    thread.Start();
});
```

Provides a thread-safe Random instance that may be used concurrently from any thread.

WaitAsync

```
// Timeout
await Task.Delay(TimeSpan.FromSeconds(5))
  .WaitAsync(TimeSpan.FromSeconds(3));


using var cancellationTokenSource = new CancellationTokenSource();
    cancellationTokenSource.CancelAfter(TimeSpan.FromSeconds(2));

// CancellationToken
await Task.Delay(TimeSpan.FromSeconds(5))
  .WaitAsync(cancellationTokenSource.Token);

// Timeout and cancellationToken
await Task.Delay(TimeSpan.FromSeconds(5))
  .WaitAsync(TimeSpan.FromSeconds(10), cancellationTokenSource.Token);
```

TimeoutException: for timeout
TaskCanceledException: for cancellation token

JSON Writable DOM

```
var testObj = new {Name = "Ming", Age = 10};
var jsonString = JsonSerializer.Serialize(testObj);
var jsonNode = JsonNode.Parse(jsonString);
if (jsonNode is JsonObject jsonObject)
{
    jsonObject["Name"]?.GetValue<string>().Dump();
    jsonObject["Age"]?.GetValue<int>().Dump();

    jsonObject["Name"] = "Michael";
    jsonObject.ToJsonString().Dump();
}
```

- JsonNode: JSON 文档中的一个节点，对应 Newtonsoft.Json 里的 JToken
- JsonObject: JSON 对象，对应 Newtonsoft.Json 里的 JObject
- JsonArray: JSON 数组，对应 Newtonsoft.Json 里的 JArray
- JsonValue: JSON 中的一个值，对应 Newtonsoft.Json 里的 JValue

More Linq

更好的 Index/Range 支持

```
Enumerable.Range(1, 10).ElementAt(^2).Dump(); // returns 9
Enumerable.Range(1, 10).Take(^2..).Dump(); // returns [9,10]
Enumerable.Range(1, 10).Take(..2).Dump(); // returns [1,2]
Enumerable.Range(1, 10).Take(2..4).Dump(); // returns [3,4]
```

Default extensions 优化

```
Enumerable.Empty<int>().FirstOrDefault(-1).Dump();
Enumerable.Empty<int>().SingleOrDefault(-1).Dump();
Enumerable.Empty<int>().LastOrDefault(-1).Dump();
```

XxxBy 扩展

```
// DistinctBy/UnionBy/IntersectBy/ExceptBy
Enumerable.Range(1, 20).DistinctBy(x => x % 3).Dump();
// [1, 2, 3]
var first = new (string Name, int Age)[] { ("Francis", 20), ("Lindsey", 30), ("Ashley", 40) };
var second = new (string Name, int Age)[] { ("Claire", 30), ("Pat", 30), ("Drew", 33) };

// { ("Francis", 20), ("Lindsey", 30), ("Ashley", 40), ("Drew", 33) }
first.UnionBy(second, person => person.Age).Select(x=>$"{x.Name}, {x.Age}").Dump();

// MaxBy/MinBy
var people = new (string Name, int Age)[] { ("Francis", 20), ("Lindsey", 30), ("Ashley", 40) };
people.MaxBy(person => person.Age).Dump(); // ("Ashley", 40)
people.MinBy(x => x.Name).Dump(); // ("Ashley", 40)
```

Chuck 扩展

```
var list = Enumerable.Range(1, 10).ToList();
var chucks = list.Chunk(3);
chucks.Dump();

// [[1,2,3],[4,5,6],[7,8,9],[10]]
```

With expression

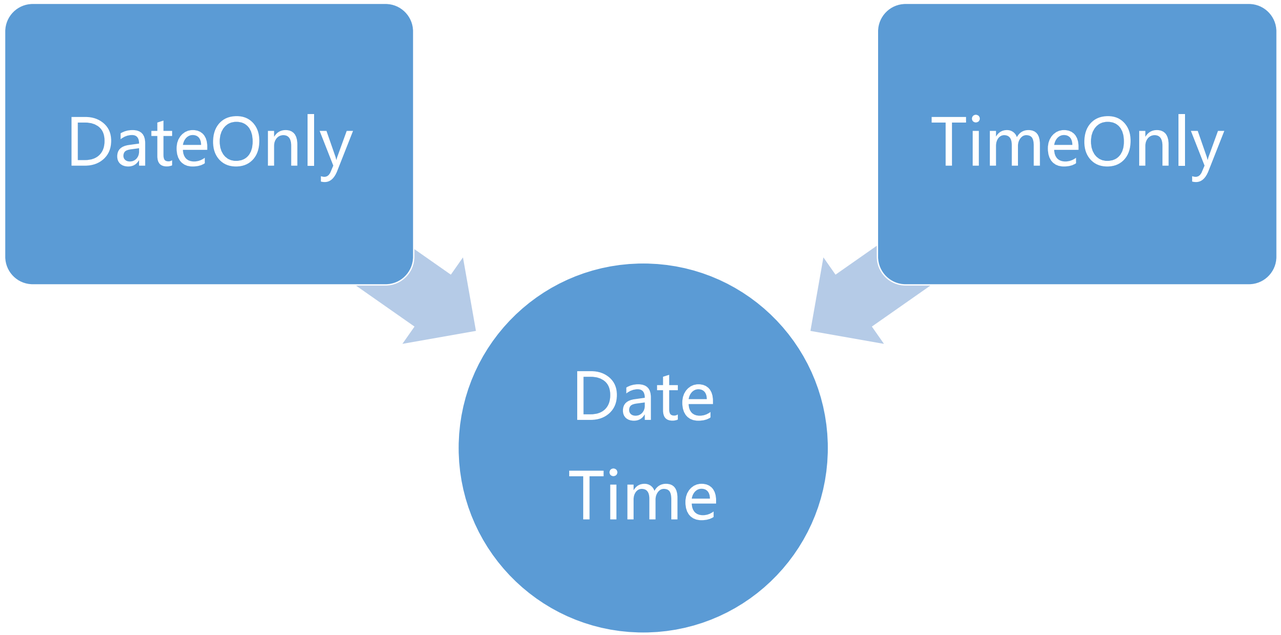
- Record class
- Record struct
- Struct
- Anonymous types

```
// record struct
public record struct Point(int X, int Y);

var updatedPoint = point with { X = 20 };


// record class
public record class Person(string FirstName, string LastName);

var updatedPerson = person with { FirstName = "Mary" };


// common struct
public struct Point2(int X, int Y, int Z);

var updatedPoint2 = point2 with { Z = 10 };


// anonymous object
var p = new { Name = "Kangkang", Age = 10 };

var updatedP = p with { Age = 20 };
```

Socks proxy support

```
var handler = new HttpClientHandler
{
    Proxy = new WebProxy("socks5://127.0.0.1", 9050)
};
var httpClient = new HttpClient(handler);
```

DateOnly/TimeOnly




```
var date = new DateOnly(2021, 12, 18);
var time = new TimeOnly(10, 24);

var currentDate = DateOnly.FromDateTime(DateTime.Now);
var currentDate = TimeOnly.FromDateTime(DateTime.Now);
```

Minimal API

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

```
var app = WebApplication.Create(args);
app.Map("/", () => "Hello World");
app.Run();
```

Http logging middleware


```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();
var app = builder.Build();

app.UseHttpLogging();
app.MapControllers();

app.Run();
```

```
info: Microsoft.AspNetCore.HttpLogging.HttpLoggingMiddleware[1]
      Request:
      Protocol: HTTP/1.1
      Method: GET
      Scheme: http
      PathBase:
      Path: /weatherforecast
      Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed
-exchange;v=b3;q=0.9
      Connection: keep-alive
      Host: localhost:5084
      User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/94.0.4606.54 Safari/537.36
      Accept-Encoding: gzip, deflate, br
      Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
      Cache-Control: [Redacted]
      Upgrade-Insecure-Requests: [Redacted]
      sec-ch-ua: [Redacted]
      sec-ch-ua-mobile: [Redacted]
      sec-ch-ua-platform: [Redacted]
      Sec-Fetch-Site: [Redacted]
      Sec-Fetch-Mode: [Redacted]
      Sec-Fetch-User: [Redacted]
      Sec-Fetch-Dest: [Redacted]
info: Microsoft.AspNetCore.HttpLogging.HttpLoggingMiddleware[2]
      Response:
      StatusCode: 200
      Content-Type: application/json; charset=utf-8
      Date: [Redacted]
      Server: [Redacted]
      Transfer-Encoding: chunked
```

```
builder.Services.AddHttpLogging(options =>
{
    options.LoggingFields = Microsoft.AspNetCore.HttpLogging.HttpLoggingFields.All;
    options.RequestHeaders.Add("Cache-Control");
    options.ResponseHeaders.Add("Server");
});
```

More

- Hot reload
- Workload
  - MAUI
  - WebAssembly AOT
- Single-file app compression
- HTTP3
- New metrics for Open Telemetry
- More Roslyn Analyzers
- ...

2021.NET Conf China

开源共建 | 开放创新 | 开发赋能

Thank you