

RSCG nr 7 : Microsoft.Extensions.Logging

Info

Nuget : <https://www.nuget.org/packages/Microsoft.Extensions.Logging/>

You can find more details at : <https://learn.microsoft.com/en-us/dotnet/core/extensions/logger-message-generator-generators/>

Author :Microsoft

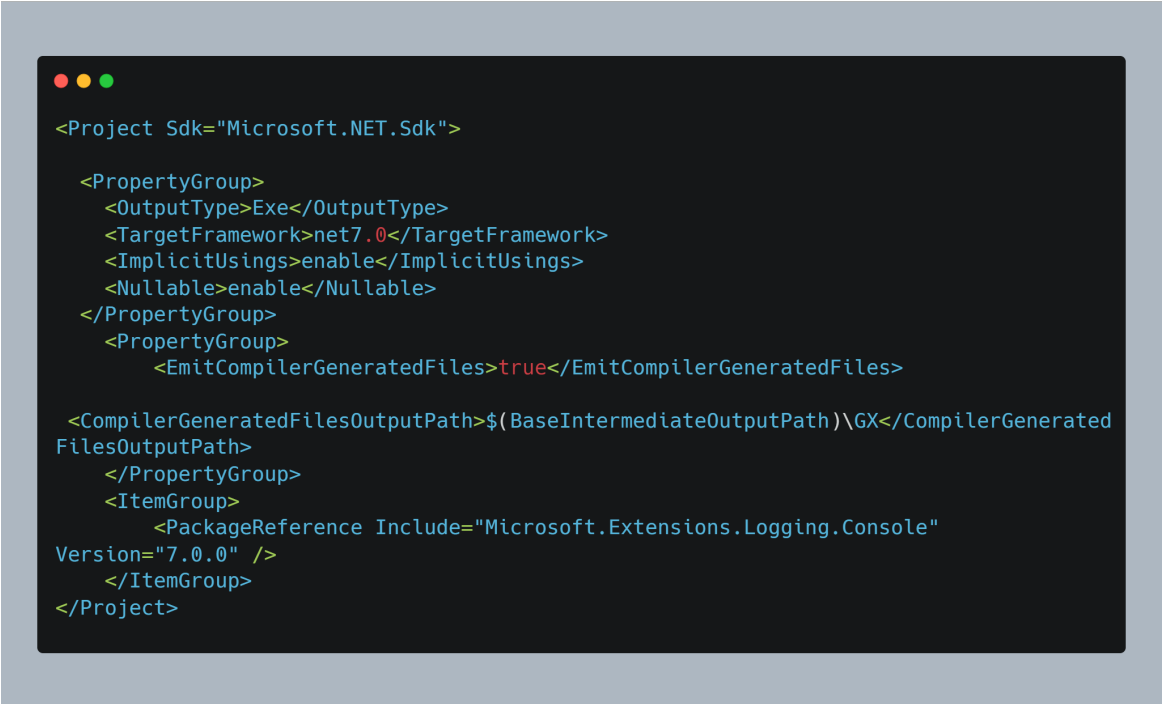
Source : <https://github.com/dotnet/runtime>

About

Logging defined and compiled

How to use

Add reference to the Microsoft.Extensions.Logging in the csproj



```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>
  <PropertyGroup>
    <EmitCompilerGeneratedFiles>true</EmitCompilerGeneratedFiles>

    <CompilerGeneratedFilesOutputPath>$(BaseIntermediateOutputPath)\GX</CompilerGeneratedFilesOutputPath>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.Extensions.Logging.Console"
Version="7.0.0" />
  </ItemGroup>
</Project>
```

This was for me the starting code

I have **coded** the file Program.cs



```
using System.Text.Json;
using Microsoft.Extensions.Logging;

using ILoggerFactory loggerFactory = LoggerFactory.Create(
    builder =>
        //https://learn.microsoft.com/en-us/dotnet/core/extensions/console-log-formatter
        builder.AddSimpleConsole()
        //builder.AddJsonConsole(
        //    options =>
        //        options.JsonWriterOptions = new JsonWriterOptions()
        //        {
        //            Indented = true
        //        })
    );

ILogger<SampleObject> logger = loggerFactory.CreateLogger<SampleObject>();
logger.LogInformation("test");
//https://learn.microsoft.com/en-us/dotnet/core/extensions/logger-message-generator
(new LoggingSample(logger)).TestLogging();
file readonly record struct SampleObject { }
```

I have **coded** the file LogDemo.cs

```

using Microsoft.Extensions.Logging;

public partial class LoggingSample
{
    private readonly ILogger _logger;

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

    [LoggerMessage(
        EventId = 20,
        Level = LogLevel.Critical,
        Message = "Value is {value:E}")]
    public static partial void UsingFormatSpecifier(
        ILogger logger, double value);

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

    [LoggerMessage(
        EventId = 10,
        Message = "Welcome to {city} {province}!")]
    public partial void LogWithDynamicLogLevel(
        string city, LogLevel level, string province);

    public void TestLogging()
    {
        LogWithCustomEventName();

        LogWithDynamicLogLevel("Vancouver", LogLevel.Warning, "BC");
        LogWithDynamicLogLevel("Vancouver", LogLevel.Information, "BC");

        UsingFormatSpecifier(_logger, 12345.6789);
    }
}

```

And here are the *generated* files

The file *generated* is LoggerMessage.g.cs

```

// Microsoft.Extensions.Logging
#nullable enable
{
    partial class LoggingSample
    {
        [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.Extensions.Logging.Generators", "7.0.2.1805")]
        private static readonly
        global::System.Action<global::Microsoft.Extensions.Logging.Logger,
        global::System.Double> global::System.Action<global::Microsoft.Extensions.Logging.Logger,
        global::System.Double> _stringFormatSpecificCallback =
        global::Microsoft.Extensions.Logging.LoggerMessage.Define<global::System.Double>
        (global::Microsoft.Extensions.Logging.LogLevel.Critical, new
        global::Microsoft.Extensions.Logging.EventId(), nameof(stringFormatSpecific)), "value
        is {value:0}", new global::Microsoft.Extensions.Logging.LogLevelOptions() {
            _isDebugEnabled = true });

        [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.Extensions.Logging.Generators", "7.0.2.1805")]
        public static partial void
        using(stringFormatSpecificCallback global::Microsoft.Extensions.Logging.Logger,
        global::System.Double value)
        {
            if
            (logger.IsEnabled(global::Microsoft.Extensions.Logging.LogLevel.Critical))
            {
                _stringFormatSpecificCallback(logger, value, null);
            }
        }

        [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.Extensions.Logging.Generators", "7.0.2.1805")]
        private static readonly
        global::System.Action<global::Microsoft.Extensions.Logging.Logger,
        global::System.Exception> _loggerMessageSpecificCallback =
        global::Microsoft.Extensions.Logging.LoggerMessage.Define<global::Microsoft.Extensions.Logging.LogLevel.Trace, new global::Microsoft.Extensions.Logging.EventId(),
        "CustomEventName", 1, 1> (new Logger, new
        global::Microsoft.Extensions.Logging.LogLevelOptions() { _isDebugEnabled = true });

        [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.Extensions.Logging.Generators", "7.0.2.1805")]
        public partial void LogCustomEventName()
        {
            if
            (!logger.IsEnabled(global::Microsoft.Extensions.Logging.LogLevel.Trace))
            {
                _loggerMessageSpecificCallback(logger, null);
            }
        }

        // Comments that are supported by the logging infrastructure and is not intended
        // to be used directly from your code. It is subject to change in the future. <summary>

        [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.Extensions.Logging.Generators", "7.0.2.1805")]
        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Never)]
        private readonly struct _LogWithDynamicLogLevelStruct :
        global::System.Collections.Generic.IReadOnlyList<global::System.Collections.Generic.KeyValuePair<string, object>
        {
            private readonly global::System.String _city;
            private readonly global::System.String _province;

            public _LogWithDynamicLogLevelStruct(global::System.String city,
            global::System.String province)
            {
                this._city = city;
                this._province = province;
            }

            public override string ToString()
            {
                var city = this._city;
                var province = this._province;
                return $"Welcome to {city} {province}!";
            }

            public static readonly global::System.Func<_LogWithDynamicLogLevelStruct,
            global::System.Exception, string> Format = (state, ex) => state.ToString();

            public int Count => 1;

            public global::System.Collections.Generic.KeyValuePair<string, object>
            this[int index]
            {
                get => index switch
                {
                    0 => new global::System.Collections.Generic.KeyValuePair<string,
                    object>("city", this._city);
                    1 => new global::System.Collections.Generic.KeyValuePair<string,
                    object>("province", this._province);
                    _ => new global::System.Collections.Generic.KeyValuePair<string,
                    object>("OriginalFormat", "Welcome to {city} {province}!");
                }
                => throw new
            global::System.InvalidOperationException(nameof(index)), // Return the same exception
            whenever the exception is thrown in this code
            };
            public
            global::System.Collections.Generic.IEnumerator<global::System.Collections.Generic.KeyValuePair<string,
            object> GetEnumerator()
            {
                for (int i = 0; i < 1; i++)
                {
                    yield return this[i];
                }
            }

            global::System.Collections.IEnumerator
            global::System.Collections.IEnumerable.GetEnumerator() => GetEnumerator();
        }

        [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.Extensions.Logging.Generators", "7.0.2.1805")]
        public partial void LogWithDynamicLogLevel(global::System.String city,
        global::Microsoft.Extensions.Logging.LogLevel level, global::System.String province)
        {
            if (!logger.IsEnabled(level))
            {
                logger.Log
                level,
                new global::Microsoft.Extensions.Logging.EventId(),
                nameof(LogWithDynamicLogLevel),
                new _LogWithDynamicLogLevelStruct(city, province),
                null,
                _LogWithDynamicLogLevelStruct.Format);
            }
        }
    }
}

```

You can download the code and this page as pdf from https://ignatandrei.github.io/RSCG_Examples/v2/docs/Microsoft.Extensions.Logging

You can see the whole list at https://ignatandrei.github.io/RSCG_Examples/v2/docs/List-of-RSCG