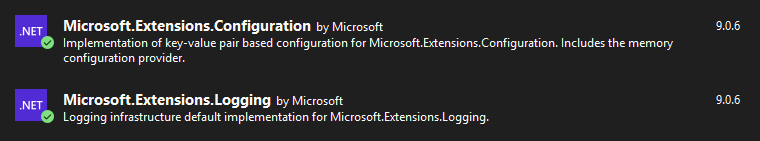
Logging using Microsoft.Extensions.Logging

Using Microsoft ILogger interface to add logging features. This can be used for production process but for scaling up to a bigger user base and high volume use a third-party system instead of simple implementation of ILogger.

## Guide Topics:

1. Installing the packages
2. Code setup
3. Usage
4. Additional notes

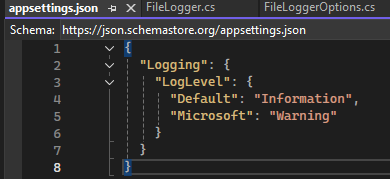
### Installing the packages



Note: Microsoft.Extensions.Configuration is optional but useful to control the systems variables.

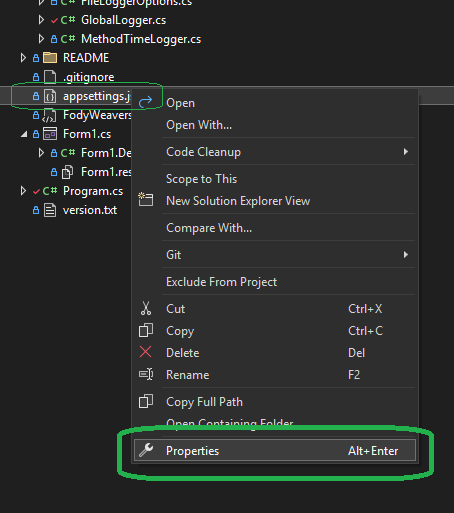
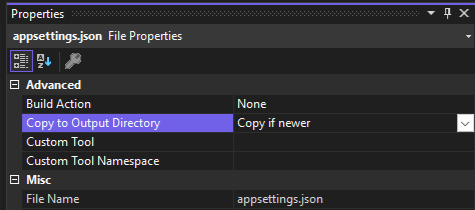
### Code Setup

1. Add FileLogger inheriting from ILogger.  
   Minimally override this function to print to console or save log to file.  
   public void Log<TState>
2. Create the appsettings.json into the project. This is loaded through the initiating class **GlobalLogger.cs**   
     
   example: appsettings.json



Important note: To make this work the appsettings.json must be copy over to the build folder.

Steps to tell build to copy the file appsettings.json to build folder.

1. Right click on appsettings.json > Properties  
   
2. Set the Build Action=None, Copy to Output Directory=<pick any Copy option>
3. This will turn into this code in the project xml file **CSharpAppPlayground.csproj**

<ItemGroup>

<None Update="appsettings.json">

<CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>

</None>  
</ItemGroup>

1. Initialize the logger in Program.cs before the app begins.

### Usage

Simple example:

// ILogger logger = new FileLogger("CSharpAppPlayground", new FileLoggerOptions { FilePath = "log.txt", LogLevel = LogLevel.Information });

// logger.LogInformation("Application started.");

Once initialized in the project the ILogger has logLevels

Possible LogLevels from ILogger

|  |  |  |
| --- | --- | --- |
| **Enum name** | **Enum value** | **ILogger extension method** |
| Trace | 0 | LogTrace |
| Debug | 1 | LogDebug |
| Information | 2 | LogInformation |
| Warning | 3 | LogWarning |
| Error | 4 | LogError |
| Critical | 5 | LogCritical |
| None | 6 |  |

Based on the LogLevel passed the function gets called or not, If the LogLevel is above Warning then LogInformation will not be recorded  
  
GlobalLogger.Instance.LogInformation("Run Clicked");

### Additional Notes

The Log will be saved in the bin folder once it’s built and ran.  
example:  
~\CSharpAppPlayground\bin\Debug\net8.0-windows

Sources:

<https://www.roundthecode.com/dotnet-tutorials/how-aspnet-core-logging-works-ilogger-loglevel>