# Logging:

Logging is an important tool in analyzing app performance.

**Logging Levels:**

| **Level** | **Usage** |
| --- | --- |
| Verbose | Verbose is the noisiest level, rarely (if ever) enabled for a production app. |
| Debug | Debug is used for internal system events that are not necessarily observable from the outside, but useful when determining how something happened. |
| Information | Information events describe things happening in the system that correspond to its responsibilities and functions. Generally these are the observable actions the system can perform. |
| Warning | When service is degraded, endangered, or may be behaving outside of its expected parameters, Warning level events are used. |
| Error | When functionality is unavailable or expectations broken, an Error event is used. |
| Fatal | The most critical level, Fatal events demand immediate attention. |

Most popular framework is SeriLog.

# Serilog:

## What is Serilog?

## Serilog is an excellent, easy to use and configure structured logging framework. It provides structured event logging and powerful support for structured diagnostic data. Serilog has great integration with all the major .NET frameworks, including ASP.NET Core, ASP.NET MVC, Entity Framework (RDBMS), EF Core, NHibernate, and more.

## Serilog is a logging library for .NET and C# that allows for more detailed and structured logging than the default .NET logging library. Serilog can be used to log information about application events, errors, and performance metrics. This information can be used to troubleshoot issues with an application or to monitor its performance over time.

## **Serilog has a few benefits for C# developers, including:**

## Simple to configure

## Flexible

## Supports structured logging

## **What are the Benefits of Serilog?**

## Serilog provides a more robust and flexible logging solution than the default .NET logging library. It can be used to track Application Insights events, log custom information, and generate rich log formatting. This makes it an ideal choice for applications that require detailed logging or for applications that are running in production environments.

## **What are Serilog Sinks?**

## Sinks in Serilog are the log targets (i.e., they are the destinations where you would like to send your logs). Some of the popular sinks are file and console targets. A Serilog sink is a destination for your logs. This can be a file, database, cloud service, etc.

## Anything that can store log data can be used as a Serilog sink. Serilog comes with several built-in sinks, but you can also create your own custom sinks. This allows you to send your log data to any destination that you want.

## **Enrichers:**

## Enrichers are simple components that add, remove, or modify the properties attached to a log event. This can be used for the purpose of attaching a thread id to each event, for example.

**Installing:**

Nuget:

Install Serilog.Asp.netCore

Install Serilog.sinks.File

On Program.cs:

builder.Host.UseSerilog((context, configuration) =>

configuration.ReadFrom.Configuration(context.Configuration));

//request logging not always required

app.UseSerilogRequestLogging();

On each Class we want to Log:

private ILogger<EmployeeController> Logger { get; }

public EmployeeController(IEmployeeService employeeService,

ILogger<EmployeeController> logger)

{

EmployeeService = employeeService;

Logger = logger;

}

//use:

Logger.LogInformation("This is an informational message.");

Configuration:

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"Serilog": {

"Using": [ "Serilog.Sinks.File", "Serilog.Sinks.Console" ],

"MinimumLevel": {

"Default": "Information",

"Override": {

"Microsoft": "Information",

"System": "Warning"

}

},

"WriteTo": [

{

"Name": "Console"

},

{

"Name": "File",

"Args": {

"path": "log-.txt",

"rollOnFileSizeLimit": true,

"formatter": "Serilog.Formatting.Compact.CompactJsonFormatter,Serilog.Formatting.Compact",

"rollingInterval": "Hour"

}

}

],

"Enrich": [ "FromLogContext", "WithThreadId", "WithMachineName" ]

}