



.NET Anwendungen richtig loggen

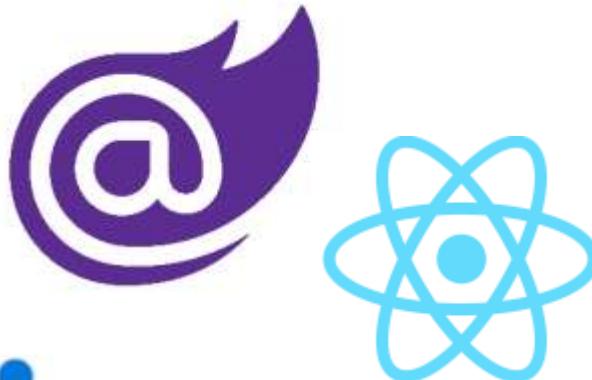
Wege und Möglichkeiten



Thorsten Kansy (tkansy@dotnetconsulting.eu)

Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor



Azure Cosmos DB



Mein Service- Ihr Benefit

- Individuelle Inhouse Trainings
- (Online on-demand) Projektbegleitung
- Beratung
 - Problemanalyse und Lösungen
 - Technologieentscheidungen





Warum überhaupt loggen?

Warum überhaupt loggen?

- Fehlersuche
- Fehlersuche
- Fehlersuche



Alternativen?



Gibt es Alternativen?

Eigentlich nicht...

Aber, wie immer, diese Praxis:

- Texte in Datei schreiben bei Ausnahmen
- E-Mails senden bei Ausnahmen
- Screenshots erstellen und per E-Mail versenden



Das richtige Maß

Das richtige Maß

- Aufruf einer wichtigen Funktion (keine Hilfsfunktionen)
- Bei Fehlern/ Ausnahmen
- Wenn wichtige Codepfade ablaufen (warum, welcher Codepfad?)
- So viel wie nötig, so wenig wie möglich
- Adressat beachten. Wer liest das Log?



Wie sensible Daten loggen?

Wie sensible Daten loggen?

- Achtung!
- Keine sensiblen Daten loggen!
 - (Wer kann das Log lesen?)
- EF Core hat dafür einen Konfigurationsschalter
 - Logt Parameter-Werte

The background image shows the interior of the Sagrada Família cathedral in Barcelona, featuring Gaudí's signature organic architecture with columns resembling tree trunks and a ceiling with a complex, star-like pattern of light fixtures. A solid blue rectangular overlay covers the lower half of the image, containing the text.

Wann Logging in den Code schreiben?

Wann Logging den Code schreiben?

- Direkt beim Coden
- Nachträglich ist es eine Qual und ungenau
- Vorsicht vor pauschalem Loggen (Attribute, PostSharp)



Logging-Provider

Logging

- 3 gute Gründe:
- Fehlersuche, Fehlersuche & Fehlersuche!

ILoggerFactory **Logger Factory**

ILogger **Logger Service**

```
Install-Package Microsoft.Extensions.Logging
```

```
Install-Package Microsoft.Extensions.Logging.Abstractions
```

Logging-Provider

Level	NuGet-Paket
Konsole	Microsoft.Extensions.Logging.Console
Json Konsole (structured)	Microsoft.Extensions.Logging.Console
Einfache Konsole (+Systemd)	Microsoft.Extensions.Logging.Console
Debugger-Monitor (Output window)	Microsoft.Extensions.Logging.Debug
Trace Listener (Output window)	Microsoft.Extensions.Logging.TraceSource
Windows-Ereignisprotokoll	Microsoft.Extensions.Logging.EventLog
EventSource/EventListener	Microsoft.Extensions.Logging.EventSource
Azure-App-Dienste „Diagnoseprotokolle“ und „Log Stream“	Microsoft.Extensions.Logging.AzureAppServices
Datei, Datenbank, SMTP etc.	Serilog.* // NLog.* // usw.
Custom	-

Logging konfigurieren

```
static void LoggingDemo1(string scopeName = null)
{
    // Logger Factory konfigurieren
    ILoggerFactory loggerFactory = new LoggerFactory()
        .AddConsole(LogLevel.Trace, true)
        .AddDebug();

    ILogger logger = loggerFactory.CreateLogger<Program>();

    demoLogging(logger, scopeName);
}
```



Demo



Logging-Level

Variante	Lebenszeit
Trace (0)	Ablaufverfolgung, die auch sensible Informationen enthalten kann
Debug (1)	Technische Debug-Informationen für die Fehlersuche
Information (2)	Nachverfolgung des allgemeinen Ablaufs der Anwendung
Warning (3)	Warnungen bei ungewöhnlichen oder unerwarteten Ereignissen
Error (4)	Fehler und Ausnahmen, die vom Code nicht behandelt werden können
Critical (5)	Für Fehler, die sofortige Aufmerksamkeit erfordern

```
... public enum LogLevel  
{  
    ... Trace = 0,  
    ... Debug = 1,  
    ... Information = 2,  
    ... Warning = 3,  
    ... Error = 4,  
    ... Critical = 5,  
    ... None = 6  
}
```



Demo

Logging Initialisierung



Logging & Dependency Injection

```
// Factory
serviceCollection.AddSingleton<ILoggerFactory, LoggerFactory>();

// Generischer Logger
serviceCollection.AddSingleton(typeof(ILogger<>), typeof(Logger<>));

public class AppController
{
    private readonly ILogger<AppController> _logger;

    public Application(ILogger<AppController> Logger)
    {
        _logger = Logger;
    }
}
```



Demo



Loggen

Logging in Action !

```
logger.Log(LogLevel.Trace, "Trace-Log");
logger.Log(LogLevel.Debug, "Debug-Log");
logger.Log(LogLevel.Information, "Information-Log");

logger.LogWarning("Warning-Log");
logger.LogError("Error-Log");
logger.LogCritical("Critical-Log");
```

Logging in Action II

```
// Wenn etwas schief geht
try
{
    throw new Exception("Passierschein A38 nicht gefunden.");
}
catch (Exception ex)
{
    logger.LogError(ex, "Error");
    throw;
}
```

Logging in Action III

```
// Wenn mal etwas richtig schief geht
try
{
    throw new Exception("Passierschein A38 nicht vorhanden.");
}
catch (Exception ex)
{
    logger.LogCritical(ProgramEventId.PermitNotFound, ex,
                      "Permit A38 not found");
    throw;
}
```

Logging in Action IV

```
// Oder wenn etwas nicht dort ist, wo es sein sollte
string importFilename = @"c:\nix\data.csv";
try
{
    string import = File.ReadAllText(importFilename);
}
catch (Exception ex)
{
    logger.LogCritical(ProgramEventId.ImportFileFailed, ex,
        "Import file '{0}' not ", importFilename);
}
```

String Interpolation

- Nicht so:

A screenshot of a code editor interface. At the top, the status bar shows '264' and 'No issues found'. Below it is a toolbar with icons for search, file operations, and code navigation. The main area displays the following code:

```
logger.LogWarning($"demoLogging({DateTime.Now})");
```

The code editor has a light blue background. The code itself is in a dark blue font, while the string interpolation placeholder '{DateTime.Now}' is highlighted in red. Below the code editor is the 'Error List' window, which has a dark blue header. It shows the following message:

Entire Solution 0 Errors 0 Warnings 1 Message Build + IntelliSense

Code Description

CA2254 The logging message template should not vary between calls to 'LoggerExtensions.LogWarning(I`Logger`, string?, params object?[])'

- Besser so:

```
logger.LogWarning("demoLogging({Now})", DateTime.Now);
```

Compile-time logging source generation

Logging per Attribut auf Interfaces oder partiellen Klassen

```
public static partial class MyClass
{
    [LoggerMessage( EventId = 0, Level = LogLevel.Critical, Message = "Could not open socket to '{hostName}'")]
    public static partial void CouldNotOpenSocket(ILogger logger, string hostName);
}

public static interface IMyInterface
{
    [LoggerMessage( EventId = 47, Level = LogLevel.Critical, Message = "Could not open socket to '{hostName}'")]
    static void CouldNotOpenSocket(ILogger logger, string hostName);
}
```



Demo



Logging Filter

Logging Filter

- Namespace
- Min Log Level

Konfiguration in appsettings.json

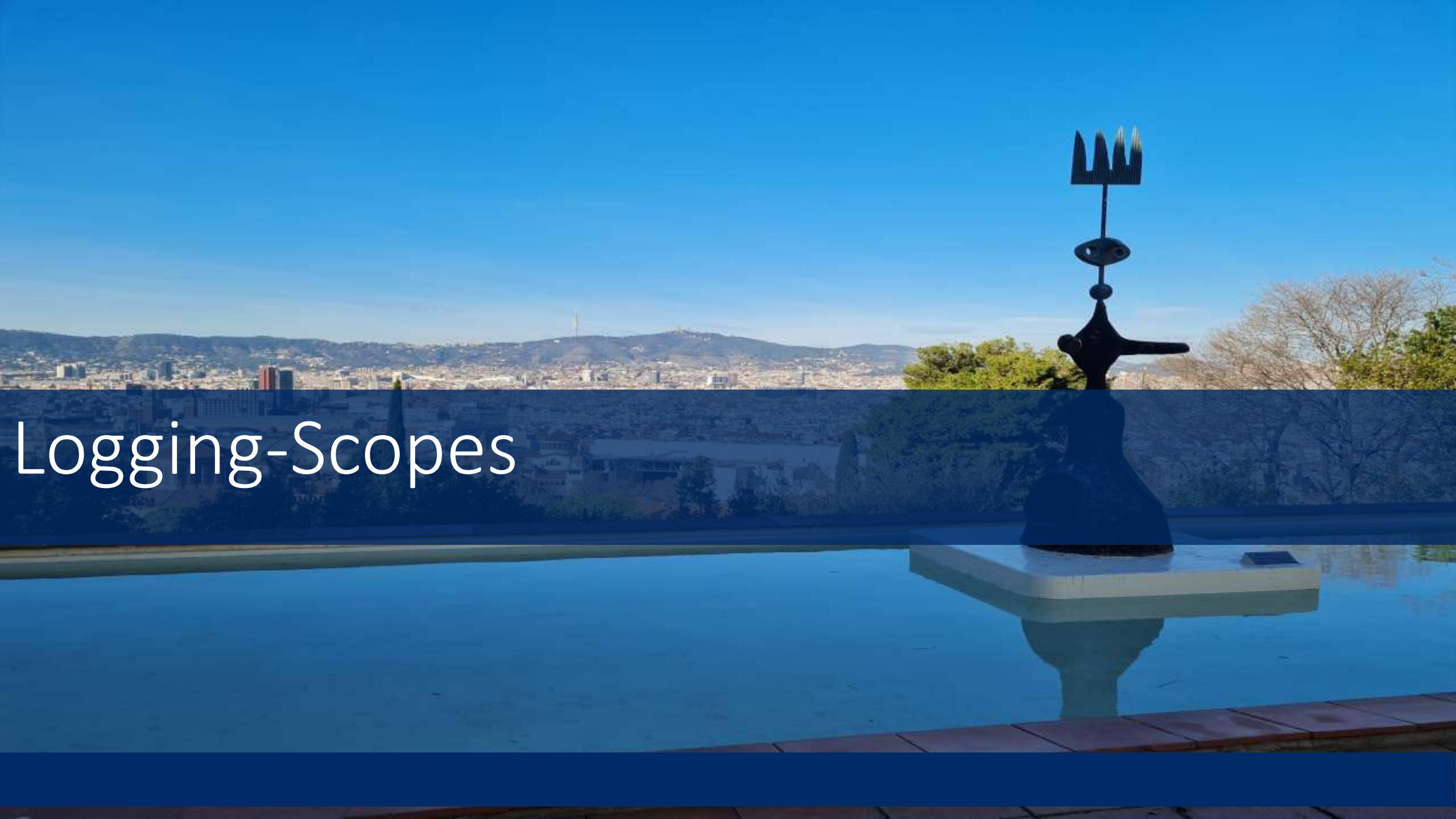
```
JSON Kopieren

{
  "Logging": {
    "LogLevel": { // All providers, LogLevel applies to all the enabled providers.
      "Default": "Error", // Default logging, Error and higher.
      "Microsoft": "Warning" // All Microsoft* categories, Warning and higher.
    },
    "Debug": { // Debug provider.
      "LogLevel": {
        "Default": "Information", // Overrides preceding LogLevel:Default setting.
        "Microsoft.Hosting": "Trace" // Debug:Microsoft.Hosting category.
      }
    },
    "EventSource": { // EventSource provider
      "LogLevel": {
        "Default": "Warning" // All categories of EventSource provider.
      }
    }
  }
}
```



Demo

Logging-Sscopes



Logging-Scopes

Bereich im Logging zwecks besserer Lesbarkeit

```
using (logger.BeginScope($"Scope y = {y}"))
{
    for (int z = 0; z < 2; z++)
    {
        logger.LogDebug($"y = {y}, z = {z}");
    }
}
```



Demo



EventIDs

EventIds

```
public static class ProgramEventId
{
    public static readonly EventId CriticalSituation201
        = new EventId(201, "CriticalSituation201");

    public static readonly EventId PermitNotFound
        = new EventId(202, "PermitNotFound");

    public static readonly EventId ImportFileFailed
        = new EventId(203, "ImportFileFailed");
}
```



Demo

A wide-angle photograph of a tropical beach under a bright blue sky with scattered white clouds. In the foreground, a wooden swing set is visible on a sandy beach. The ocean extends to the horizon. A large green leafy branch is partially visible on the left side of the frame.

Eigene Logging-Provider schreiben

Ein Beispiel

Code-Klasse	Aufgabe
ColorConsoleLoggingExtentions	Erweiterung der Fluent API
ColorConsoleLogger	Der eigentliche Logger, der das ILogger-Interface implementiert.
ColorConsoleLoggerConfiguration	Die Konfiguration die der Logger verwendet. Dies ist ein schlichtes POCO
ColorConsoleLoggingProvider	Der Provider, der die benötigten Instanzen des Loggers erzeugt und das ILoggerProvider-Interface implementiert.



Demo



Logging-Provider von Drittherstellern

Logging-Provider von Drittherstellern

- Nlog
- SeriLog
- Log4Net
- ...



Demo

NLog – Fully-featured Logging Framework



Welcome to NLog!

NLog is a flexible and free logging platform for various .NET platforms, including .NET standard. NLog makes it easy to write to several [targets](#), (database, file, console) and change the logging configuration on-the-fly.

NLog has support for [structured](#) and traditional logging.

The focus for NLog: high performance, easy-to-use, easy to extend and flexible to configure.

Features

Easy to configure

NLog is very easy to configure, both through [configuration file](#) and [programmatic](#). Even without restarting the application, the configuration can be changed.

Templatable

Every log message can be templated with various [layout renderers](#).

Extensible

NLog already includes several targets and pre-defined layouts, but you can also extend with your own [custom targets](#) and [layouts](#) or include own [custom context values](#).



<http://nlog-project.org>

Install-Package NLog

Install-Package NLog.Extensions.Logging



Demo

Fragen? Jetzt oder später!



Kontakt

 E-Mail

tkansy@dotnetconsulting.eu

 LinkedIn

[Link me](#)

 Telefon

+49 (0) 6187 / 2009090

 XING

[Xing me](#)

 Microsoft Teams

[Meet now](#)

 X (Twitter)

@tkansy



www.dotnetconsulting.eu

SQL Server meets .NET (Core)- professionally!



Ich berate, coache und trainiere im Bereich Entwicklung von .NET (Core) Anwendungen mit Microsoft SQL Server- mit Allem, was dazu gehört- und was man vielleicht weglassen sollte.