

ASP.NET CORE LOGGING UND DOKUMENTATION



- Sourcen mit Beispielen zum Skript finden sie unter [florianwachs/AspNetWebServicesCourse \(github.com\)](https://github.com/florianwachs/AspNetWebServicesCourse)

LOGGING

WARUM IST LOGGING SO WICHTIG

- In der Produktionsumgebung lässt es sich meist schwer debuggen
- Snapshots sind meist aufwändig zu erstellen
- Gutes Logging bietet meist zumindest einen Anhaltspunkt wo die Analyse beginne sollte (Stacktrace)

LOGGING

```
var builder = WebApplication.CreateBuilder(args);  
  
// Standard mäßig werden folgende Logger konfiguriert (defaults)  
// Console  
// Debug  
// EventSource  
// EventLog (nur auf Windows)  
  
// Die Konfiguration kann aber beliebig angepasst werden  
// ConfigureOnlyConsole(builder);  
  
var app = builder.Build();
```

LOGGING

```
void ConfigureOnlyConsole(WebApplicationBuilder builder)
{
    builder.Logging.ClearProviders();
    builder.Logging.AddConsole();
}
```

Konfiguration

Die Default Konfiguration für das Logging lässt sich vollständig anpassen.
ClearProviders() entfernt alle defaults

LOGGING

```
app.MapGet("/", IndexRouteHandler);  
app.MapGet("/structural", StructuralLoggingRouteHandler);  
  
app.Run();  
  
string IndexRouteHandler(ILoggerFactory loggerFactory)  
{  
    var logger = loggerFactory.CreateLogger(nameof(IndexRouteHandler));  
  
    return "Hello World";  
}
```

Logger erzeugen

Eine Instanz eines Loggers lässt sich über eine LoggerFactory erzeugen. Der Logger erhält einen Namen, üblicherweise die Klasse in der er verwendet wird. Damit werden Logmeldungen angereichert.

LOGGING

```
string ConcreteDiHandler(ILogger<Program> logger)
{
    logger.LogInformation("Berechne komplexe Nachricht");

    return "Hello World";
}
```

Logger erzeugen

Über das DI-System kann auch direkt eine konkrete Instanz für einen Typ erzeugt werden. Als „Typ“ wird meist die umgebende Klasse verwendet.

LOGGING

```
public class ValuesController : Controller
{
    private ILogger Logger { get; set; }

    // Logger manuell über Factory erzeugen
    public ValuesController(ILoggerFactory logger)
    {
        Logger = logger.CreateLogger<ValuesController>();
    }

    // Oder Logger vom DI-System erzeugen lassen
    public ValuesController(ILogger<ValuesController> logger)
    {
        Logger = logger;
    }
}
```

Logger erzeugen

Hier ein Beispiel für einen ApiController

LOGLEVEL

- LogTrace
 - Für maximal möglichen Informationsgehalt.
- LogDebug
 - Sollte Informationen enthalten die einen Mehrwert während der Entwicklungsphase bieten
- LogInformation
 - Geben Auskunft über aktuelle Abläufe innerhalb der Applikation
- LogWarning => Standard-LogLevel für Produktion
 - Für abnormales / unerwartetes Verhalten welches aber nicht den Ablauf der Applikation stoppt
- LogError
 - Für Fehler welche die aktuell ausgeführte Tätigkeit betreffen
- LogCritical
 - Für die schlimmsten Fälle welche sofortiges eingreifen erfordern, wie z.B.: Applikationscrash, Systemausfall

LOGGING

```
app.MapGet("/", IndexRouteHandler);
```

```
app.Run();
```

```
string IndexRouteHandler ILoggerFactory loggerFactory)
{
    var logger = loggerFactory.CreateLogger(nameof(IndexRouteHandler));
    logger.LogTrace("Im Handler der Index route");
    logger.LogDebug("Hoffentlich klappt es");
    logger.LogInformation("Berechne komplexe Nachricht");
    logger.LogWarning("Berechnung dauert länger als erwartet....");
    logger.LogError("Der Microservice für die Berechnung ist schon wieder down....");
    logger.LogCritical("Hat da grad jemand Cola in den Server verschüttet?");

    return "Hello World";
}
```

```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7036
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5275
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\src\github\AspNetWebservicesCourse\modules\aspnet_logging\lessons\integrated\BasicLogging\
info: IndexRouteHandler[0]
      Berechne komplexe Nachricht
warn: IndexRouteHandler[0]
      Berechnung dauert länger als erwartet....
fail: IndexRouteHandler[0]
      Der Microservice für die Berechnung ist schon wieder down....
crit: IndexRouteHandler[0]
      Hat da grad jemand Cola in den Server verschüttet?
```

LOGLEVEL

LogLevel = Information

```
Microsoft Visual Studio Debug Console
+
-
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7036
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5275
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\src\github\AspNetWebServicesCourse\modules\aspnet_logging\lessons\integrated\BasicLogging\
info: IndexRouteHandler[0]
      Berechne komplexe Nachricht
warn: IndexRouteHandler[0]
      Berechnung dauert länger als erwartet....
fail: IndexRouteHandler[0]
      Der Microservice für die Berechnung ist schon wieder down....
crit: IndexRouteHandler[0]
      Hat da grad jemand Cola in den Server verschüttet?

C:\src\github\AspNetWebServicesCourse\modules\aspnet_logging\lessons\integrated\BasicLogging\bin\Debug\net6.0\BasicLogging.exe (p
rocess 34064) exited with code -1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when d
ebugging stops.
Press any key to close this window . . .|
```

LOGLEVEL

LogLevel = Trace

```
C:\src\github\AspNetWebsern x + v
dbug: Microsoft.Extensions.Hosting.Internal.Host[1]
      Hosting starting
dbug: Microsoft.AspNetCore.Server.Kestrel.Core.KestrelServer[0]
      Using development certificate: CN=localhost (Thumbprint: 822E37DE5DE8DF3CD5CS1BF1A95EE32A83A43C)
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7036
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5275
dbug: Microsoft.AspNetCore.Hosting.Diagnostics[13]
      Loaded hosting startup assembly BasicLogging
dbug: Microsoft.AspNetCore.Hosting.Diagnostics[13]
      Loaded hosting startup assembly Microsoft.AspNetCore.Watch.BrowserRefresh
dbug: Microsoft.AspNetCore.Hosting.Diagnostics[13]
      Loaded hosting startup assembly Microsoft.WebTools.BrowserLink.Net
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\src\github\AspNetWebServicesCourse\modules\aspnet_logging\lessons\integrated\BasicLogging\
dbug: Microsoft.Extensions.Hosting.Internal.Host[2]
      Hosting started
dbug: Microsoft.AspNetCore.Server.Kestrel.Transport.Sockets[6]
      Connection id "0HMHK3CEF5I0H" received FIN.
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[39]
      Connection id "0HMHK3CEF5I0H" accepted.
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[1]
      Connection id "0HMHK3CEF5I0H" started.
dbug: Microsoft.AspNetCore.Server.Kestrel.Https.Internal.HttpsConnectionMiddleware[1]
      Failed to authenticate HTTPS connection.
      System.IO.IOException: Received an unexpected EOF or 0 bytes from the transport stream.
         at System.Net.Security.SslStream.ReceiveBlobAsync[TIOAdapter](TIOAdapter adapter)
         at System.Net.Security.SslStream.ForceAuthenticationAsync[TIOAdapter](TIOAdapter adapter, Boolean receiveFirst, Byte[] reAuthenticationData, Boolean isApm)
         at Microsoft.AspNetCore.Server.Kestrel.Https.Internal.HttpsConnectionMiddleware.OnConnectionAsync(ConnectionContext context)
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[2]
      Connection id "0HMHK3CEF5I0H" stopped.
dbug: Microsoft.AspNetCore.Server.Kestrel.Transport.Sockets[7]
      Connection id "0HMHK3CEF5I0H" sending FIN because: "The Socket transport's send loop completed gracefully."
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[39]
      Connection id "0HMHK3CEF5I0I" accepted.
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[1]
      Connection id "0HMHK3CEF5I0I" started.
dbug: Microsoft.AspNetCore.Server.Kestrel.Https.Internal.HttpsConnectionMiddleware[3]
      Connection 0HMHK3CEF5I0I established using the following protocol: Tls13
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[49]
      Connection id "0HMHK3CEF5I0I" sending SETTINGS frame for stream ID 0 with length 18 and flags NONE.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[49]
      Connection id "0HMHK3CEF5I0I" sending WINDOW_UPDATE frame for stream ID 0 with length 4 and flags 0x0.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[37]
      Connection id "0HMHK3CEF5I0I" received SETTINGS frame for stream ID 0 with length 24 and flags NONE.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[49]
      Connection id "0HMHK3CEF5I0I" sending SETTINGS frame for stream ID 0 with length 0 and flags ACK.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[37]
      Connection id "0HMHK3CEF5I0I" received WINDOW_UPDATE frame for stream ID 0 with length 4 and flags 0x0.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[37]
      Connection id "0HMHK3CEF5I0I" received HEADERS frame for stream ID 1 with length 478 and flags END_STREAM, END_HEADERS, PRIORITY.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[37]
      Connection id "0HMHK3CEF5I0I" received SETTINGS frame for stream ID 0 with length 0 and flags ACK.
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/2 GET https://localhost:7036/ - -
dbug: Microsoft.AspNetCore.Hosting.Filtering.HostFilteringMiddleware[0]
      Wildcard detected, all requests with hosts will be allowed.
trce: Microsoft.AspNetCore.Hosting.Filtering.HostFilteringMiddleware[2]
      All hosts are allowed.
dbug: Microsoft.AspNetCore.Routing.Matching.DfaMatcher[1001]
      1 candidate(s) found for the request path '/'
dbug: Microsoft.AspNetCore.Routing.EndpointRoutingMiddleware[1]
      Request matched endpoint 'HTTP: GET / => IndexRouteHandler'
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[0]
      Executing endpoint 'HTTP: GET / => IndexRouteHandler'
trce: IndexRouteHandler[0]
      Im Handler der Index route
dbug: IndexRouteHandler[0]
      Öffentlich klappt es
info: IndexRouteHandler[0]
      Berechne komplexe Nachricht
warn: IndexRouteHandler[0]
      Berechnung dauert länger als erwartet.....
crit: IndexRouteHandler[0]
      Der Microservice für die Berechnung ist schon wieder down....
crit: IndexRouteHandler[0]
      Hat da grad jemand Cola in den Server verschüttet?
|
```

LOGLEVEL

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  }
}
```

LogLevel

Das Logging-Framework erlaubt unterschiedliche LogLevel pro (Teil-) Kategorie. Die Kategorie entspricht meist dem Namespace.

LOGGING PROVIDER

- Debug-Window
- Console
- EventSource (ETW auf Windows)
- EventLog (Windows)

3RD PARTY LOGGING PROVIDER

- Das Logging-System von ASP.NET Core ist erweiterbar.
- **Serilog**
- Log4net
- Nlog

PROBLEME VON „NORMALEM“ LOGGING

- Output ist Text
- Zum Parsen werden meist reguläre Ausdrücke verwendet
- Queries gegen Logfiles nur nach Aufbereitung / Parsen der Logfiles möglich
- Die „Semantik“ eines Logeintrags geht verloren
- Problem verschlimmert sich je mehr Logfiles analysiert werden müssen

SEMANTIC LOGGING

- Zeig mir alle Meldungen die Produkt 123 betreffen
- Zeige mir alle fehlgeschlagenen Login-Versuche des Users „Chuck“
- Zeig mir alle Requests deren Laufzeit 500ms überstieg
- Wie oft wurde Operation X heute aufgerufen

PROBLEME VON „NORMALEM“ LOGGING

- Log ohne Semantik

```
2017-05-17 19:00:53.706 +02:00 [Information] Request starting HTTP/1.1 GET http://localhost:24976/api/values
2017-05-17 19:00:54.092 +02:00 [Information] Request finished in 406.5738ms 404
2017-05-17 19:01:00.383 +02:00 [Information] Request starting HTTP/1.1 GET http://localhost:24976/api/books
ModelState is Valid
2017-05-17 19:01:05.052 +02:00 [Warning] No book was found by the provided Id 200.
2017-05-17 19:01:05.064 +02:00 [Information] Executing HttpStatusCodeResult, setting HTTP status code 404
2017-05-17 19:01:05.069 +02:00 [Information] Executed action "AspNetCore.Logging.Serilog.Controllers.BooksController.GetBookById
(AspNetCore.Logging.Serilog)" in 119.036ms
2017-05-17 19:01:05.075 +02:00 [Information] Request finished in 209.2111ms 404
2017-05-17 19:01:07.371 +02:00 [Information] Request starting HTTP/1.1 POST http://localhost:24976/api/books/200 application/json
163
2017-05-17 19:01:07.393 +02:00 [Information] Request finished in 23.5851ms 404
2017-05-17 19:01:08.807 +02:00 [Information] Request starting HTTP/1.1 POST http://localhost:24976/api/books/200 application/json
163
2017-05-17 19:01:08.826 +02:00 [Information] Request finished in 19.98ms 404
2017-05-17 19:01:16.080 +02:00 [Information] Request starting HTTP/1.1 POST http://localhost:24976/api/books application/json 163
2017-05-17 19:01:16.214 +02:00 [Information] Executing action method
"AspNetCore.Logging.Serilog.Controllers.BooksController.CreateBook (AspNetCore.Logging.Serilog)" with arguments
(["AspNetCore.Logging.Serilog.Models.Book"]) - ModelState is Invalid
2017-05-17 19:01:18.550 +02:00 [Error] The supplied data for book creation is invalid (Book { Id: 1, Isbn: "", Title: "", Price:
30, Authors: ["Troelson"], ReleaseDate: 05/16/2015 20:37:28 }) with Errors "The Isbn field is required.;The Title field is
required.".
2017-05-17 19:01:18.568 +02:00 [Information] Executing ObjectResult, writing value "Microsoft.AspNetCore.Mvc.ControllerContext".
2017-05-17 19:01:18.583 +02:00 [Information] Executed action "AspNetCore.Logging.Serilog.Controllers.BooksController.CreateBook
(AspNetCore.Logging.Serilog)" in 2487.6788ms
2017-05-17 19:01:18.592 +02:00 [Information] Request finished in 2512.2555ms 400 application/json; charset=utf-8
```

SEMANTIC LOGGING

- Log mit Semantik

```
{
  "Timestamp": "2017-05-17T19:01:58.1796327+02:00",
  "Level": "Error",
  "MessageTemplate": "The supplied data for book creation is invalid ({@Book}) with Errors {Errors}.",
  "RenderedMessage": "The supplied data for book creation is invalid (Book { Id: 1, Isbn: \"\", Title: \"\", Price: 30, Authors: [\\\"Troelson\\\"], ReleaseDate: 05/16/2015 20:37:28 }) with Errors \\\"The Isbn field is required.;The Title field is required.\\\".",
  "Properties": {
    "Book": {
      "_typeTag": "Book",
      "Id": 1,
      "Isbn": "",
      "Title": "",
      "Price": 30,
      "Authors": [ "Troelson" ],
      "ReleaseDate": "2015-05-16T20:37:28.3436361+02:00"
    },
    "Errors": "The Isbn field is required.;The Title field is required.",
    "SourceContext": "AspNetCore.Logging.Serilog.Controllers.BooksController",
    "ActionId": "56d7134f-25ae-4f20-a0f5-4b390908e337",
    "ActionName": "AspNetCore.Logging.Serilog.Controllers.BooksController.CreateBook (AspNetCore.Logging.Serilog)",
    "RequestId": "0HL4T8PK97VEP",
    "RequestPath": "/api/books",
    "Release": "0.0.1-beta-nightmare"
  }
}
```

SEMANTIC LOGGING

The screenshot displays the Seq application interface. The top bar shows the Seq logo and navigation links: events, dashboards, help, settings, and admin. The main area is divided into two panes. The left pane shows a list of log events with timestamps and descriptions. The right pane shows a detailed view of a selected event, including its metadata and raw JSON data.

Log Events List:

- 17 May 2017 18:50:19.078 Request finished in 72.2613ms 404
- 17 May 2017 18:50:19.071 Executed action ASP.NETCore.Logging.Serilog.Controllers.BooksController.GetBookById (ASP.NETCore.Logging.Serilog) in 51.227000000000004ms
- 17 May 2017 18:50:19.066 Executing HttpStatusCodeResult, setting HTTP status code 404
- 17 May 2017 18:50:19.055 No book was found by the provided Id 200.
- 17 May 2017 18:50:19.048 Executing action method ASP.NETCore.Logging.Serilog.Controllers.BooksController.GetBookById (ASP.NETCore.Logging.Serilog) with arguments ("200") -...
- 17 May 2017 18:50:19.006 Request starting HTTP/1.1 GET http://localhost:24976/api/books/200 application/json
- 17 May 2017 18:49:01.546 Request finished in 4373.7321ms 400 application/json; charset=utf-8
- 17 May 2017 18:49:01.538 Executed action ASP.NETCore.Logging.Serilog.Controllers.BooksController.CreateBook (ASP.NETCore.Logging.Serilog) in 4230.7713ms
- 17 May 2017 18:49:01.521 Executing ObjectResult, writing value Microsoft.AspNetCore.Mvc.ControllerContext.
- 17 May 2017 18:48:59.892 The supplied data for book creation is invalid (Book{"Id":1,"Isbn":"","Title":"","Price":30,"Authors":["Troelson"],"ReleaseDate":"2015-05-16T2...}) with Errors

Error Event Details:

- Id:** df011457-62ea-499b-805e-adea7a96a124
- ActionName:** ASP.NETCore.Logging.Serilog.Controllers.BooksController.CreateBook (ASP.NETCore.Logging.Serilog)
- Book:** {"Id": 1, "Isbn": "", "Title": "", "Price": 30, "Authors": [...], "ReleaseDate": "2015-05-16T20:37:28.3436361+02:00", "_typeTag": "Book"}
- Errors:** The Isbn field is required.;The Title field is required.
- Release:** 0.0.1-beta-nightmare
- RequestId:** 0HL4T8HRHN4A5
- RequestPath:** /api/books
- SourceContext:** ASP.NETCore.Logging.Serilog.Controllers.BooksController

Right Sidebar:

- SIGNALS:** None, Errors, Exceptions, Warnings
- QUERIES:** Available Properties, Count by Hour, Latest as Table, Space by Event Type

Footer: Seq 4.0.58 Single-User License - Copyright © 2017 Enable update check

Tools wie Kibana oder Seq können semantische Logs verarbeiten und analysieren. Es können SQL ähnliche Abfragen durchgeführt werden.

SERIOLOG

SERilog|Installation

```
<PackageReference Include="Serilog.AspNetCore" Version="5.0.0" />  
<PackageReference Include="Serilog.Sinks.Console" Version="4.0.1" />  
<PackageReference Include="Serilog.Sinks.File" Version="5.0.0" />
```

SERilog|Sinks

- Sinks definieren das Ziel der Log-Aufrufe und deren Formatierung
- Sinks sind als NuGet-Pakete verfügbar
- Typische Sinks
 - Serilog.Sinks.Console
 - Serilog.Sinks.File
- [Weitere Sinks https://github.com/serilog/serilog/wiki/Provided-Sinks](https://github.com/serilog/serilog/wiki/Provided-Sinks)

SERilog|Konfiguration

Mit „Enrich-ern“ können den Logmeldungen zusätzliche Informationen mitgegeben werden. Auch die Enricher werden über NuGet-Pakete bereitgestellt.

```
Serilog.Enrichers.Environment => WithMachineName()  
,WithEnvironmentUserName()  
Serilog.Enrichers.Process => WithProcessId()  
Serilog.Enrichers.Thread => WithThreadId()
```

```
void ConfigureSerilog(WebApplicationBuilder builder)  
{  
    // Default Logger entfernen  
    builder.Logging.ClearProviders();  
  
    // Serilog verwendet eine eigene Konfiguration, kann auch in der  
    // appsettings.json angegeben werden,  
    // hat jedoch ein anderes Format als die Default Logger  
    var logger = new LoggerConfiguration()  
        .WriteTo.Console()  
        .CreateLogger();  
  
    // Serilog als Logger  
    builder.Logging.AddSerilog(logger);  
}
```

An WriteTo werden die gewünschten Sinks registriert. Jedes Sink hat in der Regel eine sinnvolle Default-Konfiguration und zusätzlich weitere Konfigurationsmöglichkeiten. Bei machen Sinks müssen Authentifizierungsinformationen oder API-Keys hinterlegt werden damit sie korrekt funktionieren.

CreateLogger erzeugt aus der Konfiguration einen Logger, welcher als Basis für weitere Logger-Instanzen verwendet werden kann.

SERilog|Konfiguration

```
Log.Logger = new LoggerConfiguration()  
    .ReadFrom.Configuration(Configuration)  
    .CreateLogger();
```

```
{  
  "Serilog": {  
    "MinimumLevel": {  
      "Default": "Debug",  
      "Override": {  
        "System": "Information",  
        "Microsoft": "Information"  
      }  
    },  
    "Enrich": [ "FromLogContext" ]  
  }  
}
```

Für die Extension-Method `Configuration()` an `ReadFrom` wird das NuGet-Paket `Serilog.Settings.Configuration` benötigt

SERilog|Konfiguration

```
// Globalen Serilog-Logger konfigurieren
Log.Logger = new LoggerConfiguration()
    .ReadFrom.Configuration(Configuration)
    .WriteTo. ...("Log-{Date}.txt")
    .Enrich.WithProperty("Release", "0.0.1-beta-nightmare")
    .CreateLogger();
```

Konfigurationsdatei und Konfiguration
per Code können gemischt werden

```
{
  "Serilog": {
    "MinimumLevel": {
      "Default": "Verbose",
      "Override": {
        "System": "Information",
        "Microsoft": "Information"
      }
    },
    "WriteTo": [
      { "Name": "LiterateConsole" }
    ],
    "Enrich": [ "FromLogContext" ]
  }
}
```