

ASP.NET CORE LOGGING UND DOKUMENTATION



- Quellen 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();
}
```

LOGGING

```
public void ConfigureLogging(ILoggerFactory loggerFactory)
{
    // Ein Logger kann manuell mit CreateLogger erzeugt werden
    // Der TypParameter definiert die Category unter der geloggt wird
    var startupLogger = loggerFactory.CreateLogger("AspNetCore.Logging.Startup");

    var startupLogger = loggerFactory.CreateLogger(typeof(Startup));

    var startupLogger = loggerFactory.CreateLogger<Startup>();

    // Log[LogLevel]
    startupLogger.LogInformation("configuration phase");
    app.UseMvc();
    startupLogger.LogInformation("request pipeline successfully configured at {TimeStamp}", DateTime.Now);
}
```

LOGGING

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

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

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


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";
}
```

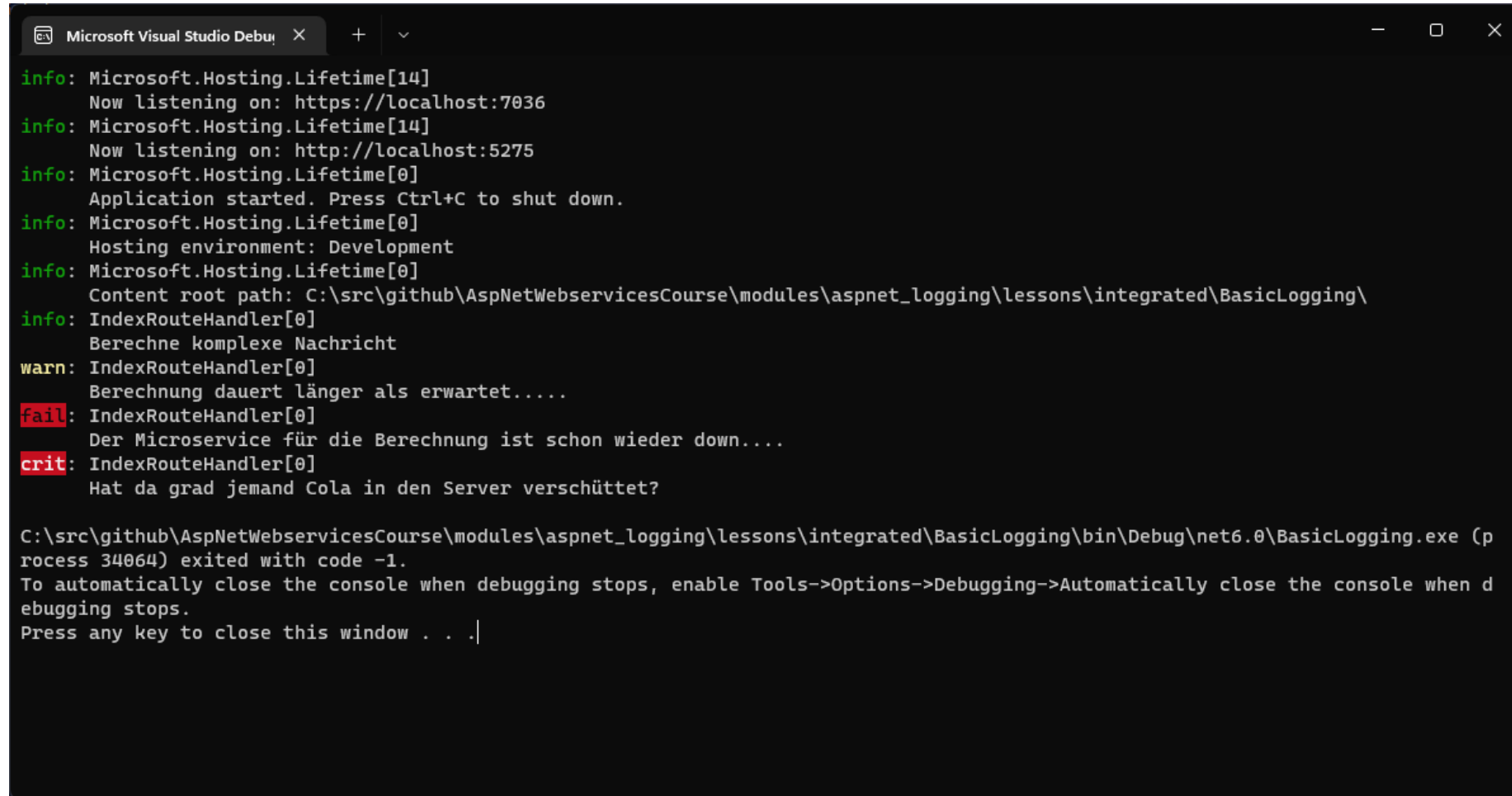
```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7036
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5275
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      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

- 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

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\AspNetWebser... 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: 8222E37DE50E8D8F3CD5C51BF1A95EE32A83A43C)
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 "0HMHK3CEF5IOH" received FIN.
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[39]
      Connection id "0HMHK3CEF5IOH" accepted.
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[1]
      Connection id "0HMHK3CEF5IOH" 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 "0HMHK3CEF5IOH" stopped.
dbug: Microsoft.AspNetCore.Server.Kestrel.Transport.Sockets[7]
      Connection id "0HMHK3CEF5IOH" sending FIN because: "The Socket transport's send loop completed gracefully."
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[39]
      Connection id "0HMHK3CEF5IOI" accepted.
dbug: Microsoft.AspNetCore.Server.Kestrel.Connections[1]
      Connection id "0HMHK3CEF5IOI" started.
dbug: Microsoft.AspNetCore.Server.Kestrel.Https.Internal.HttpsConnectionMiddleware[3]
      Connection 0HMHK3CEF5IOI established using the following protocol: Tls13
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[49]
      Connection id "0HMHK3CEF5IOI" sending SETTINGS frame for stream ID 0 with length 18 and flags NONE.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[49]
      Connection id "0HMHK3CEF5IOI" sending WINDOW_UPDATE frame for stream ID 0 with length 4 and flags 0x0.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[37]
      Connection id "0HMHK3CEF5IOI" received SETTINGS frame for stream ID 0 with length 24 and flags NONE.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[49]
      Connection id "0HMHK3CEF5IOI" sending SETTINGS frame for stream ID 0 with length 0 and flags ACK.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[37]
      Connection id "0HMHK3CEF5IOI" received WINDOW_UPDATE frame for stream ID 0 with length 4 and flags 0x0.
trce: Microsoft.AspNetCore.Server.Kestrel.Http2[37]
      Connection id "0HMHK3CEF5IOI" 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 "0HMHK3CEF5IOI" 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.Diagnostics[0]
      Wildcard detected, all requests with hosts will be allowed.
trce: Microsoft.AspNetCore.Hosting.Diagnostics[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]
      Hoffentlich klappt es
info: IndexRouteHandler[0]
      Berechne komplexe Nachricht
warn: IndexRouteHandler[0]
      Berechnung dauert länger als erwartet.....
err: IndexRouteHandler[0]
      Der Microservice für die Berechnung ist schon wieder down....
err: 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 shows the Seq logging tool interface. The top bar includes navigation icons and the address bar shows 'localhost:5341/#/events'. The main area displays a list of log events on the left and a detailed view of a selected event on the right. The selected event is an error message: 'The supplied data for book creation is invalid (Book{"Id":1,"Isbn":"","Title":"","Price":30,"Authors":["Troelson"],"ReleaseDate":"2015-05-16T20:37:28.3436361+02:00", "_typeTag": "Book"}) with Errors The Isbn field is required.;The Title field is required..'. The detailed view shows the event's metadata and the JSON payload.

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

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" ]
  }
}
```

OPEN API

Open API / SWAGGER

- Swagger / Open API definiert eine Tool-Chain zur automatisierten Beschreibung von APIs und deren Dokumentation
- Auf Basis des Swagger-Protokolls können automatisiert SDKs generiert werden
- NuGet: Swashbuckle.AspNetCore

Open API / SWAGGER

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddRouting(o =>
{
    o.LowercaseUrls = true; // Api Controller Namen klein ausgeben
});

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new() { Title = "Jokes API", Version = "v1" });

    // C# XML-Kommentare für API-Beschreibung nutzen
    // ACHTUNG: Funktioniert nur für Controller aktuell
    var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
    var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
    c.IncludeXmlComments(xmlPath);
});

var app = builder.Build();

// Swagger Endpunkt nur in DEV aktivieren
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.Run();
```

Open API / SWAGGER

Damit XML-Kommentare in die Swagger-API generiert werden, muss in der *.csproj die Generierung aktiviert werden.

```
<PropertyGroup>  
  <GenerateDocumentationFile>true</GenerateDocumentationFile>  
  <NoWarn>$(NoWarn);1591</NoWarn>  
</PropertyGroup>
```

Open API / SWAGGER

```
[Route("api/[controller]")]
[ApiController]
[Produces("application/json")]
public class BooksController : ControllerBase
{
    /// <summary>
    /// Returns a single book
    /// </summary>
    /// /// <remarks>
    /// Sample request:
    ///
    ///     GET /books/1
    ///
    /// </remarks>
    /// <param name="id">id of the book</param>
    /// <returns>the requested book</returns>
    [HttpGet("{id}")]
    public async Task<ActionResult<Book>> GetById(string id)
    {
        var book = await _bookRepository.GetById(id);
        if (book == null)
        {
            return NotFound();
        }

        return Ok(book);
    }
}
```

Open API / SWAGGER

```
/// <summary>
/// Updates a book partially by the provided properties
/// </summary>
/// <param name="id">id of the book to update</param>
/// <param name="doc">operations to perform</param>
/// <returns></returns>
/// <response code="200">The updated book</response>
/// <response code="400">If the validation fails</response>
/// <response code="404">no book found to update</response>
[HttpPatch("{id}")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public async Task<ActionResult<Book>> PartialUpdate(...)
{
    var existing = await _bookRepository.GetById(id);
    if (existing == null)
    {
        return NotFound();
    }

    doc.ApplyTo(existing);
    var result = await _bookRepository.Update(existing);

    return Ok(result);
}
```

Responses

Code	Description	Links
200	<div>The updated book</div> <div>Media type application/json</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>{ "id": "string", "title": "string", "description": "string", "categories": [{ "id": "string", "name": "string" }], "author": { "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6", "firstName": "string", "lastName": "string", "books": ["string"] } }</pre></div> <div>No links</div>	No links
400	<div>If the validation fails</div> <div>Media type application/json</div> <div>Example Value Schema</div> <div><pre>{ "type": "string", "title": "string", "status": 0, "detail": "string", "instance": "string", "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" }</pre></div> <div>No links</div>	No links
404	<div>no book found to update</div> <div>Media type application/json</div> <div>Example Value Schema</div> <div><pre>{ "type": "string", "title": "string", "status": 0, "detail": "string", "instance": "string", "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" }</pre></div> <div>No links</div>	No links

Open API / SWAGGER

Jokes API ^{v1}

/swagger/v1/swagger.json

Authors



GET /api/Authors Gets all authors

Books



GET /api/Books Returns all books

POST /api/Books Creates a new book

GET /api/Books/{id} Returns a single book

PUT /api/Books/{id} Complete update of a book

DELETE /api/Books/{id} removes a book

PATCH /api/Books/{id} Updates a book partially by the provided properties

Open API / SWAGGER

Books

GET

/api/Books

Returns all books

POST

/api/Books

Creates a new book

Parameters

Try it out

Name	Description
book <small>(body)</small>	<div>data for the book to create</div> <div>Example Value Model</div> <pre>{ "id": "string", "title": "string", "description": "string", "categories": [{ "id": "string", "name": "string" }], "author": { "id": "string", "firstName": "string", "lastName": "string", "books": [null] }}</pre> <div>Parameter content type application/json-patch+json</div>

Responses

Response content type
application/json

Code	Description
201	<div>Success</div> <div>Example Value Model</div> <pre>{ "id": "string", "title": "string", "description": "string", "categories": [{ "id": "string", "name": "string" }]}</pre>

Open API / SWAGGER

```
1 {  
2   "swagger": "2.0",  
3   "info": {  
4     "version": "v1",  
5     "title": "Awesome Book API"  
6   },  
7   "host": "localhost:16388",  
8   "basePath": "/",  
9   "paths": {  
10    "/api/Books": {  
11      "get": {  
12        "tags": [  
13          "Books"  
14        ],  
15        "operationId": "ApiBooksGet",  
16        "consumes": [],  
17        "produces": [  
18          "text/plain",  
19          "application/json",  
20          "text/json"  
21        ],  
22        "responses": {  
23          "200": {  
24            "description": "Success",  
25            "schema": {  
26              "type": "array",  
27              "items": {  
28                "$ref": "#/definitions/Book"  
29              }  
30            }  
31          }  
32        }  
33      }  
34    }  
35  }
```

Der Swagger-Generator liefert ein maschinenlesbares JSON-File, welches die API beschreibt. Mit diesen Informationen kann für eine API automatisch ein Client generiert werden

Open API / SWAGGER MINIMAL API

```
void MapMinimalApi(WebApplication app)
{
    app.MapGet("api/v1/minimal/noinfo", () =>
    {
        return GenerateForecast();
    });

    app.MapGet("api/v1/minimal/fluent", () =>
    {

        return GenerateForecast();

    }).Produces<WeatherForecast>(contentType: "application/json")
    .WithTags("MinimalAPI");

    app.MapGet("api/v1/minimal/fluent/{id}", () =>
    {
        bool fail = Random.Shared.Next(1, 11) > 5;

        if (fail)
        {
            return Results.BadRequest(new ApiError("112", "OH NOOOOO, fehlender Infinity-Stein..."));
        }
        else
        {
            var forecast = GenerateForecast();
            return Results.Ok(forecast);
        }
    }).Produces<WeatherForecast>(contentType: "application/json")
    .Produces<ApiError>(statusCode: StatusCodes.Status404NotFound, contentType: "application/json")
    .WithTags("MinimalAPI");
}
```

Open API / SWAGGER MINIMAL API

AspNetOpenApi ^	
GET	/api/v1/minimal/noinfo ✓
MinimalAPI ^	
GET	/api/v1/minimal/fluent ✓
GET	/api/v1/minimal/fluent/{id} ✓