

Online on-demand Project Support



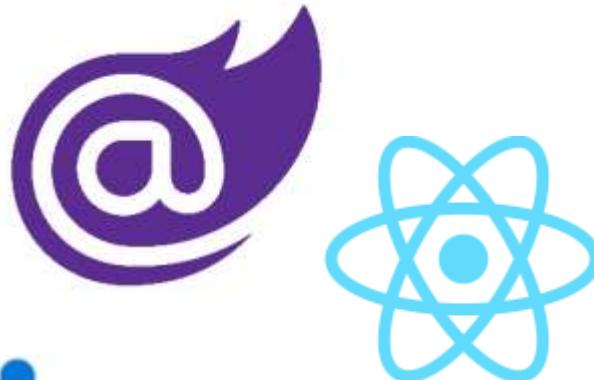
# Aus der Praxis: ASP.NET Core Web API



Thorsten Kansy ([tkansy@dotnetconsulting.eu](mailto: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



# Agenda

- Übersicht
  - Swagger, OpenAPI, Sicherheit,, ...
- Persönlicher Teil
  - Swagger
  - Globaler Exception Handler
  - Self Hosting
  - Application Key
  - SignalR
  - Asynchrones
  - K6
  - Health Checks

Vielleicht ist ja etwas dabei...



# Wie ASP.NET Core?

- Laut Microsoft
  - Cross-platform
  - High-performance,
  - Open source
- Basis von
  - Web.API
  - MVC
  - Blazor
  - ...

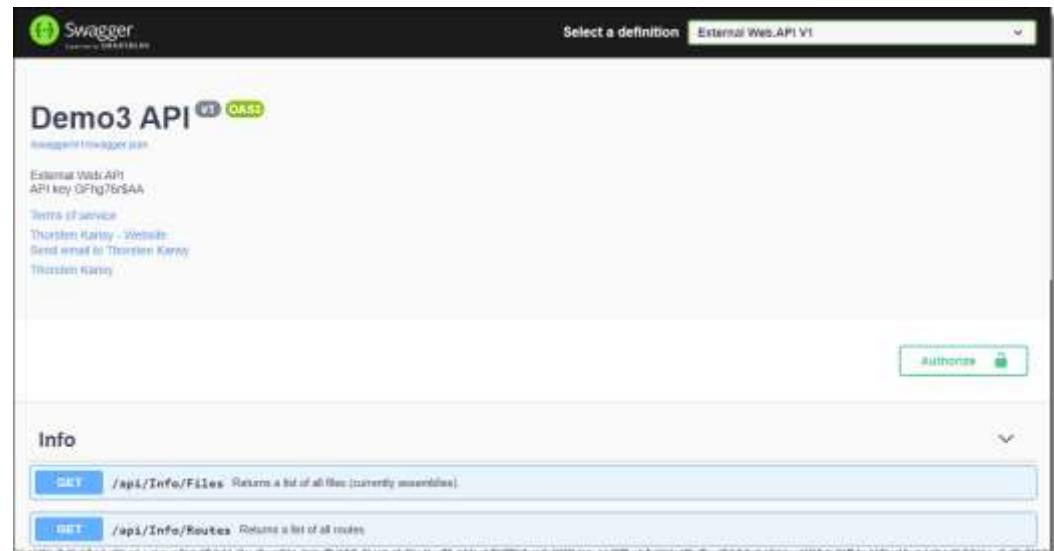


# Swagger



# Swagger

- OpenAPI
- Swashbuckle Middleware
- swagger.json
  - Client generieren lassen
  - swagger-codegen



<https://marketplace.visualstudio.com/items?itemName=ChristianResmaHelle.ApiClientCodeGenerator>



**Demo**



# Globaler Exception Handler (GEH)

- Zentrale Verarbeitung von Ausnahmen
- Vereinheitliche Fehlerinfos
- Optimaler Weise auswertbar am Client

# GEH: ConfigureServices

```
0 references | Thorsten, 251 days ago | 1 author, 2 changes
public void ConfigureServices(IServiceCollection services)
{
    // Set up controllers
    services.AddControllers(o =>
    {
        o.Filters.Add(new RestrictionFilter());
        o.Filters.Add(new ExceptionHandler());
    });
}
```

# GEH: Exception Handler

```
public class ExceptionHandler : ExceptionFilterAttribute
{
    0 references | Thorsten, 251 days ago | 1 author, 2 changes
    public override Task OnExceptionAsync(ExceptionContext context)
    {
        if (context.Exception is CustomException exception)
        {
            context.Result = new JsonResult(ErrorDetails.Create(exception));
            context.HttpContext.Response.StatusCode = 400;
        }
        else if (context.Exception is ArgumentOutOfRangeException)
            context.Result = new NotFoundResult();

        return base.OnExceptionAsync(context);
    }
}
```

# GEH: Error Details

```
public class ErrorDetails
{
    3 references | Thorsten, 252 days ago | 1 author, 1 change
    public int Id { get; set; }

    2 references | Thorsten, 252 days ago | 1 author, 1 change
    public string Message { get; set; }

    2 references | Thorsten, 252 days ago | 1 author, 1 change
    public object Details { get; set; }

    1 reference | Thorsten, 251 days ago | 1 author, 1 change
    public static ErrorDetails Create(ExternalApi.CustomException ex)
    {
        return new ErrorDetails
        {
            Id = ex.EventId.Id,
            Message = $"E: {ex.Message}",
            Details = ex.Details
        };
    }
}
```

# GEH: Custom Exception

```
12 references | Thorsten, 251 days ago | 1 author, 1 change
public class CustomException : Exception
{
    public readonly EventId EventId;
    public readonly object? Details;

    4 references | Thorsten, 251 days ago | 1 author, 1 change
    public CustomException(EventId EventId, object? Details = null)
    {
        this.EventId = EventId;
        this.Details = Details!;
    }
}
```



**Demo**

# Asynchrones

# Asynchrones

```
[HttpPost(nameof(Delete))]
[Produces("application/json")]
1 reference | 0 changes | 0 authors, 0 changes
public async Task<IActionResult> Create(Specification Specification)
{
    _logger.LogInformation($"{nameof(Create)} ({Specification})");

    await _externalLogic.CreateSpecificationAsync(Specification);

    return Ok();
}
```

Aber was ist bei einem Abbruch?

# Asynchrones mit Abbruch

```
public async Task<IActionResult> Create(CancellationToken cancellationToken)
{
    EntryDto trackingEntry = await _appLogic.CreateNewTrackingEntryAsync(cancellationToken);
    cancellationToken.ThrowIfCancellationRequested();

    IList<OrderDto> listOrder = await _appLogic.GetOrdersAsync(cancellationToken);
    cancellationToken.ThrowIfCancellationRequested();
```

```
// Ergebnis abrufen
IList<Order> rawResult = await query.ToListAsync(cancellationToken);  
cancellationToken.ThrowIfCancellationRequested();
```

# Http Status?

```
0 references | Thorsten, 29 days ago | 1 author, 1 change
public class OperationCancelledExceptionFilter : ExceptionFilterAttribute
{
    private readonly ILogger _logger;

    0 references | Thorsten, 29 days ago | 1 author, 1 change
    public OperationCancelledExceptionFilter	ILoggerFactory loggerFactory)
    {
        _logger = loggerFactory.CreateLogger<OperationCancelledExceptionFilter>();
    }

    0 references | Thorsten, 29 days ago | 1 author, 1 change
    public override void OnException(ExceptionContext context)
    {
        if (context.Exception is OperationCanceledException)
        {
            _logger.LogInformation("Request was cancelled");
            context.ExceptionHandled = true;
            context.Result = new StatusCodeResult(499); // 499 CLIENT CLOSED REQUEST (non standard)
        }
    }
}
```





**Demo**



# Self hosting



# Self hosting

- Mehrere (unabhängige) Web.APIs in einem Programm
  - Service/ Demon

# Self hosting: AddHostedService

```
1 reference | 0 changes | 0 authors, 0 changes
public static IHostBuilder CreateHostBuilder(string[] args)
{
    return Host.CreateDefaultBuilder(args)
        .ConfigureServices((hostContext, services) =>
    {
        // External Api
        IConfigurationSection externalApiConfigurationSection = Configuration.GetSection("ExternalApi");
        services.Configure<ExternalApi.WebApiSettings>(externalApiConfigurationSection);
        services.AddHostedService<ExternalApi.WebApiService>();

        // Internal Api
        IConfigurationSection internalApiConfigurationSection = Configuration.GetSection("InternalApi");
        services.Configure<InternalApi.WebApiSettings>(internalApiConfigurationSection);
        services.AddHostedService<InternalApi.WebApiService>();
    });
}
```



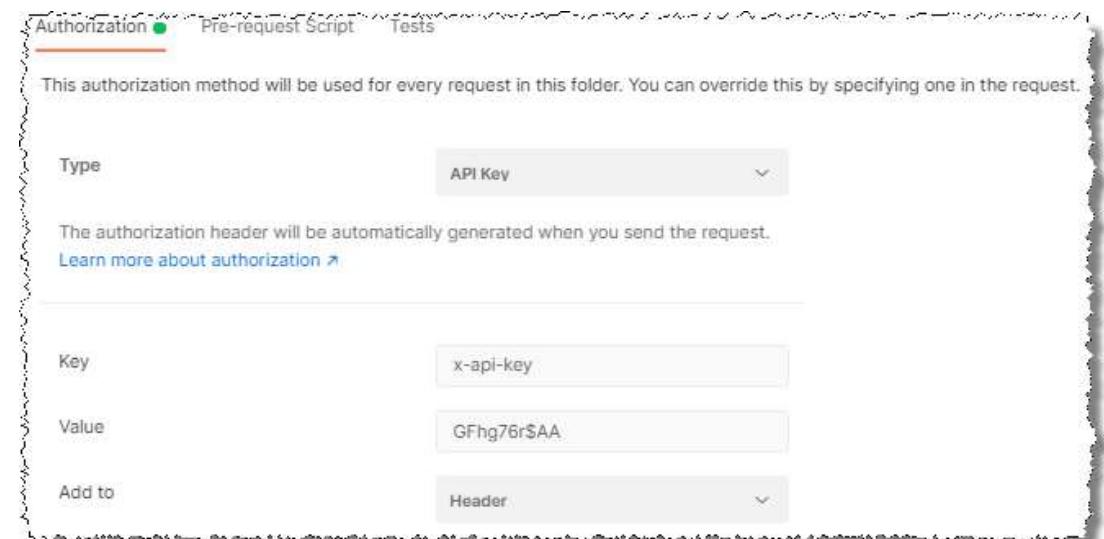
**Demo**



# Application Key

# Application Key

- Sicherstellen/ Kontrollieren welche Anwendung welche API verwendet
- **x-api-key** Header
- Schwache Sicherheitsfeatures



# Application Key: IAuthorizationFilter

```
/// <summary>
/// Filter to check api key.
/// </summary>
5 references | 0 changes | 0 authors, 0 changes
public class ApiKeyFilter : IAuthorizationFilter
{
    /// <summary>
    /// Name of header.
    /// </summary>
    public const string APIKEYNAME = "x-api-key";
```

# Application Key: OnAuthorization

```
0 references | 0 changes | 0 authors, 0 changes
public void OnAuthorization(AuthorizationFilterContext context)
{
    bool skipApiKeyCheck = false;

    // Currently we only skip the check if the Info-Controller is invoked.
    if (context.ActionDescriptor is ControllerActionDescriptor cad)
    {
        // Skip for InfoController
        skipApiKeyCheck = cad.ControllerTypeInfo.AsType() == typeof(Common.InfoController);
    }

    if (skipApiKeyCheck)
        return;

    // Verify API key
    string apiKey = context.HttpContext.Request.Headers[APIKEYNAME].ToString();

    if (string.Compare(_apiKey, apiKey) != 0)
        context.Result = new UnauthorizedResult();
}
```

# Application Key: ConfigureServices

```
0 references | Thorsten, 251 days ago | 1 author, 2 changes
public void ConfigureServices(IServiceCollection services)
{
    // Set up controllers
    services.AddControllers(o =>
    {
        o.Filters.Add(new RestrictionFilter());
        o.Filters.Add(new ExceptionHandler());
    });
}
```

# IExceptionHandler-Middleware

```
public class GlobalExceptionHandler(ILogger<GlobalExceptionHandler> logger) : IExceptionHandler
{
    0 references | 0 changes | 0 authors, 0 changes
    public async ValueTask<bool> TryHandleAsync(
        HttpContext httpContext,
        Exception exception,
        CancellationToken cancellationToken)
    {
        logger.LogError(exception, "Exception occurred: {Message}", exception.Message);

        var problemDetails = new ProblemDetails
        {
            Status = StatusCodes.Status500InternalServerError,
            Title = "Server error"
        };

        httpContext.Response.StatusCode = problemDetails.Status.Value;

        await httpContext.Response
            .WriteAsJsonAsync(problemDetails, cancellationToken);

        return true;
    }
}
```



**Demo**

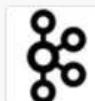


# SignalR

# SignalR

- Echtzeitverbindung Web.API => Client
- Nachrichten gehen vom Server (HUB) aus

TOP ALTERNATIVES TO SIGNALR

 <b>Firebase</b> Firebase is a cloud service designed to power real-time, collaborative applications. ...	 <b>Pusher</b> Pusher is the category leader in delightful APIs for app developers building ...
 <b>RabbitMQ</b> RabbitMQ gives your applications a common platform to send and receive messages, ...	 <b>WebRTC</b> It is a free, open project that enables web browsers with Real-Time Communications ...
 <b>MQTT</b> It was designed as an extremely lightweight publish/subscribe messaging transport. ...	 <b>gRPC</b> gRPC is a modern open source high performance RPC framework that can run in ...
 <b>WCF</b> It is a framework for building service-oriented applications. Using this, you ...	 <b>Kafka</b> Kafka is a distributed, partitioned, replicated commit log service. It provides ...

# SignalR: Server (Hub)

```
[Authorize(Policy = "ApiKeyPolicy")]
```

7 references | 0 changes | 0 authors, 0 changes

```
public class SignalRHub : Hub<ISignalRMethods>
{
```

```
    // Endpoint to register to
```

```
    public const string ENDPOINT = "/ExternalHub";
```

```
    private readonly ILogger<SignalRHub> _logger;
```

0 references | 0 changes | 0 authors, 0 changes

```
    public SignalRHub(ILogger<SignalRHub> logger)
```

```
    {
```

```
        _logger = logger;
```

```
        _logger.LogInformation("SignalR hub started.");
```

```
}
```

```
}
```

5 references | 0 changes | 0 authors, 0 changes

```
public interface ISignalRMethods
```

```
{
```

1 reference | 0 changes | 0 authors, 0 changes

```
    Task NewScan();
```

```
}
```

```
public static IHubContext<SignalRHub, ISignalRMethods> externalSignalRHub = null!;
```

```
await Common.GlobalObjects.externalSignalRHub.Clients.All.NewScan();
```

# SignalR: Client

```
#region Setup SignalR Connection
connection = new HubConnectionBuilder()
    .WithUrl(SIGNALRHUB, options =>
{
    options.Headers.Add("x-api-key", apiKey);
})
    .WithAutomaticReconnect(new TimeSpan[] { TimeSpan.FromSeconds(10) })
    .Build();

connection.Closed += async (error) =>
{
    await Task.Delay(new Random().Next(0, 5) * 1000);
    await connection.StartAsync();
};

connection.Reconnecting += error =>
{
    Console.WriteLine($"Error: connection.State == {connection.State}");
    return Task.CompletedTask;
};

connection.StartAsync().ContinueWith(t => ...).Wait();
#endregion
```



**Demo**



# HashId

## Vorhersagbare IDs vermeiden

Հյուրներ

Բառը այս հաշիվը

բառական աշխատանքը պահպանությունը  
Հյուրները այս հաշիվը պահպանությունը

սեղանը օլ շնորհես Տիգլեսդեղանլի և ի ամեն

```
Install-Package AspNetCore.Hashids
```



**Demo**



# K6 Load test

# K6 Load test

```
cmd C:\Windows\System32\cmd.exe
scenarios: (100.00%) 1 scenario, 5 max VUs, 1m0s max duration (incl. graceful stop):
  * default: 5 looping VUs for 30s (gracefulStop: 30s)

running (0m30.0s), 0/5 VUs, 18825 complete and 0 interrupted iterations
default [=====] 5 VUs 30s

  ↳ allUsers

  checks.....: 100.00% ↳ 18825      ↳ 0
  data_received.....: 386 MB 13 MB/s
  data_sent.....: 2.5 MB 82 kB/s
  http_req_blocked.....: avg=3.62µs min=0s med=0s max=9.08ms p(90)=0s p(95)=0s
  http_req_connecting.....: avg=1.87µs min=0s med=0s max=9.08ms p(90)=0s p(95)=0s
  http_req_duration.....: avg=7.81ms min=2.01ms med=7.46ms max=212.74ms p(90)=9.32ms p(95)=10.11ms
    { expected_response:true }....: avg=7.81ms min=2.01ms med=7.46ms max=212.74ms p(90)=9.32ms p(95)=10.11ms
  http_req_failed.....: 0.00% ↳ 0      ↳ 18825
  http_req_receiving.....: avg=364.63µs min=0s med=0s max=9.02ms p(90)=972µs p(95)=1.6ms
  http_req_sending.....: avg=9.52µs min=0s med=0s max=12.84ms p(90)=0s p(95)=0s
  http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
  http_req_waiting.....: avg=7.44ms min=1.05ms med=7.13ms max=212.18ms p(90)=9.01ms p(95)=9.82ms
  http_reqs.....: 18825 627.395714/s
  iteration_duration.....: avg=7.96ms min=2.02ms med=7.58ms max=212.74ms p(90)=9.44ms p(95)=10.21ms
  iterations.....: 18825 627.395714/s
  vus.....: 5 min=5 max=5
  vus_max.....: 5 min=5 max=5

C:\Workshops\asp.net-core\6.0\dotnetconsulting.UserManagement\dotnetconsulting.UserManagement.LoadTest>
```



<https://k6.io>



**Demo**



# Health Checks

# Health Checks

Prüfen, ob die genutzten Ressourcen verfügbar sind

čuîlđêş Şêswîçêş AđđHêáłtjhChêçlş

Ačêş xiê şôl'l xêlçhêş Têştj đusçhûgûhstj xêsdêñ

Ínrlêñêñtjiêşunû wôy ÍHêáłtjhChêçl

árr̃ ŅářHêáłtjhChêçlş hêáłtjh

```
Install-Package AspNetCore.HealthChecks.*
```



<https://github.com/Xabaril/AspNetCore.Diagnostics.HealthChecks>

# UI

The screenshot shows the 'Health Checks Status' page of the AspNetCore.HealthChecks.UI application. On the left, there's a sidebar with icons for Home, Health Checks (selected), and Webhooks. The main area has a title 'Health Checks Status' with a 'Polling interval: 10 secs' button and a 'Stop polling' button. Below is a table with three sections: 'Health Checks', 'Webhooks', and another 'Health Checks' section.

NAME	HEALTH	ON STATE FROM	LAST EXECUTION		
endpoint1	🔴 Unhealthy	2020-09-16T00:11:09.2075445+02:00	16/9/2020 0:11:19		
NAME	TAGS	HEALTH	DESCRIPTION	DURATION	DETAILS
random1	random	🔴 Unhealthy	The healthcheck random1 failed at minute 11	00:00:00.0002128	⌚
random2	random	🔴 Unhealthy	The healthcheck random2 failed at minute 11	00:00:00.0001129	⌚
NAME	TAGS	HEALTH	DESCRIPTION	DURATION	DETAILS
process_allocated_memory	process memory	🟢 Healthy	Allocated megabytes in memory: 15 mb	00:00:00.0001688	⌚

```
Install-Package AspNetCore.HealthChecks.UI.*
```

# Fragen? Jetzt oder später!



## Kontakt

 E-Mail  
[tkansy@dotnetconsulting.eu](mailto:tkansy@dotnetconsulting.eu)

 LinkedIn  
[Link me](#)

 Telefon  
[+49 \(0\) 6187 / 2009090](tel:+49(0)6187/2009090)

 XING  
[Xing me](#)

 Microsoft Teams  
[Meet now](#)

 X (Twitter)  
[@tkansy](https://twitter.com/tkansy)



# [www.dotnetconsulting.eu](http://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.