

Online on-demand Project Support



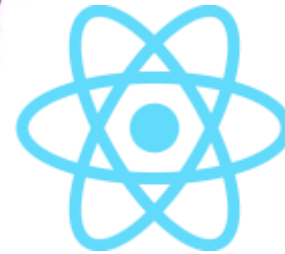
# Aus der Praxis: ASP.NET Core Web API



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



# Folien und Skripte- in meinem Blog

## SQL Server meets .NET (Core)- professionally!

Meine Name ist Thorsten Kansy und ich berate, coache und trainiere im Bereich Entwicklung von .NET (Core) Anwendungen in Verbindung mit Microsoft SQL Server- mit Allem, was dazu gehört.

 [Kontakt](#)



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

# Agenda

- Swagger
- Sicherheit
- 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
  - ...

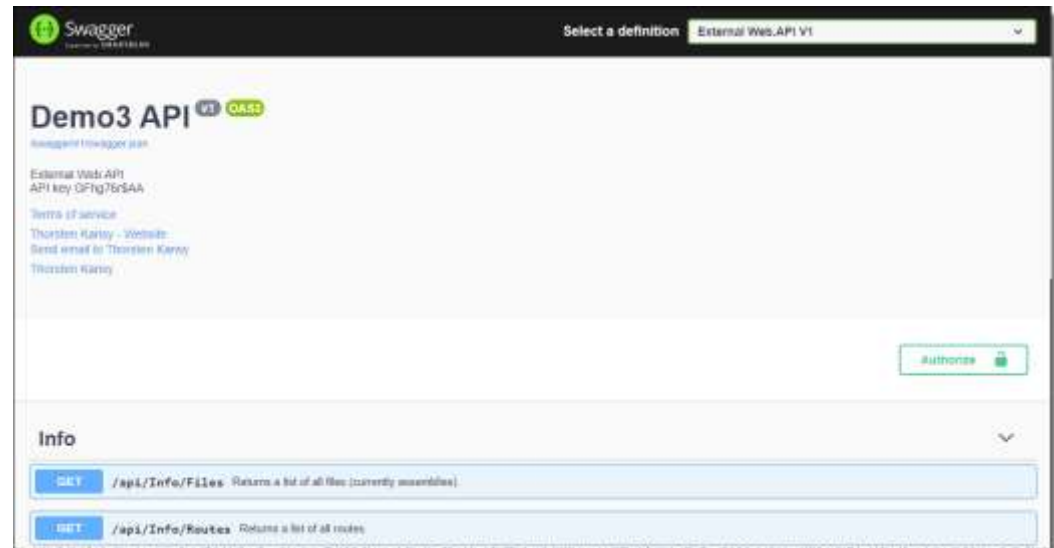


# Swagger



# Swagger

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



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



Demo



# Globaler Exception Handler



ALL VISITORS  
NO VISITOR VEHICLES  
ALLOWED IN THIS PARK!

SLOW  
Down  
it  
Mite Be  
your  
child

# 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



# Asynchrone

```
[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 OperationCancelledException)  
        {  
            _logger.LogInformation("Request was cancelled");  
            context.ExceptionHandled = true;  
            context.Result = new StatusCodeResult(499); // 499 CLIENT CLOSED REQUEST (non standard)  
        }  
    }  
}
```



Demo



A photograph of a modern high-speed train, possibly a Shinkansen, stopped at a station platform. The train is white with a prominent orange stripe along the bottom. The platform is well-lit with overhead lights and has a yellow tactile paving strip along the edge. In the background, other trains and station infrastructure are visible. A semi-transparent blue banner is overlaid across the middle of the image, containing the text "Self hosting".

# 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



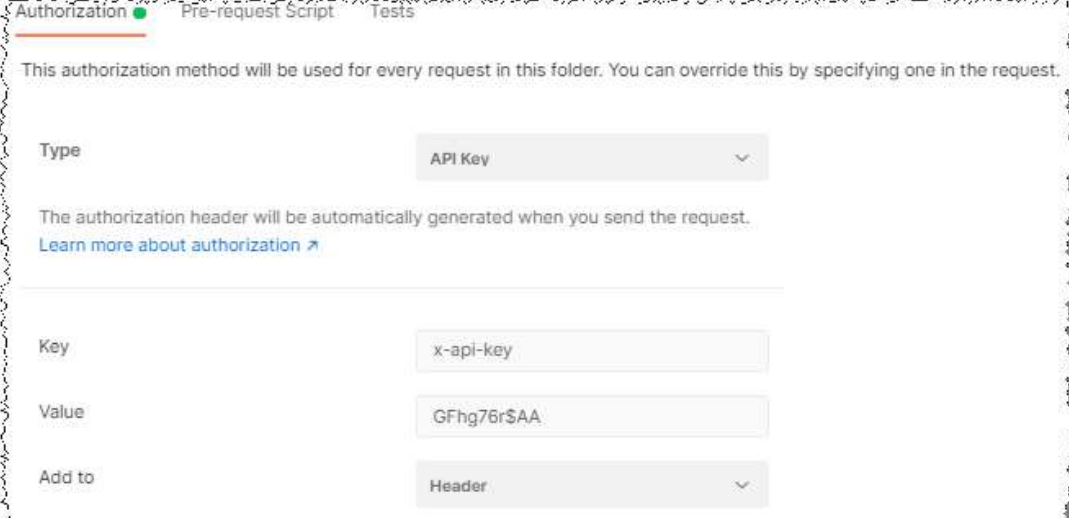


# Application Key



# Application Key

- Sicherstellen/ Kontrollieren welche Anwendung welche API verwendet
- `x-api-key` Header
- Schwaches Sicherheitsfeatures



The screenshot shows the 'Authorization' tab in a testing tool. It contains the following configuration:

- Authorization:** Selected tab, indicated by a green dot.
- Pre-request Script:** Tab.
- Tests:** Tab.
- Description:** This authorization method will be used for every request in this folder. You can override this by specifying one in the request.
- Type:** API Key (dropdown menu).
- Header Info:** The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)
- Key:** x-api-key (text input field).
- Value:** GFhg76r\$AA (text input field).
- Add to:** Header (dropdown menu).

# 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());
    });
}
```



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

# HashedId

## Vorhersagbare IDs vermeiden

```
[HttpGet]
[Route("{id:hashids}")]
public ActionResult<CustomerDto>
    Get([ModelBinder(typeof(HashidsModelBinder))] int id)
{
    return Ok(customers.SingleOrDefault(c => c.Id == id));
}
```

Install-Package AspNetCore.Hashids

 Demo 





K6 Load test

# K6 Load test

```
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 Check

# Health Check

Prüfen, ob die genutzten Ressourcen verfügbar sind

```
builder.Services.AddHealthChecks();  
// Aber wie sollen welcher Test durchgeführt werden?  
// Implementierung von IHealthCheck  
  
app.MapHealthChecks("/_health");
```

# Health Check

Prüfen, ob die genutzten Ressourcen verfügbar sind

```
builder.Services.AddHealthChecks();  
// Aber wie sollen welcher Test durchgeführt werden?  
// Implementierung von IHealthCheck  
  
app.MapHealthChecks("/_health");
```

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



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







# UI

Health Checks Status Polling interval: 10 secs [Stop polling](#)

+	NAME	HEALTH	ON STATE FROM	LAST EXECUTION	
-	endpoint1	<span style="color: red;">!</span>	2020-09-16T00:11:09.2075445+02:00	16/9/2020 0:11:19	
NAME	TAGS	HEALTH	DESCRIPTION	DURATION	DETAILS
random1	<span>random</span>	<span style="color: red;">!</span> Unhealthy	The healthcheck random1 failed at minute 11	00:00:00.0002128	<a href="#">↻</a>
random2	<span>random</span>	<span style="color: red;">!</span> Unhealthy	The healthcheck random2 failed at minute 11	00:00:00.0001129	<a href="#">↻</a>
-	endpoint2	<span style="color: green;">✓</span>	2020-09-16T00:10:16.9574111+02:00	16/9/2020 0:11:19	
NAME	TAGS	HEALTH	DESCRIPTION	DURATION	DETAILS
process_allocated_memory	<span>process</span> <span>memory</span>	<span style="color: green;">✓</span> Healthy	Allocated megabytes in memory: 15 mb	00:00:00.0001688	<a href="#">↻</a>

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

# Fragen? Jetzt oder später!

-  **Email**  
tkansy@dotnetconsulting.eu
-  **Telefon**  
+49 (0) 6187 / 2009090
-  **Microsoft Teams**  
tkansy@dotnetconsulting.eu
-  **LinkedIn**  
Zu meinem Profil
-  **XING**  
Zu meinem Profil
-  **Twitter**  
Zu meinem Profil





# www.dotnetconsulting.eu

[Start](#)[Schwerpunkte](#)[Über mich](#)[Leistungen](#)[Video](#)[Kontakt](#)[Blog](#)

## SQL Server meets .NET (Core)- professionally!

Meine Name ist Thorsten Kansy und ich berate, coache und trainiere im Bereich Entwicklung von .NET (Core) Anwendungen in Verbindung mit Microsoft SQL Server- mit Allem, was dazu gehört.

[Kontakt](#)