ASP.NET 8 IN PRODUCTION

“Force Developers into success!”

Robin-Manuel Thiel

Microsoft

.NET Developer

Sebastian Küsters

Space Blocks

.NET Developer

Space Blocks

Pricing Documentation Blog Developer Portal

Book a demo

The platform for permission management.

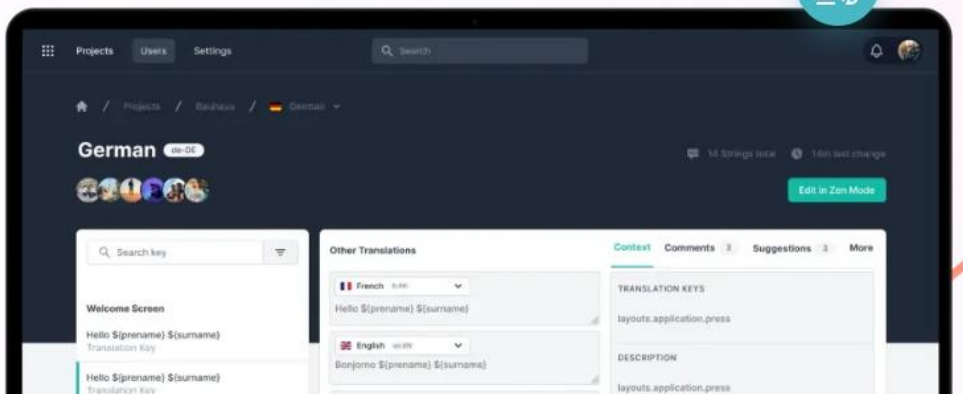
Create roles and access rights for your applications and perform permission checks against our service with ease.

- Free Developer tier
- No credit card needed
- No Setup

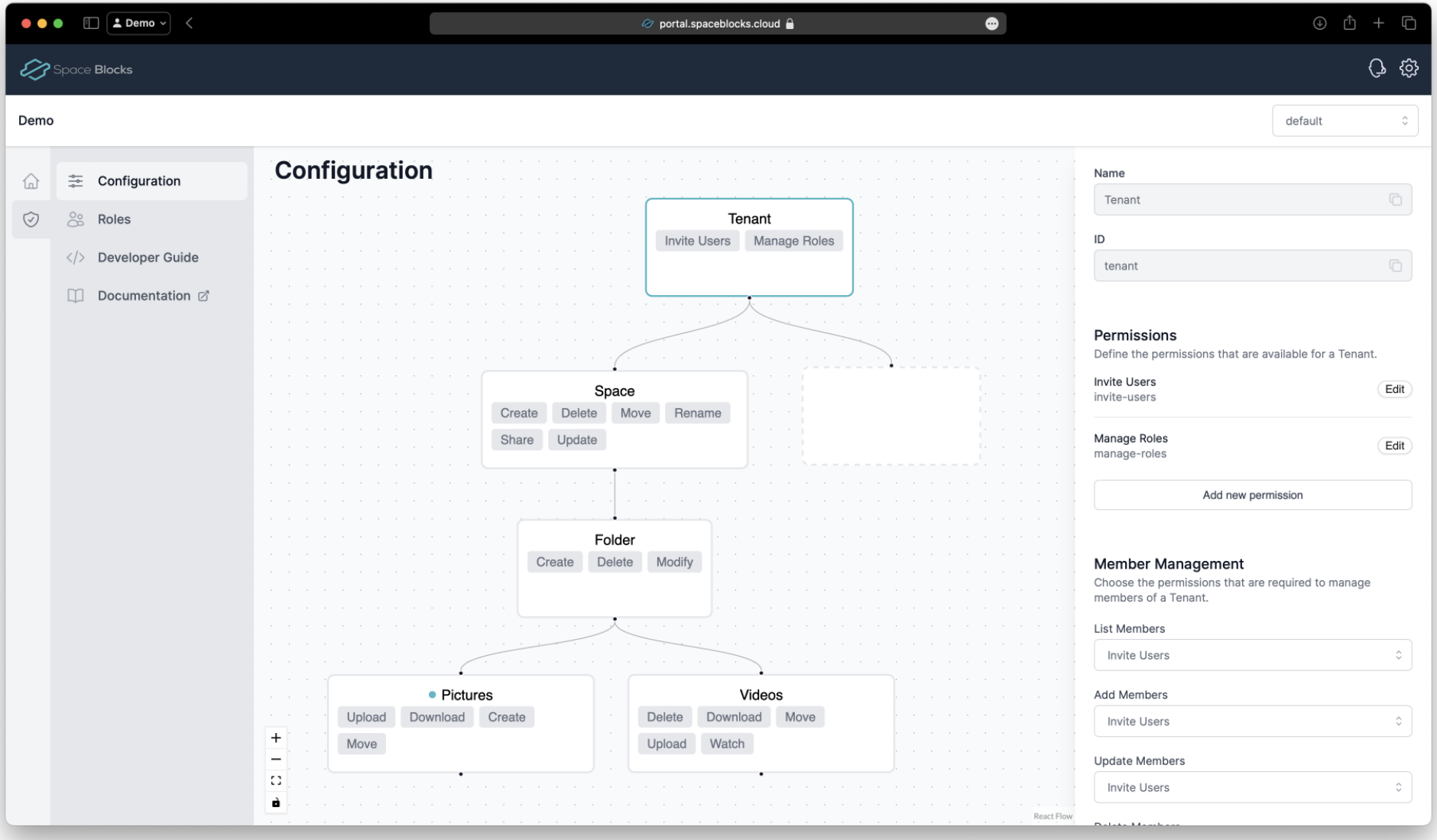
Get started

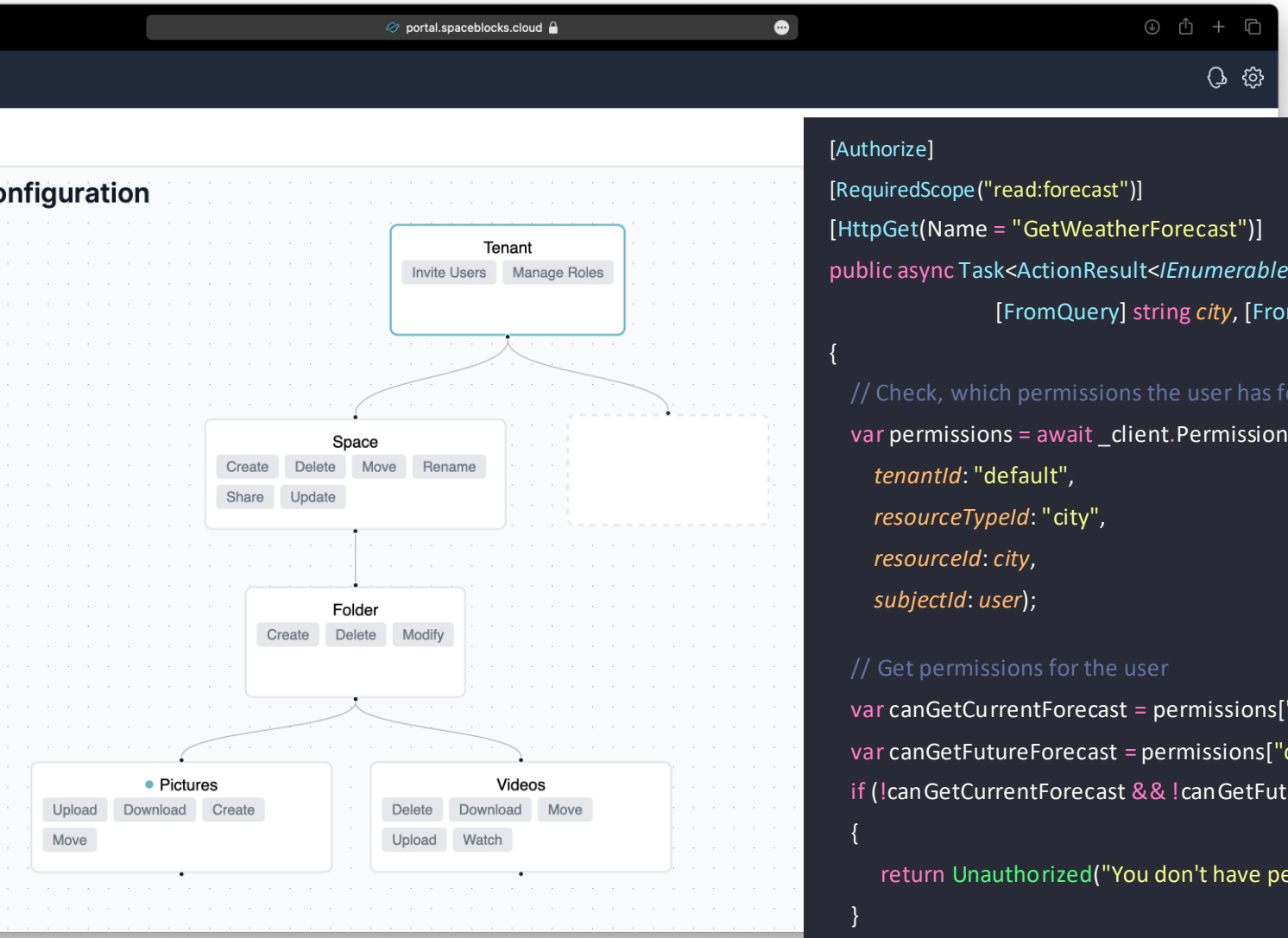
Identity

Subscriptions



Realtime


Demo
portal.spaceblocks.cloud
Space Blocks
Demo
default
Configuration
Roles
Developer Guide
Documentation
Configuration
Tenant
Invite Users
Manage Roles
Space
Create
Delete
Move
Rename
Share
Update
Folder
Create
Delete
Modify
Pictures
Upload
Download
Create
Move
Videos
Delete
Download
Move
Upload
Watch
React Flow
Name
Tenant
ID
tenant
Permissions
Define the permissions that are available for a Tenant.
Invite Users
invite-users
Edit
Manage Roles
manage-roles
Edit
Add new permission
Member Management
Choose the permissions that are required to manage members of a Tenant.
List Members
Invite Users
Add Members
Invite Users
Update Members
Invite Users




```
[Authorize]
[RequiredScope("read:forecast")]
[HttpGet(Name = "GetWeatherForecast")]
public async Task<ActionResult<IEnumerable<WeatherForecast>?>> Get(
                    [FromQuery] string city, [FromQuery] string user)
{
  // Check, which permissions the user has for the city
  var permissions = await _client.PermissionApi.ListPermissionsAsync(
    tenantId: "default",
    resourceTypeId: "city",
    resourceId: city,
    subjectId: user);

  // Get permissions for the user
  var canGetCurrentForecast = permissions["city"].Contains("get-current-forecast");
  var canGetFutureForecast = permissions["city"].Contains("get-future-forecast");
  if (!canGetCurrentForecast && !canGetFutureForecast)
  {
     return Unauthorized("You don't have permissions to access this resource.");
  }

  // ...
}
```

AGENDA

What this session will cover

1. DTOs vs. Models
2. Swagger done right
3. Error Handling
4. Logging
5. Metrics and Tracing with Open Telemetry
6. CQRS
7. Health Checks
8. Containers & Kubernetes

START WITH AN EMPTY PROJECT

```
dotnet new webapi --use-controllers
```

.NET-8-in-Production

EXPLORER

OPEN EDITORS
- Program.cs src/Demo.Api 1

.NET-8-IN-PRODUCTION
- .github
- .vscode
- src
  - Demo.Api
    - bin
    - Controllers
    - obj
    - Properties
    - appsettings.Development.json
    - appsettings.json
    - Demo.Api.csproj
    - Demo.Api.http
    - Program.cs 1
    - WeatherForecast.cs
  - .dockerignore
  - .editorconfig
  - .gitattributes
  - .gitignore
  - .gitlint
  - .pre-commit-config.yaml
  - .prettierignore
  - .prettierrc
  - Directory.Build.props
  - LICENSE
  - README.md
  - Solution.sln

TODOS: TREE
LIVE SHARE: SESSION DETAILS
OUTLINE
TIMELINE
SOLUTION EXPLORER

Program.cs 1

src > Demo.Api > Program.cs

You, 2 minutes ago | 1 author (You)

```
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

You, 2 minutes ago • Initial Repository Structure

main | 0 1 | 0 | Projects: 1 | sbcsharedaks | sbc-dev | Watch | You, 2 minutes ago | Ln 1, Col 50 | Spaces: 4 | UTF-8 | CRLF | C# | Prettier

DTOs & MODELs

Code duplication to prevent data leaks

DTOS & MODELS

APIs only return and accept DTOs – never Models

```
public class Todo
{
    public string Id { get; set; }

    public string Name { get; set; }

    public bool IsCompleted { get; set; }

    public string CreatedBy { get; set; }

    public DateTime CreatedAt { get; set; }
}
```

```
public class TodoDto
{
    public string Id { get; set; }

    public string Name { get; set; }

    public bool IsCompleted { get; set; }
}
```

```
public class CreateTodoRequest
{
    public string Name { get; set; }

    public bool IsCompleted { get; set; }
}
```

DTOS & MODELS

Use a Mapping library like Mapster or AutoMapper

```
[HttpGet]
public IEnumerable<WeatherForecastDto> GetWeatherForecast([FromQuery] ForecastRequest request)
{
    var forecast = Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = Summaries[Random.Shared.Next(Summaries.Length)]
    }).ToArray();

    var dto = forecast.Adapt<WeatherForecastDto>();
    return dto;
}
```

SWAGGER

SWAGGER

Prepare your .csproj file

A Swagger file can be automatically exported from a .dll with the Swashbuckle Swagger CLI tool. We can run this tool at every build.

Add a Tool Manifest to your ASP.NET project

```
dotnet new tool-manifest
```

Install the Swashbuckle Swagger CLI tool

```
dotnet tool install Swashbuckle.AspNetCore.Cli
```

> Use this for SDK generation

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.4.0" />
  </ItemGroup>

  <!-- Generate XML documentation file -->
  <PropertyGroup>
    <GenerateDocumentationFile>true</GenerateDocumentationFile>
    <NoWarn>$(NoWarn);1591</NoWarn>
  </PropertyGroup>

  <!-- Generate swagger.json file after each build -->
  <Target Name="PostBuild" AfterTargets="PostBuildEvent">
    <Exec Command="dotnet tool restore" />
    <Exec Command="dotnet swagger tofile --output $(OutputPath)swagger.json
              $(OutputPath)\$(AssemblyName).dll v1" />
  </Target>
</Project>
```

SWAGGER

Configure Swagger for Production in Program.cs

```
// Swagger
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
  c.SwaggerDoc("v1", new() { Title = "Demo API", Version = "v1" });

  // Include XML comments to Swagger documentation
  c.IncludeXmlComments(Path.Combine(AppContext.BaseDirectory, $"{Assembly.GetCallingAssembly().GetName().Name}.xml"));

  // Remove trailing "Dto" from schema names
  c.CustomSchemaIds(x => x.Name[..x.Name.LastIndexOf("Dto")]);

  // Add Operation ID based on controller method name and remove trailing "Async"
  c.CustomOperationIds(e => "{e.ActionDescriptor.RouteValues["action"]}"[..$"{e.ActionDescriptor.RouteValues["action"]}".LastIndexOf("Async")]);
});

// Skip rest of setup, if app is executed in context of dotnet-swagger
if (Assembly.GetEntryAssembly()?.FullName?.Contains("dotnet-swagger") == true)
{
   return;
}

// ...
var app = builder.Build();
app.UseSwagger();
app.UseSwaggerUI();
```

SWAGGER

Document API Endpoints in Controllers

```
/// <summary>
/// Query the weather forecast
/// </summary>
/// <param name="city">Name of the city for the forecast</param>
/// <returns>A list of <see cref="WeatherForecast"/></returns>
[HttpGet]
[ProducesResponseType(StatusCodes.Status200OK)]
[Produces(MediaTypeNames.Application.Json)]
public IEnumerable<WeatherForecast> GetWeatherForecast([FromQuery] string city)
{
    return Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = Summaries[Random.Shared.Next(Summaries.Length)]
    }).ToArray();
}
```

ERROR HANDLING

ERROR HANDLING

Errors vs. Exceptions

ERRORS

Expected cases, that we planned for. Have an Error Code that can be documented and returned to the user.

VS

EXCEPTIONS

Classical exceptions, that occur, when something unexpected happens.

```
public static class ErrorCodes
{
    public const string OAuthClientNotFound = "oauth-100";
    public const string OAuthClientCreationFailed = "oauth-200";
    public const string OAuthClientDeletionFailed = "oauth-210";
    public const string ProjectNotFound = "project-100";
    public const string EnvironmentNotFound = "environment-100";
    public const string SlugDuplicated = "validate-100";
    public const string SpaceBlockNotInProject = "validate-200";
    public const string SpaceBlockAlreadyInProject = "validate-300";
    public const string SpaceBlockNotFoundInEnvironmentDeployments = "validate-400";
    public const string EntityNotFound = "notfound-100";
}
```

ERROR HANDLING

Errors vs. Exceptions

Base exception class

```
public class ErrorExceptionBase(string code, string message = "") : Exception
{
   public string Code { get; set; } = code;

   public string Message { get; set; } = message;
}
```

Type per exception

```
public class ValidationErrorException(string code, string message = "") : ErrorExceptionBase(code, message)
{
}
```

Declare all errors at central place

```
public static class Error
{
   public static AuthorizationErrorException Authorization(string code, string message = "") => new(code, message);

   public static NotFoundErrorException NotFound(string code, string message = "") => new(code, message);

   public static UnexpectedErrorException Unexpected(string code, string message = "") => new(code, message);

   public static ValidationErrorException Validation(string code, string message = "") => new(code, message);
}
```

ERROR HANDLING

Implement & Register the ErrorHandlerMiddleware

Create the Middleware

```
// Primary constructor (Introduced in C# 12)
public class ErrorHandlerMiddleware(RequestDelegate next)
{
    public async Task InvokeAsync(HttpContext context)
    {
        try
        {
            await next(context);
        }
        catch (ValidationErrorException exception)
        {
            var response = context.Response;
            response.StatusCode = (int)HttpStatusCode.BadRequest;
            await response.WriteAsJsonAsync(exception.Message);
        }
        catch (AuthorizationErrorException exception)
        {
            var response = context.Response;
            response.StatusCode = (int)HttpStatusCode.Forbidden;
            await response.WriteAsJsonAsync(exception.Message);
        }
    }
}
```

ErrorHandlerMiddleware.cs

Register the Middleware in the Application Builder in Program.cs

```
app.UseMiddleware<ErrorHandlerMiddleware>();
```

ERROR HANDLING

Exception Handling in Controller Action

Before

```
public ActionResult<IEnumerable<WeatherForecast>> Get([FromQuery] string city)
{
    try
    {

        // business logic, that throws exception
        if (city == "London")
        {
            throw new ValidationException("City not found");
        }

        // generate weather forecast
        return new OkObjectResult(Enumerable.Range(1, 5).Select(
            index => new WeatherForecast
        {
            Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = Summaries[Random.Shared.Next(Summaries.Length)]
        }).ToArray());
    }
    catch (ValidationException exception)
    {
        return new BadRequestObjectResult(exception.Message);
    }
}
```

WeatherForecastController.cs

After

```
public IEnumerable<WeatherForecast> Get([FromQuery] string city)
{
    // business logic, that throws exception
    if (city == "London")
    {
        throw Error.NotFound(
            ErrorCodes.CityNotFound,
            $"The city {city} was not found");
    }

    // generate weather forecast
    return Enumerable.Range(1, 5).Select(
        index => new WeatherForecast
    {
        Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = Summaries[Random.Shared.Next(Summaries.Length)]
    }).ToArray();
}
```

WeatherForecastController.cs

LOGGING

LOGGING

Unified Logging with ILogger

Configure Logging centrally

```
public static class Observability
{
    public static void ConfigureStandardLogger(this ILoggingBuilder builder)
    {
        builder.ClearProviders();
        builder.AddConsole();
    }
}
```

Register Logging Centrally in Program.cs

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddLogging(c => c.ConfigureStandardLogger());
```

LOGGING

Unified Logging with ILogger

Get Logger from Dependency Injection

```
private readonly ILogger<WeatherForecastController> _logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)
{
 _logger = logger;
}

public IEnumerable<WeatherForecast> Get([FromQuery] string city)
{
 _logger.LogInformation("Getting weather forecast for {city}", city);

 // ...
}
```

Get Logger from elsewhere (e.g. in Program.cs)

```
var logger = LoggerFactory.Create(logger => logger.ConfigureStandardLogger()).CreateLogger<Program>();
logger.LogInformation("Demo Information");
logger.LogWarning("Demo Warning");
logger.LogError("Demo Error");
```

Console output

```
info: Program[0]
      Demo Information
warn: Program[0]
      Demo Warning
fail: Program[0]
      Demo Error
```

LOGGING

Log correctly

Use Logger Variables correctly to see and filter them later (e.g. in Application Insights)

```
_logger.LogWarning("Payment with id {PaymentId} failed with status {PaymentStatus}", paymentId, PaymentStatus.Canceled);
```

timestamp [UTC]	2023-04-03T16:21:40.0575856Z
message	Payment with id 7ea66ecc-cb24-4537-a73a-5ef36b7a2296 failed with status canceled
EventId	-1
EventName	LogPaymentError
OriginalFormat	Payment with id {PaymentId} failed with status {PaymentStatus}
PaymentId	7ea66ecc-cb24-4537-a73a-5ef36b7a2296
PaymentStatus	canceled

LOGGING

Choose the right Log Level

> Whenever you choose Information as the right log level for whatever you want to log, ask yourself why! Because when are you looking at your logs? Whenever things don't go well!
>
> - Nick Chapsas

> You should consider using *Warning* as the default Log Level.
>
> Logs should tell you a story about how and why something failed.
>
> Keep in mind, that the amount of Logs cause costs.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information",
      "Microsoft.EntityFrameworkCore": "Warning"
    },
    "ApplicationInsights": {
      "LogLevel": {
      "Default": "Warning",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "None"
    }
  }
}
```

METRICS & TRACING

METRICS & TRACING

METRICS & TRACING

```
public static class Observability
{
    public static readonly string ServiceName = typeof(Observability).Assembly.GetName().Name!;

    public static readonly string ServiceVersion = typeof(Observability).Assembly.GetName().Version != null
        ? typeof(Observability).Assembly.GetName().Version!.ToString()
        : "0.0.0";

    // Define a default ActivitySource
    public static readonly ActivitySource Default = new ActivitySource(ServiceName);

    // Define a default Meter with name and version
    public static readonly Meter Meter = new(ServiceName, ServiceVersion);

    // Create Counters, Histograms, etc. from that default Meter
    public static readonly Counter<long> Pings = Meter.CreateCounter<long>("service_countername", description: "Total number of pings");
    public static readonly Histogram<int> PingDelay = Meter.CreateHistogram<int>("service_histgramname", "ms", "Think time in ms");

    public static void ConfigureStandardLogger(this ILoggingBuilder builder)
    {
        builder.ClearProviders();
        builder.AddConsole();
    }
}
```

METRICS & TRACING

```
// Metrics
builder.Services.AddOpenTelemetry().WithMetrics(builder =>
{
   builder.AddMeter("MyMeter");
   builder.AddRuntimeInstrumentation();
   buildxer.AddHttpClientInstrumentation();
   builder.AddAspNetCoreInstrumentation();
   builder.AddPrometheusExporter();
});

// Traces
builder.Services.AddOpenTelemetry().WithTracing(builder =>
{
   builder.AddSource(Observability.ServiceName);
   builder.ConfigureResource((resource) =>
   {
      resource.AddService(Observability.ServiceName, Observability.ServiceVersion);
   });

   builder.AddAspNetCoreInstrumentation();
   builder.AddEntityFrameworkCoreInstrumentation();
   builder.AddOtlpExporter(oltpOptions =>
});
```

Program.cs

METRICS & TRACING

Increase Counter

```
Observability.Pings.Add(1);
```

Register a span

```
public async Task<List<Todo>> HandleAsync(GetTodosQuery query, CancellationToken cancellationToken)
{
    using var activity = Observability.Default.StartActivity("Getting tasks from database");
    await Task.Delay(500);
    return _todoRepository.ToList();
}
```

METRICS & TRACING

View in Application Insights

Microsoft Azure

Search resources, services, and docs (G+/)

robin-manuel@wemog...
WEMOGY (WEMOGY.COM)

Home > Application Insights > sbcprodinsights | Performance >

End-to-end transaction details

sbcprodinsights

Search results | Learn more | Copy link | Feedback | Leave preview

End-to-end transaction
Operation ID: 583dcaa223d2d56f36c6a1316af3e6ae

= Request (incoming) = Internal = Dependency (outgoing)

Search

EVENT	RES.	DURATION
gelzsx.permissions.eu1.spaceblocks.cloud GET management/permis	200	12.1 ms
CheckPermissionsQuery Query		11.2 ms
CheckPermissionsQuery QueryHandler		11.2 ms
GetPermissionsDictionaryQuery Query		11.1 ms
GetPermissionsDictionaryQuery QueryHandler		11.1 ms
GetExplicitResourcePermissionQuery Query		9.3 ms
GetExplicitResourcePermissionQuery QueryHandler		9.3 ms
POSTGRESQL citus		1.9 ms
GetSubjectResourcePermissionsPermissionsDictio		6.8 ms
GetSubjectResourcePermissionsPermissionsDic		6.8 ms
GetResourceTypeWithDescendantsQuery Qu		6.8 ms
GetResourceTypeWithDescendantsQuery		6.8 ms
GetResourceTypesQuery Query		6.7 ms
GetResourceTypesQuery QueryHandler		6.7 ms
POSTGRESQL citus		2.4 ms

0 10 MS

Traces & events 0 Traces 0 Events

View all

Search results

CQRS

!= Event Sourcing ☺

CQRS

How a Controller using CQRS looks like

```
public class TodoController
{
    private readonly ICommands _commands;
    private readonly IQueries _queries;

    public TodoController(ICommands commands, IQueries queries)
    {
        _commands = commands;
        _queries = queries;
    }

    [HttpGet]
    public async Task<ActionResult<List<TodoDto>>> ListTodos()
    {
        var query = new GetTodosQuery();
        var result = await _queries.QueryAsync(query);
        var dto = result.Adapt<List<TodoDto>>();
        return new OkObjectResult(dto);
    }

    [HttpPost]
    public async Task<ActionResult<TodoDto>> CreateTodo([FromBody] CreateTodoRequest request)
    {
        var command = request.Adapt<CreateTodoCommand>();
        var result = await _commands.RunAsync(command);
        var dto = result.Adapt<TodoDto>();
        return new OkObjectResult(dto)
        {
          StatusCode = 201
        };
    }
}
```

CQRS

Command Example

```
public class CreateTodoCommand : ICommand<Todo>
{
    public string Name { get; }

    public CreateTodoCommand(string name)
    {
        Name = name;
    }
}
```

```
public class CreateTodoCommandHandler : ICommandHandler<CreateTodoCommand, Todo>
{
    private readonly TodoRepository _todoRepository;

    public CreateTodoCommandHandler(TodoRepository todoRepository)
    {
        _todoRepository = todoRepository;
    }

    public Task<Todo> HandleAsync(CreateTodoCommand command)
    {
        var todo = new Todo()
        {
            Name = command.Name
        };

        // Save to the repository
        _todoRepository.Add(todo);

        // Return the created entity
        return Task.FromResult(todo);
    }
}
```

CQRS

Query Example

```
public class GetTodosQuery : IQuery<List<Todo>>
{
}
```

```
public class GetTodosQueryHandler : IQueryHandler<GetTodosQuery, List<Todo>>
{
    private readonly TodoRepository _todoRepository;

    public GetTodosQueryHandler(TodoRepository todoRepository)
    {
        _todoRepository = todoRepository;
    }

    public Task<List<Todo>> HandleAsync(
        GetTodosQuery query,
        CancellationToken cancellationToken)
    {
        return Task.FromResult(_todoRepository.ToList());
    }
}
```

CQRS

Setup CQRS library

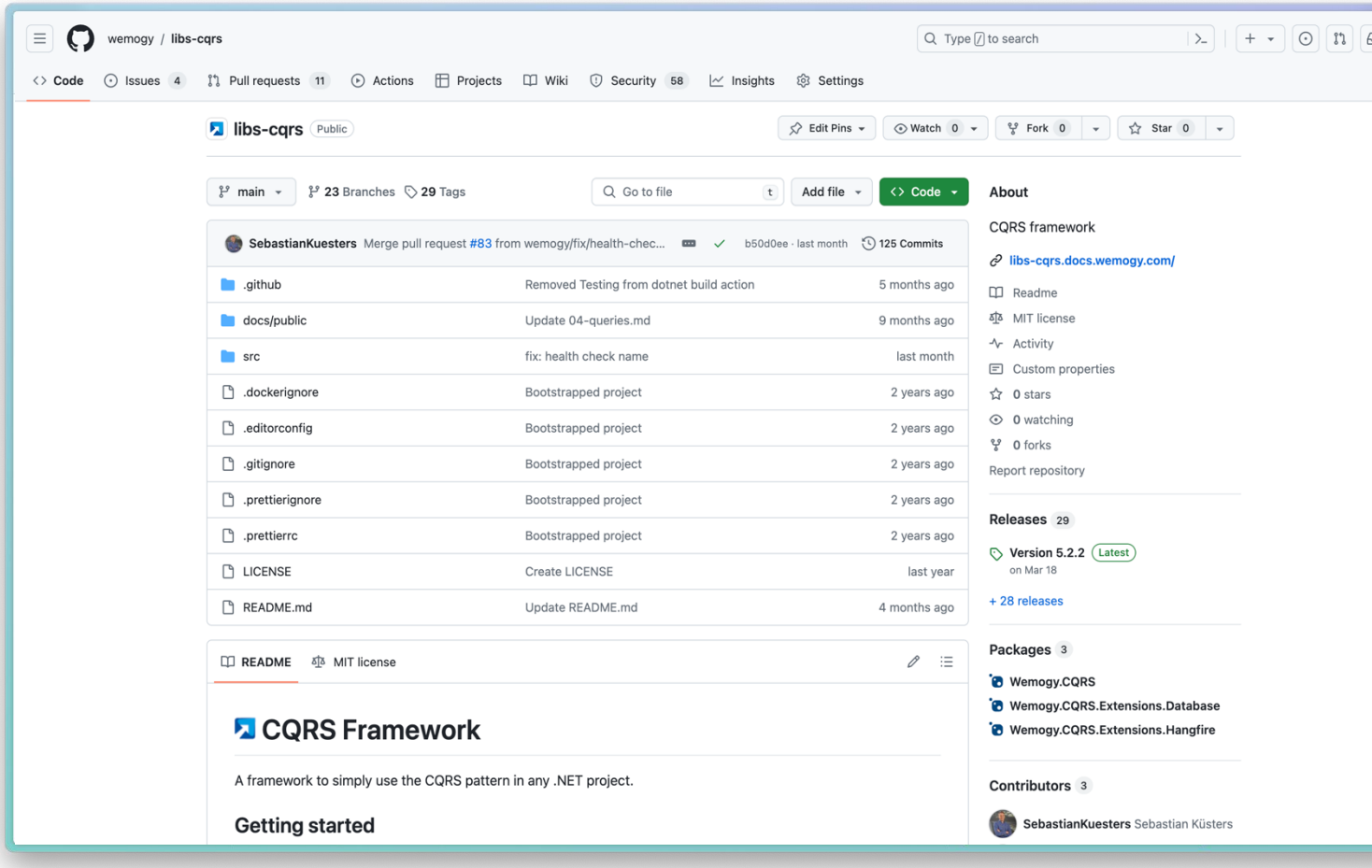
CQRS Libraries for .NET

MediatR

wemogy.CQRS

Register CQRS in Dependency Injection (in wemogy.CQRS)

```
builder.Services.AddCQRS();
```



HEALTH CHECKS

HEALTH CHECKS

Setup Health checks

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

builder.Services.AddControllers();

// Setup HealthChecks
builder.Services.AddHealthChecks();

var app = builder.Build();

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

// Add health checks endpoint
app.MapHealthChecks("/healthz");

app.Run();
```

HEALTH CHECKS

Create a custom health check

Implement the IHealthCheck interface

```
public class MyCustomHealthCheck : IHealthCheck
{
    private static int _counter = 0;

    public Task<HealthCheckResult> CheckHealthAsync(HealthCheckContext context, CancellationToken cancellationToken = default)
    {
        if (_counter++ % 2 == 0)
        {
            return Task.FromResult(HealthCheckResult.Healthy());
        }

        return Task.FromResult(HealthCheckResult.Unhealthy());
    }
}
```

Register the custom health check

```
// Setup HealthChecks
builder.Services.AddHealthChecks().AddCheck<MyCustomHealthCheck>("MyCustomHealthCheck");
```

CONTAINER & KUBERNETES

CONTAINERS AND KUBERNETES

Use Health checks

```yaml
apiVersion: v1
kind: Pod
metadata:
name: myapp-pod
spec:
  containers:
  - name: myapp-pod
    image: my-image
    resources:
      requests:
        memory: "512Mi"
        cpu: "256m"
    limits:
        memory: "1024Mi"
       cpu: "500m"
    ports:
    - containerPort: 8080
```

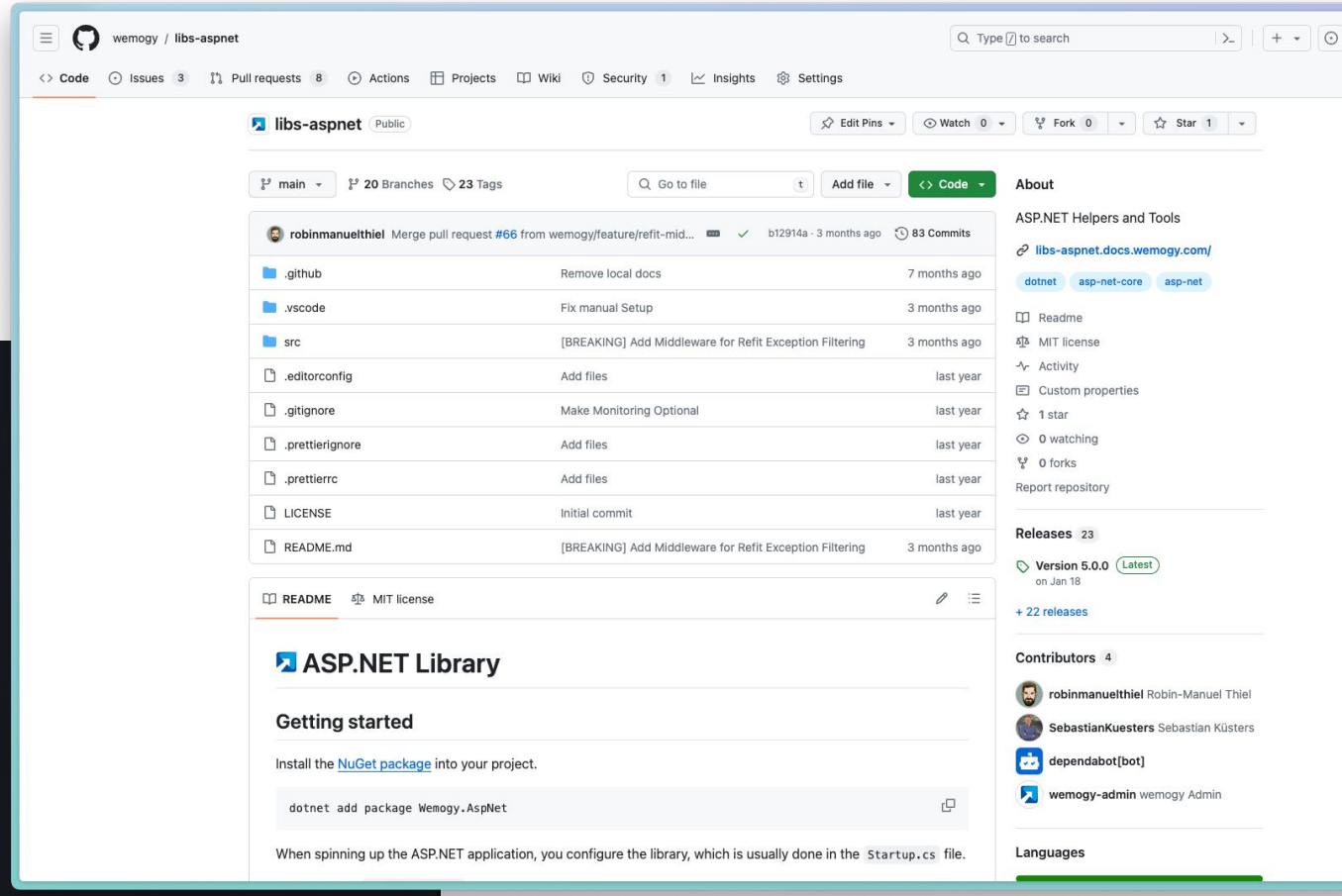
```yaml
livenessProbe:
  httpGet:
    path: /healthz
    port: http
    initialDelaySeconds: 30
    periodSeconds: 10
    failureThreshold: 3
    timeoutSeconds: 10
readinessProbe:
  httpGet:
    path: /healthz
    port: http
    initialDelaySeconds: 5
    periodSeconds: 10
    timeoutSeconds: 10
```

LIBRARY

github.com/wemogy/libs-aspnet

Setup Code Sample

```
var options = new StartupOptions();

// Middleware
options
    .AddMiddleware<ApiExceptionFilter>();

// Swagger
options
    .AddOpenApi("v1")
    .WithApiGroup("public", "Fancy API", "This is my fancy API.")
    .WithSecurityScheme(SecuritySchemeDefaults.JwtBearer);

// Monitoring
options
    .AddMonitoring()
    .WithMeter(Observability.Meter.Name)
    .WithApplicationInsights(Configuration["AzureApplicationInsightsConnectionString"])
    .WithPrometheus();

// Health Checks
options.ConfigureHealthChecks(builder =>
    builder.AddCheck("MyHealthCheck", () => HealthCheckResult.Healthy("Everything is fine."));

app.AddDefaultSetup(options);
app.UseDefaultSetup(app.Environment, options);
```



WHAT ARE YOUR QUESTIONS?

Full Sample Code:
https://github.com/wemogy/dotnet-8-in-production-sample

Space Blocks

Pricing Documentation Blog Developer Portal

Book a demo

The platform for permission management.

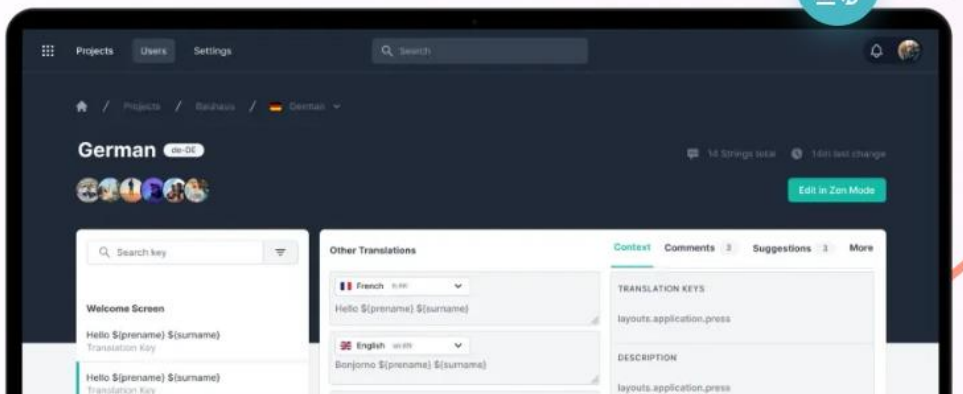
Create roles and access rights for your applications and perform permission checks against our service with ease.

- Free Developer tier
- No credit card needed
- No Setup

Get started

Identity

Subscriptions



Realtime

www.todocast.io

Jeden 2. Montag eine neue Folge