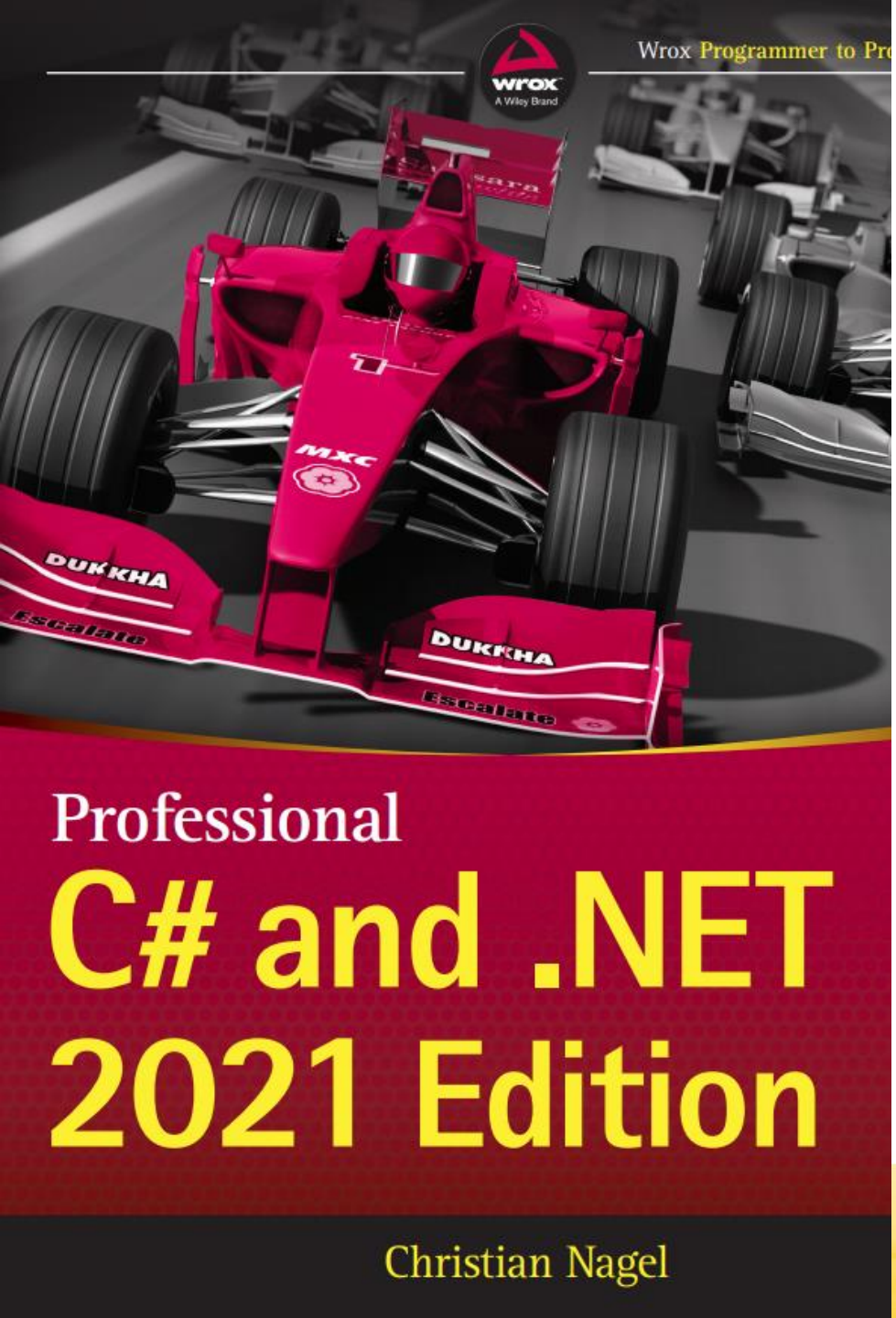
Async Streams

Christian Nagel

https://csharp.christiannagel.com



Christian Nagel

- Training
- Coaching
- Consulting
- Development

- Microsoft MVP
- www.cninnovation.com
- csharp.christiannagel.com
- @christiannagel

BASTA!

Topics

- Foundations
- Libraries

Foundations

What's the yield statement?

- *yield return*
- *yield break*
- Creates implementations for *IEnumerator*, *IEnumerable*

```
public IEnumerable<SomeData> GetSomeData()
{
    for (int i = 0; i < 100; i++)
    {
        int x = Random.Shared.Next(1, 200);
        yield return new SomeData($"text {x}", x);
        Thread.Sleep(100);
    }
}
```

yield & foreach

```
ADevice dev = new();

foreach (var item in dev.GetSomeData())
{
    Console.WriteLine($"{item.Number}, {item.Text}");
}
```

- *yield* implements *IEnumerable* and *IEnumerator* interfaces
- *foreach* uses *IEnumerable* and *IEnumerator* interfaces

Asynchronous APIs

- *Task, ValueTask, GetAwaiter*
  - Return a single result
- *IAsyncEnumerable*
  - Return a result stream

```
public async IAsyncEnumerable<SomeData> GetSomeDataAsync(
  [EnumeratorCancellation] CancellationToken cancellationToken = default)
{
    for (int i = 0; i < 1000; i++)
    {
        int x = Random.Shared.Next(1, 200);
        yield return new SomeData($"text {x}", x);
        await Task.Delay(100, cancellationToken);
    }
}
```

IAsyncEnumerable

BASTA!

await foreach

```
await foreach (var item in dev.GetSomeDataAsync()
                .WithCancellation(cts.Token))
{
    Console.WriteLine($"{item.Number}, {item.Text}");
}
```

Sync vs Async

Sync

- *IDisposable*
- *IEnumerable*
- *IEnumerator*

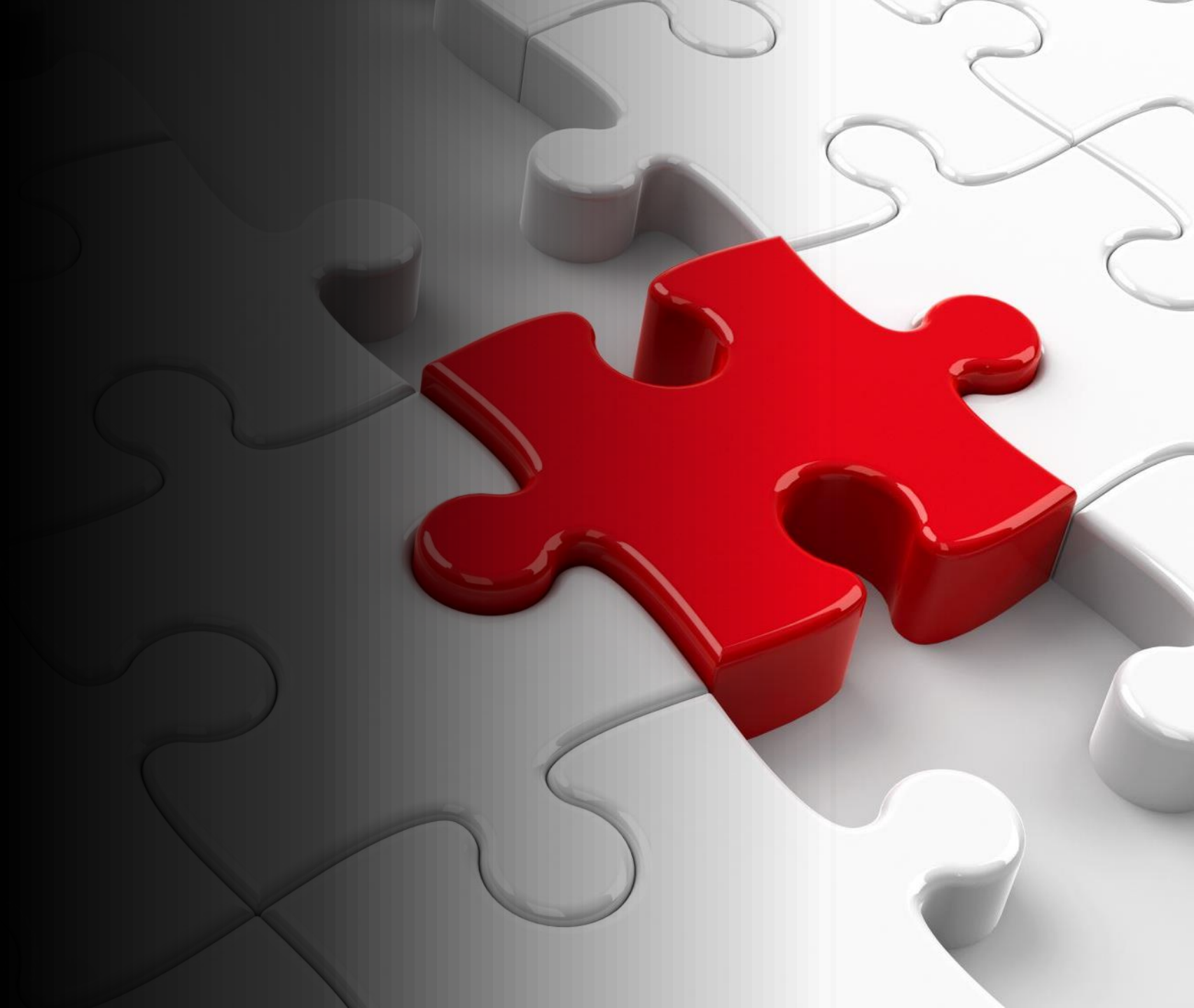
Async

- *IAsyncDisposable*
- *IAsyncEnumerable*
- *IAsyncEnumerator*

Libraries

EF Core (.NET 6)

- Mapping relational and NoSQL databases to objects
- Stream results asynchronously with…
  - *GetAsyncEnumerator*
    - *IAsyncEnumerator*
  - *AsAsyncEnumerable*
    - *IAsyncEnumerable*



Asynchronous Streaming with SignalR

- SignalR: real-time server-client communication plus streaming

- ChannelReader / ChannelWriter gives more control
- Async streaming is easier to use

SignalR - Server

```
public async IAsyncEnumerable<SensorData> GetSensorData(
  [EnumeratorCancellation] CancellationToken cancellationToken)
{
    for (int i = 0; i < 1000; i++)
    {
        yield return new SensorData(Random.Shared.Next(20), Random.Shared.Next(20), DateTime.Now);
        await Task.Delay(1000, cancellationToken);
    }
}
```

SignalR - Client

```
await foreach (var data in connection
    .StreamAsync<SensorData>("GetSensorData")
    .WithCancellation(cts.Token))
{
    Console.WriteLine(data);
}
```

gRPC

- Platform-independent binary communication…
- …including streaming

Async Streaming with gRPC - Server

```
public override async Task GetSensorData(Empty request,
                                         IServerStreamWriter<SensorData> responseStream,
                                         ServerCallContext context)
{
    while (!context.CancellationToken.IsCancellationRequested)
    {
        await Task.Delay(100, context.CancellationToken);
        SensorData data = new()
        {
            Timestamp = Timestamp.FromDateTime(DateTime.UtcNow),
            Val1 = Random.Shared.Next(100),
            Val2 = Random.Shared.Next(100)
        };
        Console.WriteLine($"returning data {data}");
        await responseStream.WriteAsync(data);
    }
}
```

Async Streaming with gRPC - Client

```
using var stream = _sensorClient.GetSensorData(new Empty());

await foreach (var data in stream.ResponseStream.ReadAllAsync().WithCancellation(cts.Token))
{
    Console.WriteLine($"data {data.Val1} {data.Val2} {data.Timestamp.ToDateTime():T}");
}
```

ASP.NET Core Controller

```
[HttpGet]
public IAsyncEnumerable<SomeData> GetSomeData()
{
    return _dataContext.SomeData.AsAsyncEnumerable();
}
```

Calling API Controller

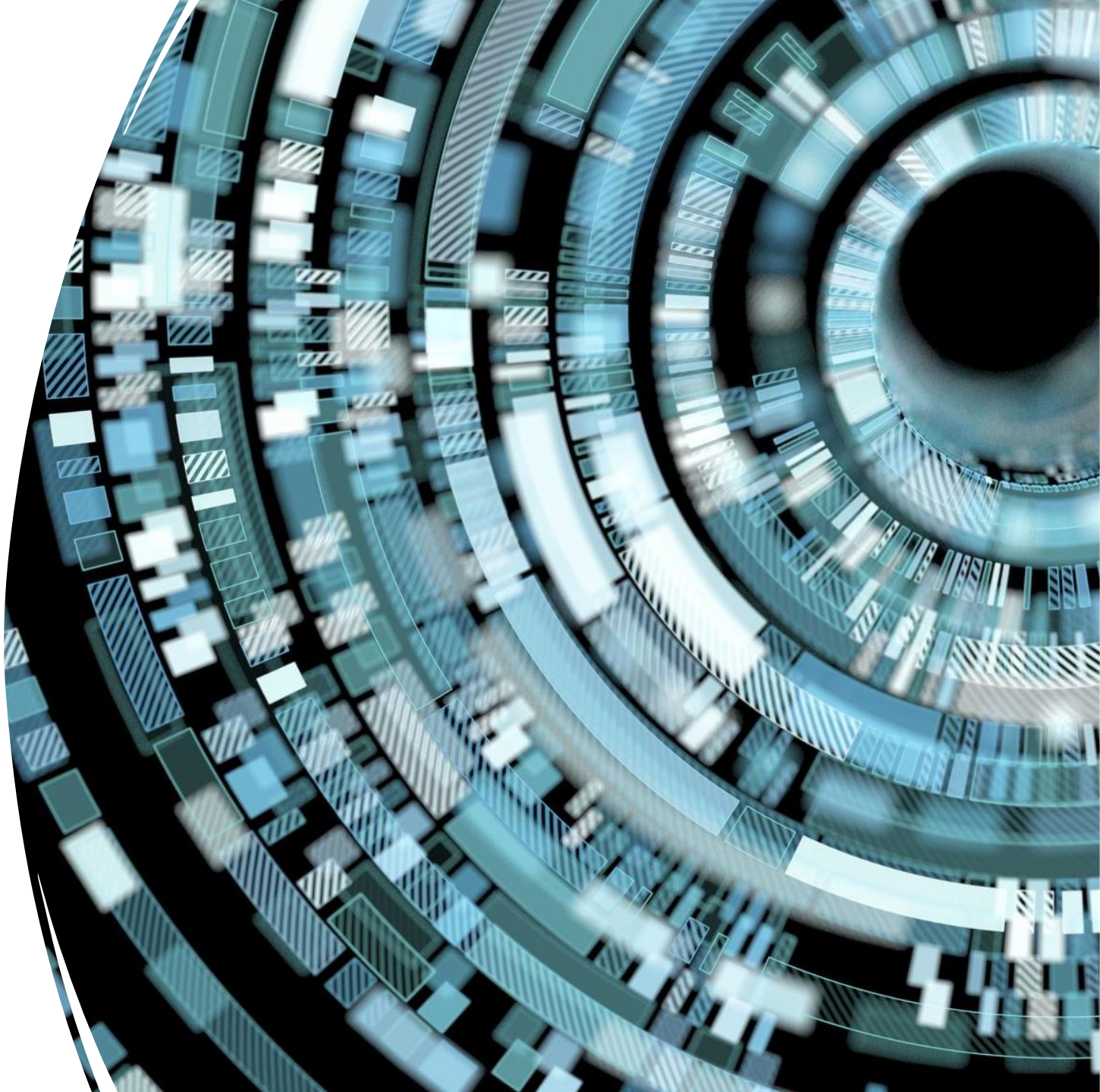
```
using HttpResponseMessage response = await httpClient.GetAsync(
  "https://localhost:5001/api/ADevice",
  HttpCompletionOption.ResponseHeadersRead).ConfigureAwait(false);

response.EnsureSuccessStatusCode();
using Stream responseStream = await response.Content.ReadAsStreamAsync().ConfigureAwait(false);

await foreach (SomeData? data in JsonSerializer
  .DeserializeAsyncEnumerable<SomeData>(responseStream,
    new JsonSerializerOptions { PropertyNameCaseInsensitive = true, DefaultBufferSize = 128 }))
{
    Console.WriteLine(data);
}
```

Azure Storage

- Iterate through a large list of blobs
- Working with the continuation token is abstracted
- *AsyncPageable* implements *IAsyncEnumerable*



Summary

- *Task* returns a result
- *IAsyncEnumerable* returns a continuous stream of results
- *yield* and *foreach* statements enhanced
- Many libraries support async streams

Thank you!

- Questions?
- Enjoy BASTA! Frankfurt!

- Source code: https://github.com/cnilearn
- More information: https://csharp.christiannagel.com

BASTA!