Module 10:
"New C# 8.0 Async Features"

TEKNOLOGISK
INSTITUT

New C# 8.0 Async Features

- Use new types only in .NET Core 3.x!

- Async Enumerables a.k.a. "Async Streams"
- **`await foreach`** keyword
- Async Disposables
- **`await using`** keyword

IEnumerable<T>

- The traditional **IEnumerable<T>** designates a sequence for use with **foreach** or LINQ.

```
namespace System.Collections.Generic
{
    interface IEnumerable<out T> : IEnumerable
    {
        IEnumerator<T> GetEnumerator();
    }
}
```

```
interface IEnumerator<T>
{
    T Current { get; }
    bool MoveNext();
    void Reset();
}
```

IAsyncEnumerable<T>

- `IAsyncEnumerable<T>` designates an asynchronous sequence for use with **`await foreach`**

```
namespace System.Collections.Generic
{
    interface IAsyncEnumerable<out T>
    {
        IAsyncEnumerator<T> GetEnumerator(CancellationToken cts = default);
    }
}
```

```
interface IAsyncEnumerator<T>
{
    T Current { get; }
    ValueTask<bool> MoveNextAsync();
}
```

IDisposable

- Traditionally, .NET has **`IDisposable`** interface built-in for implementing Dispose Pattern

```
public interface IDisposable
{
    void Dispose();
}
```

- The **`using`** keyword can be applied to ensure **`Dispose()`** is always invoked.

IAsyncDisposable

- Now, for asynchronous disposal .NET Core 3.x has **IDisposableAsync** interface built-in for implementing Dispose Pattern

```
public interface IAsyncDisposable
{
    ValueTask DisposeAsync();
}
```

- The **await using** keyword can be applied to ensure **DisposeAsync()** is always invoked.


WINCUBATE
Jesper Gulmann Henriksen
PhD, MCT, MCSD, MCPD
Phone : +45 22 12 36 31
Email : jgh@wincubate.net
WWW : http://www.wincubate.net
Ringgårdsvej 4A
8270 Højbjerg
Denmark