Parallel and Asynchronous Programming with .NET

Dr. Dietrich Birngruber

Course Outline

- Introduction
- Recapitulation: Functional programming with C#
- Threading basics in .NET
- Asynchronous Programming Patterns in .NET
- Parallel Programming Patterns in .NET
- Asynchronous programming and distributed systems
- Projects

Introduction

Who is …

Async & Parallel: What is it good for?

Dietrich - Personal Background
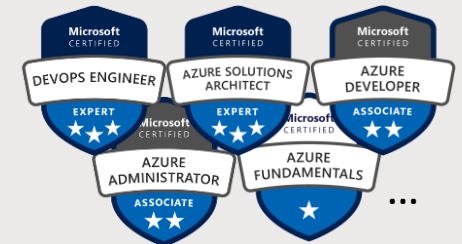
What is Dietrich all about?

- Integrity, trust, professional
- Curious, passionate, loyal
- Empower others, be transparent
- Results oriented, not finger pointing

Education

- Dr., MSc., Business Informatics,
  Johannes Kepler University Linz (Austria)
- German native speaker
- English fluent
- French basic, some Spanish (un poco)
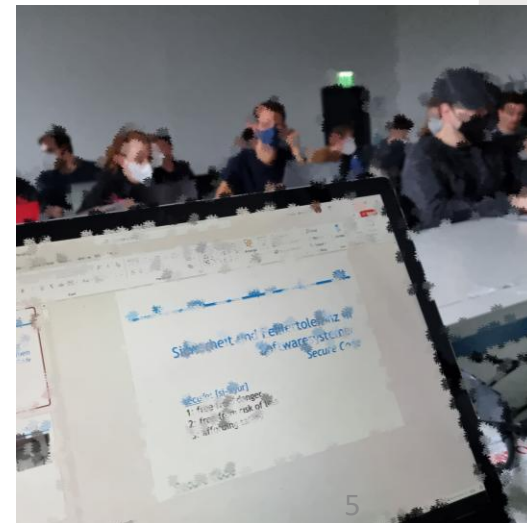
birngruber@acm.org

Personal Background

Words I live by

- Inspire others by example
- Keep learning and improving
- I 💗 working

Outside of work

- Family + dog
- Farming
- Skiing, movies, wine
- External lecturer at two universities

Dietrich's main topics @ avanade

- Director, Software Engineering, Austria – it's a people business
  - Know our people
  - Support *AT* Software Engineering, e.g., inspire "community", hiring, delivery
  - Support ASG Software Engineering
- Sales support for selected accounts
  - e.g., for "digital products", "application modernization"
  - Runs on Azure or on-premise?
  - "Technical" SME, Azure in AT
- Delivery
  - Customers in Health & Public Sector, Production industry

Avanade is the world's leading Microsoft expert

- Founded in 2000 as a joint venture between Accenture and Microsoft, Avanade is the world's most experienced innovator on the Microsoft platform. With unique industry insights, unrivaled expertise and breadth of services, our 60,000 people **do what matters** for our clients and their customers every day.

60,000
Skilled and diverse professionals – **33%** of whom are women

60,000+
Microsoft certifications, more than any other partner

5,000+
global clients since inception

6
Solutions Partner Designations

+ the 7th coveted **Microsoft Cloud Badge**

92
Locations across **26** countries

18x
Winner of Microsoft Global SI Partner of the Year

avanade

Do what matters

Experience

- "battle-scarred agile purist": Kanban, Scrum, LeSS
- digital product development - software engineering - RnD
  - Scrum master
  - Product owner
  - Product development teams (C#, platform, services, …)

- Director, @Avanade, ASG, AT, Software Engineering
- Program Architect @Microsoft, ISD, Azure Cloud & AI
  - Technical presales & delivery for s500 customers in EMEA
  - Technical SPOC for "big" delivery programs for customer C-level & program team members
  - "Agile expert" & industry focus (manufacturing, logistics)
  - participated in Supply Chain v-team
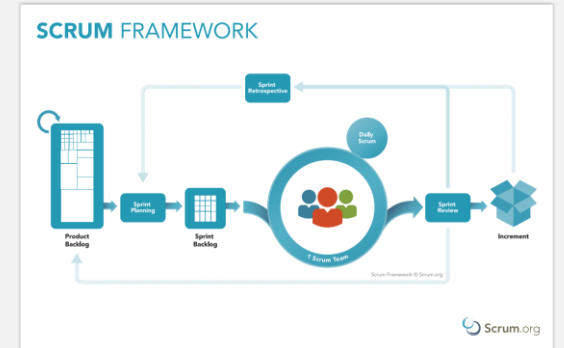
Global customer organisations from Retail, Discrete-, Process- Manufacturing. Health & Public Sector

Experience

- 17 years in logistics / manufacturing companies
  - 4 years Head of Software Development,
    laser machine manufacturer,
    build to order (build to stock)
  - 13 years intra-logistics as PO, SM, architect (lead dev),
    development manager,
    supply chain, automated fulfilment centres,
    engineered to order
- Digital transformation: "change agent"
  - Agile mindset – role model & talk about it ;-)
  - Communicate with CxOs & with operational level
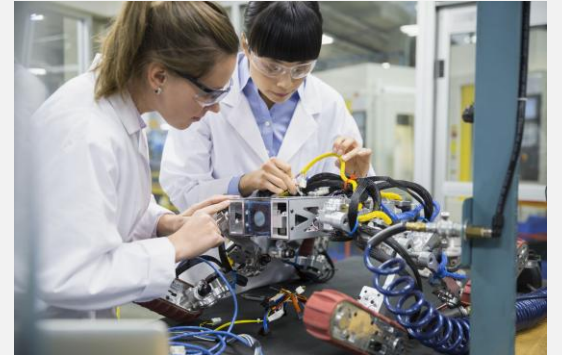
Administrative Stuff

- KV

- Sample Code:
https://github.com/birngruber/ParallelAndAsyncDemos

- You do a project and present it → Proposals?
  - You do a live demo
  - You create document & slides & VS.NET / VS Code project

Parallel & Async: What is it good for?

- Method execution takes a „long" time (> 20 - 50 ms)
  - Consumes CPU resources
  - Imagine calling a long running method in a loop 1000 times
    e.g. executing one method takes 60 ms → caller blocked for 60 seconds

- When crossing I/O boundaries
  - E.g. accessing file, send a request to a server, …

What is it good for?

- Responsive UI Clients
  - E.g.: user interface where 1-N persons interact with a multi-touch application running on tablets, notebooks, phones, large multi touch screens, industrial workplaces, …

- Server / app should process 'many' messages 'at once'
  - E.g.: download a picture, search free storage positions, emulate devices, analyze user click streams, process events from 2000 devices 'at once' …

- …

What is it good for?

- We can develop „ASYNC" methods
  - IO intensive, e.g., IO access / download ("IO bound"),
  - long running methods (> 20 ms),
  - Server processes thousand requests, UI (multi-touch), …

- We can develop „PARALLEL" (data) algorithms
  - CPU intensive ("CPU bound")
  - „delightfully" parallel loops (vs. "embarrassingly parallel algorithms" ;-)),
  - search algorithms, pack patterns, optimizations, ….

Goals – and non Goals / Expectations

- You know some tools which the .NET framework and C# offer for writing asynchronous code

- You know some tools which the .NET framework and C# offer for writing parallel algorithms

- You will not learn parallel (design & coding) principles en detail
  - Just mentioned along the way

Course Outline – Where are we?

- Introduction
- **Recapitulation: Functional programming with C#**
- Threading basics in .NET
- Asynchronous Programming Patterns in .NET
- Parallel Programming Patterns in .NET
- Asynchronous programming and distributed systems
- Projects

Functional Programming with C#

Recapitulation – you already know this, right?

Recapitulation: What is …

- A delegate in .NET?
- An Action?
- A Func(tion)?
- An anonymous function?
- To yield?
- A Lamdba expression?
- A LINQ statement?

Recap: Delegates

- Functions are first class citizens in .NET

- But .NET is a typed OOP platform, so …

- Delegates are typed functions
  - A pointer to a typed function

- So there must be type checks … when?

- Event handlers are delegate types

Demo

Delegates Sample

Recap: Delegates Sample

```
//definition of a delegate type
public delegate void DoWhat(string argument);
...
// declare delegate variable
DoWhat logInfo;

//assign a method with the same argument + return type signature
logInfo = MyLog.WriteInfo;

//later, use the delegate and invoke the assigned method
logInfo.Invoke(„Hello");     // logInfo("Hello");
```

Delegates, Delegates, …

- "type explosion"
- how many delegate types exists in a single program?

Demo

Actions and Functions Sample

Recap: Actions and Functions Sample

Action<string> log = MyLog.WriteInfo;

Func<int, int, int> mathOp = MyMath.Add;

log.Invoke(„hello, this is a log entry. X -> {0} ", mathOp(1,2));

log = MyLog.WriteError;

log.Invoke(„hello, this is a log entry. X -> {0} ", mathOp(1,2));

//what is displayed on the console?

Recap: Anonymous Methods and Functions

- unnamed "inline" statement or expression block, that can be used wherever a delegate type is expected (since .NET 2.0)

- Makes writing actions and functions very simple

```
Func<int, int, int> f = (a, b) => { return a + b; };
int x = f(3,1);     // returns 4
Action<string> aLog = delegate (string msg) { Console.WriteLine(msg); };
aLog("Hello");      // writes Hello to the console out stream
```

Questions:

- .NET runtime executes IL. Where is the anonymous method?

- Should we always use anonymous methods?

- How can we re-use anonymous methods?

Recap: yield return Statement

- Returning IEnumerator.MoveNext value
- Calculating the values instead of keeping them in memory (e.g. in a list)
- Returning inhomogeneous values
- Can't be used inside anonymous methods
- What about ThreadContext switches??

Demo

Yield

(Demo 05)

Yield Sample

```
foreach(Galaxy aGalaxy in Galaxies) { ... }


public System.Collections.Generic.IEnumerable<Galaxy> Galaxies {
 get
 {
  yield return new Galaxy { Name = "Tadpole", MegaLightYears = 400 };
  yield return new Galaxy { Name = "Pinwheel", MegaLightYears = 25 };
  yield return new Galaxy { Name = "Milky Way", MegaLightYears = 0 };
  yield return new Galaxy { Name = "Andromeda", MegaLightYears = 3 };
 }
}
```

Recap: Language Integrated Query (LINQ), Lambda Expressions, Extension Methods

- Use LINQ for querying collections of objects stored in memory, XML, in a database, …

- LINQ uses functional programming constructs, such as Lambda expressions, delegates, functions, anonymous methods

- LINQ uses extension methods
  - you know, the „strange" static ones

Demo

LINQ

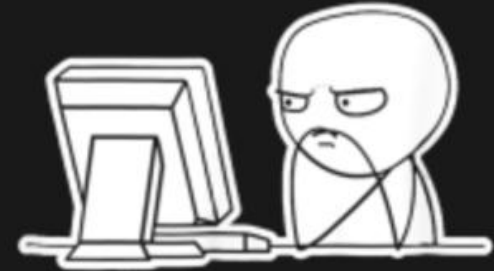# LINQ Sample - With an Extension Method

List<string> names = new List<string> { "Jack", "Jill", "Lago" };

var q = from n in names

    where n.StartsWith("J")

    select n;

q.Iterate(aName => Console.WriteLine(aName));

```
public static void Iterate(this IEnumerable<string> collection, Action<string> action)
{
   foreach (string anItem in collection) { action(anItem);}
}
```
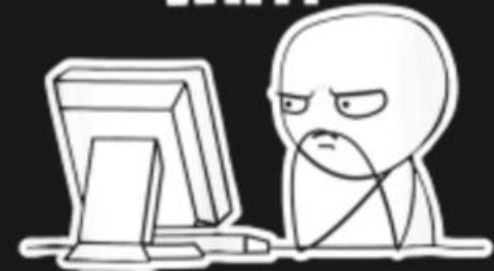
What is a Closure in C#?

WTF?

What is a Closure in C#?

- Encapsulates some behaviour

- Pass it around like any other object

- Have access to the context in which they were first declared

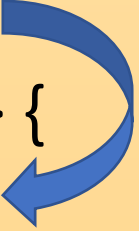- Well known concept in functional programming

Demo

Closure

Closure Sample

```
int x = 1;
Action a = () => {
        x++;
        Console.WriteLine(x);
  };
a();
//what is the value of x after this method?
// x == 1 or x == 2 ?
```

Summary: You Know

- Why bother with async and parallel programming at all?
- Delegates
- Action<>, Func<>
- Anonymous Actions / Functions
- yield return, yield break
- Lambda expressions, Closures
- LINQ statements