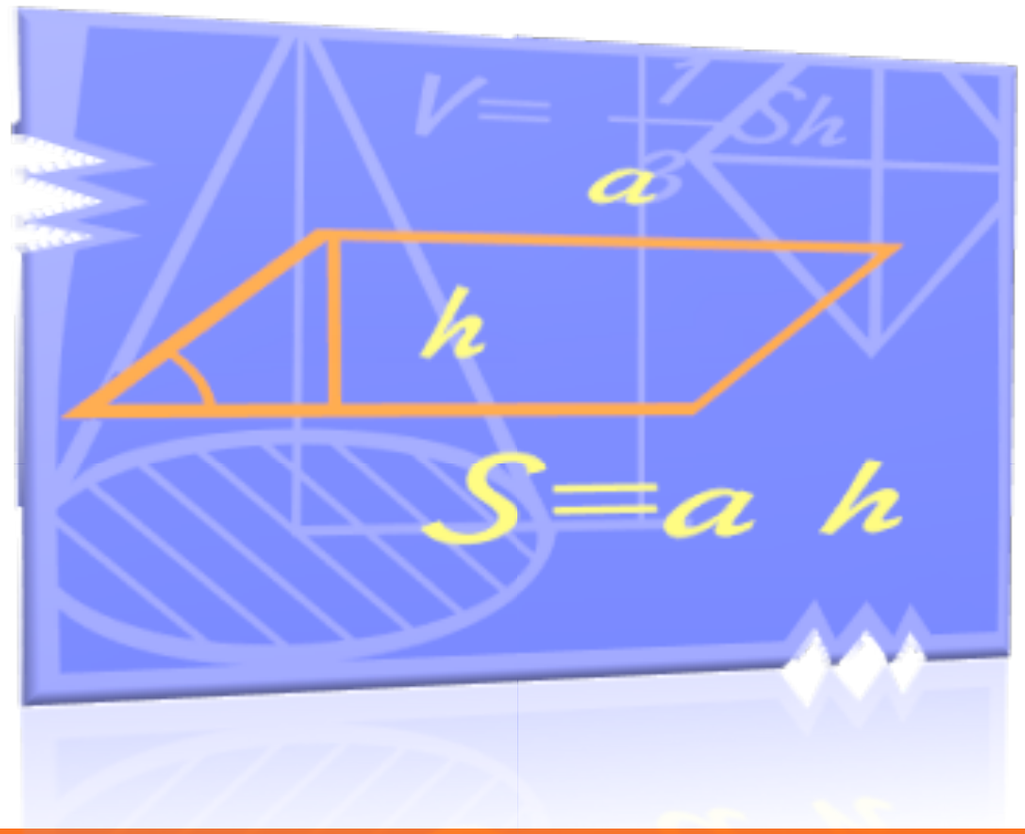
Functional Programming in C#



Oliver Sturm
oliver@oliversturm.com

http://www.oliversturm.com

Oliver Sturm

thinktecture Associate

ineta

MVP Microsoft Most Valuable Professional

Copyright © 2011 Oliver Sturm

Oliver Sturm (@olivers)

- **Consultant and Trainer**
- **Associate Consultant at thinktecture**

- **.NET Application System Architecture**
  - User Interfaces
  - Data Handling / Data Access Architectures
  - Programming Languages
  - DevExpress Component/Framework Products

- **Microsoft MVP for C#**
- **INETA Europe Speaker**

- **Services: http://www.oliversturm.com**
- **Blog: http://www.sturmnet.org/blog**

- **oliver@oliversturm.com**

Copyright © 2011 Oliver Sturm

Tweet and Win!

- **Tweet something about this talk**
- **Include these details in your tweet:**
  - **My Twitter name: @olivers**
  - **The following hashtag: #ddd9fpcs**
- **Win this brilliant prize at the end of the talk:**

A DevExpress CodeRush/Refactor! license

Copyright © 2011 Oliver Sturm

Agenda

Tweet&Win - include these: @olivers #ddd9fpcs

Focus: What C# can do with regard to FP and how it works

- What is Functional Programming?
- FP Features introduced in C# 3.0 and .NET 3.5
- Map, Filter and Reduce
- Currying, Partial Application and Composition
- How does a C# programmer benefit from FP?

Copyright © 2011 Oliver Sturm

What is Functional Programming?

Tweet&Win - include these: @olivers #ddd9fpcs

- A programming paradigm
- Focus on the application of functions
- Avoids state and mutable data
- Well-known languages include Lisp, Scheme, Haskell, ML and (recently) F#
- FP languages tend to have features that support Higher Order Functions, currying, recursion, list comprehensions, ...
- Many imperative and OO languages have FP features today

Copyright © 2011 Oliver Sturm

Why is Functional Programming interesting?

Tweet&Win - include these: @olivers #ddd9fpcs

- Promotes modularization
- Lazy evaluation → greater efficiency
- Avoid and/or manage side-effects
- The target of avoiding side effects has several advantages: scalability, optimization, debugging, testing
- C# >= 3.0 supports many important FP techniques

Copyright © 2011 Oliver Sturm

Demo

Tweet&Win - include these: @olivers #ddd9fpcs

What's in the box

Copyright © 2011 Oliver Sturm

Interlude — Map, Filter, Reduce

Tweet&Win - include these: @olivers #ddd9fpcs

- **Map/Select** does something with each element in a list
- **Filter/Where** extracts elements from a list based on some condition
- **Reduce/Fold/Aggregate** summarizes elements in a list according to some calculation
- Select, Where and Aggregate are .NET 3.5 implementations of these functions

Copyright © 2011 Oliver Sturm

Demo

Tweet&Win - include these: @olivers #ddd9fpcs

What's in the box ... continued

Copyright © 2011 Oliver Sturm

Map, Filter, Reduce

Tweet&Win - include these: @olivers #ddd9fpcs

- **Map** does something with each element in a list
- **Filter** extracts elements from a list based on some condition
- **Reduce/Fold** summarizes elements in a list according to some calculation

Copyright © 2011 Oliver Sturm

Demo

Tweet&Win - include these: @olivers #ddd9fpcs

Map, Filter, Reduce

Copyright © 2011 Oliver Sturm

Currying and Partial Application

Tweet&Win - include these: @olivers #ddd9fpcs

- Currying: Convert a function that takes multiple parameters into a chain of functions that each take one parameter and return the next function, until the deepest nested function performs the calculation with all the values and returns the result.
- Partial Application: Fixing one or more parameters of a function in curried form, creating a new function with a more specific purpose.

Copyright © 2011 Oliver Sturm

Demo

Tweet&Win - include these: @olivers #ddd9fpcs

Manual and Automatic Currying

Copyright © 2011 Oliver Sturm

Demo

Tweet&Win - include these: @olivers #ddd9fpcs

Composition

Copyright © 2011 Oliver Sturm

Function Construction

Tweet&Win - include these: @olivers #ddd9fpcs

- The idea of creating new functions from existing ones
- Promotes modularization on a function level
- Partial Application is one way to do it
- Composition is another way:

Assuming `B = f1(A), C = f2(B)`

→ `f2(f1(A))`

Copyright © 2011 Oliver Sturm

Combining approaches

- Aim: create function `int sumOfOddNumbers(int)`, based on Reduce
- Using Partial Application to define accumulation strategy for Reduce, as well as algorithm for sequence creation
- Using Composition to allow for easier usage, simplify parameters

Copyright © 2011 Oliver Sturm

Demo

Tweet&Win - include these: @olivers #ddd9fpcs

Function Construction

Copyright © 2011 Oliver Sturm

FP in C# — what are the benefits?

Tweet&Win - include these: @olivers #ddd9fpcs

- Functional modularization is not easy to get used to, but very rewarding
- Unit testing can benefit from a no-side-effects philosophy
- Programming for scalability is easier, whether you use your own threads, thread pools or toolkits like ParallelFX
- It's easier to get things done – try it yourself!
- BUT: Make sure your team members understand it, too!

Copyright © 2011 Oliver Sturm

Summary

Tweet&Win - include these: @olivers #ddd9fpcs

- C# has good support for important Functional Programming ideas
- Some “manual” work is required
- Syntax is sometimes a bit weird
- FP provides Glueing techniques (Currying, Partial Application, Composition) on a function level, introduces an additional level of modularization

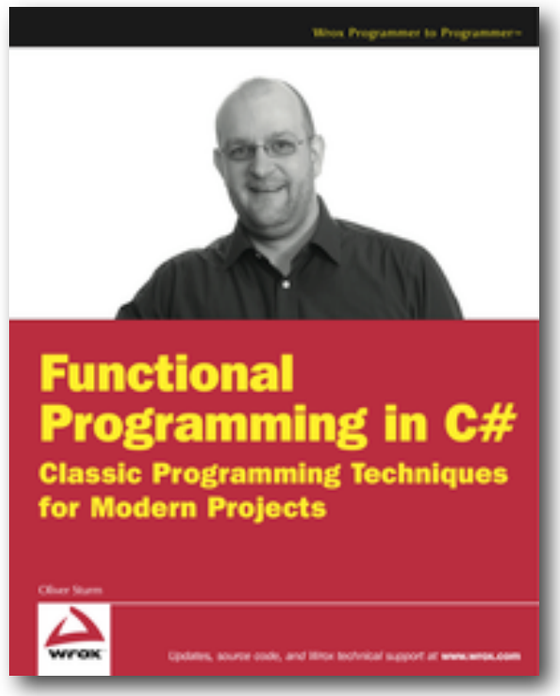
Copyright © 2011 Oliver Sturm

Thank you

Tweet&Win - include these: @olivers #ddd9fpcs

Please feel free to contact me about the content anytime.

oliver@oliversturm.com



Copyright © 2011 Oliver Sturm