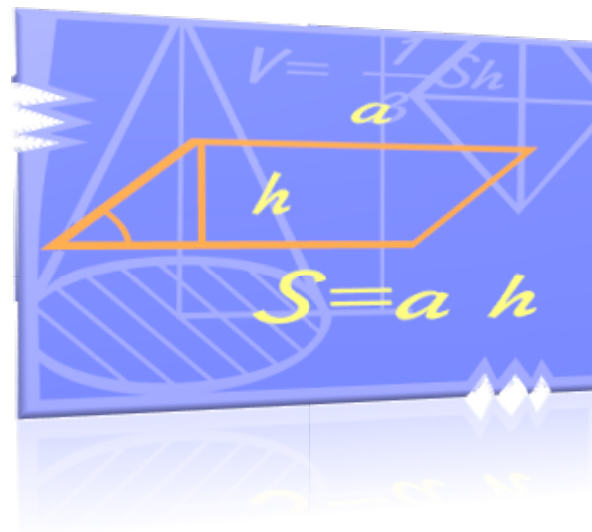


# Functional Programming in C#



Oliver Sturm  
[oliver@oliversturm.com](mailto:oliver@oliversturm.com)

<http://www.oliversturm.com>

Oliver Sturm  
.....

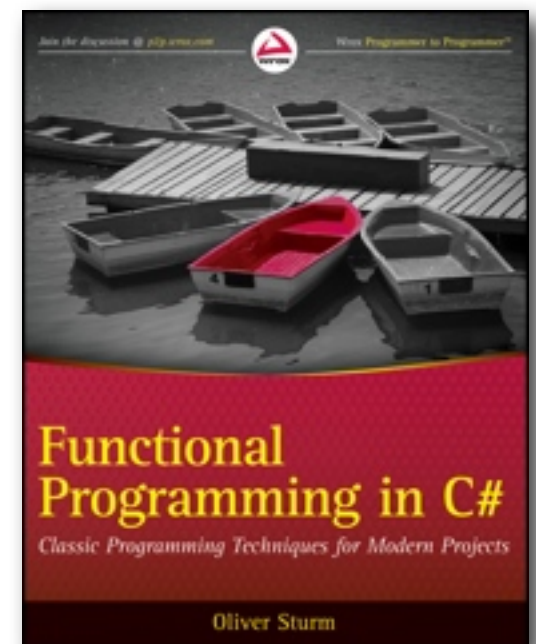
thinkecture  
Associate



# Oliver Sturm (@olivers)



- **Consultant and Trainer, Author**
- **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](mailto:oliver@oliversturm.com)**



# Agenda

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

- What is Functional Programming?
- FP Features 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?

# What is Functional Programming?

- 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

# Why is Functional Programming interesting?

- 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#  $\geq 3.0$  supports many important FP techniques

# Demo

What's in the box

# Interlude — Map, Filter, Reduce

- **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

What's in the box ... continued



# Map, Filter, Reduce

- **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

# Demo

Map, Filter, Reduce

# Currying and Partial Application

- **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.

## Manual and Automatic Currying

# Demo

## Composition

# Function Construction

- 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  $B = f_1(A), C = f_2(B)$   
 $\rightarrow f_2(f_1(A))$

# 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

## Function Construction



# FP in C# — what are the benefits?

- 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!

# Summary

- 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

# Thank you

Please feel free to contact me about the content anytime.

[oliver@oliversturm.com](mailto:oliver@oliversturm.com)

