Microsoft NT konferenca 2009

Concurrency using functional patterns in C#

Oliver Sturm - DevExpress
olivers@devexpress.com
oliver@sturmnet.org

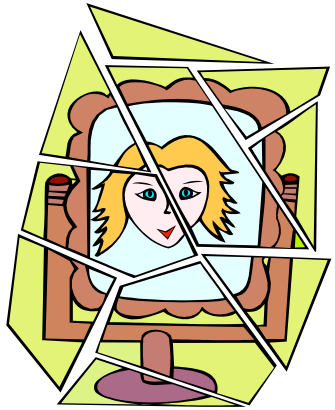
Microsoft Most Valuable Professional

ineta

Težavnost 300

Microsoft

Who's that Oliver Sturm guy, anyway?



- I am
  - Interested in programming languages, databases and a whole bunch of other things
  - Microsoft MVP for C#
- I do
  - Work for Developer Express as a Technical Evangelist and Lead Program Manager
  - Blog at http://www.sturmnet.org/blog
  - Podcast at http://www.sodthis.com
- You should
  - Email me at oliver@sturmnet.org
  - Follow me on Twitter @olivers

Težavnost 300

Microsoft NT konferenca 2009

Agenda

- The many-core shift
- Concurrency using Parallel Extensions to .NET
  - … and what we still need to do ourselves
- Relevant functional approaches
  - Black belt list filtering
  - The amazing parallel Mandelbrot

The many-core shift is upon us

- For some weird and highly physical reasons, single CPU cores don't get much faster anymore
- Instead, we have lots of CPU cores
  - Some estimations show that we might have around 256 (per CPU) within 10 years
  - This is true for all types of computers by now, not restricted to high-perf or server scenarios
- Result: if programs don't parallelize, they use an increasingly small percentage of the processing power that's available

Concurrency – really relevant?

- “Compute intensive tasks” are those that benefit from concurrency – obviously not all tasks are in that category

- But: what percentage of available computing power do you want to turn down? 50% (two cores, average laptop in 2009)? 75% (four cores, average desktop in 2009)? More?

Multi-threading vs. Concurrency vs. Parallelism

- Things going on on a single core can be multi-threaded
  - Perceived perf gains through “concurrent” updates
  - Hiding latency (network queries, …)
- Executing things literally concurrently or “in parallel” means (almost) the same to me
- Daniel Moth disagrees – let him ;-)
- He’s right about this though: multiple cores come with an opportunity to benefit from concurrency – this shouldn’t be missed!

Težavnost 300

Doing the splits

- The idea of having lots of processes and just one processor is old
- Now things change: several CPUs, more cores, too few tasks
- We need to split up large tasks in lots of chunks
  - The more chunks, the better – parallelism frameworks are more efficient that way

Demo

- TreeWalker
- Filtered list in the UI

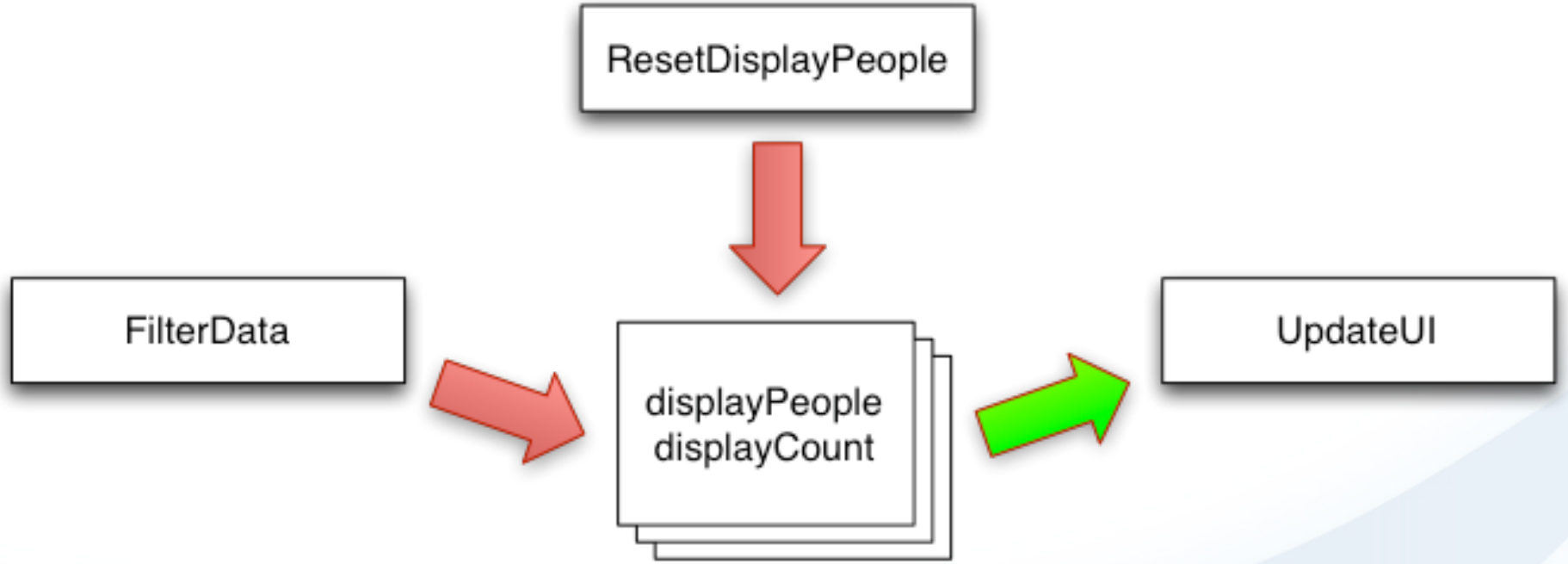
Težavnost 300

Microsoft NT konferenca 2009

Did you think that was ugly?

I wasn't even using array resizing!

Data access the way it shouldn't be



Težavnost 300

Microsoft NT konferenca 2009

Concurrency frameworks

- Example: Parallel Extensions to the .NET Framework
- Technical hurdles to multi-threading reduced considerably
- Data/state sharing somewhat simplified
  - Thread-safe data structures
  - Advanced mechanisms like software transactional memory
- Locking is becoming cheaper, but…

In the end, locks are bad

- Locks are expensive
  - They limit the amount of parallelization we can use
  - They carry a low-level cost
- They are structurally complicated to work with
- Result: the fewer locks we use, the better
- Algorithms should be structured carefully to work with data in conflicting ways as rarely as possible

Težavnost 300

Microsoft NT konferenca 2009

Functional approaches…

- … aren't strictly necessary for concurrency
- … offer one approach to a code structure which lends itself well to parallelization
- … are often used in optimization efforts without awareness of their origin
- At the core of FP ideas is function purity
- … which results – optimally – in the absence of global data/state

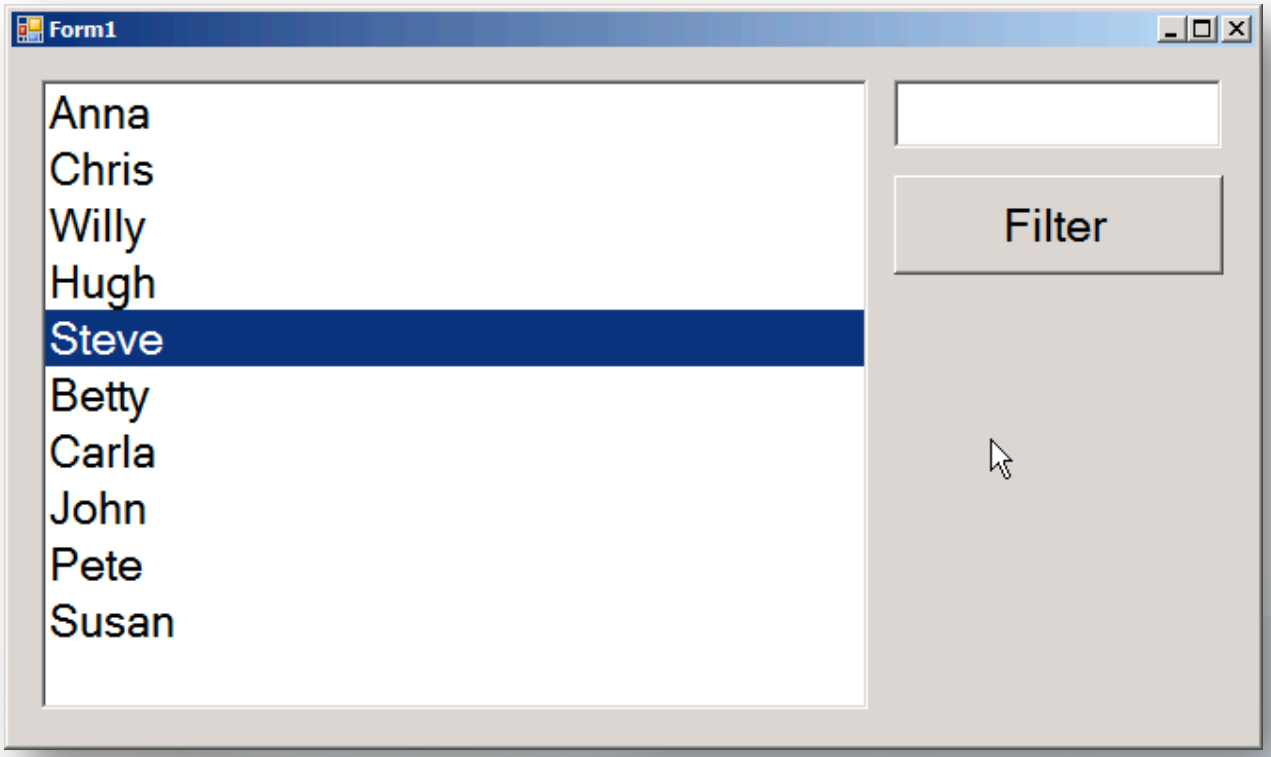
What a coincidence!

Functional approaches and concurrency

- There are “only” two approaches that are really important for concurrency
  - Try to work with immutable data
  - Try not to have data outside functions
- This won’t work all the time, so don’t worry
- It’s harder in imperative languages, because they don’t enforce the discipline
- The lack of support for tail recursion is a problem sometimes

Demo

- Filtered list in the UI (new and improved version)



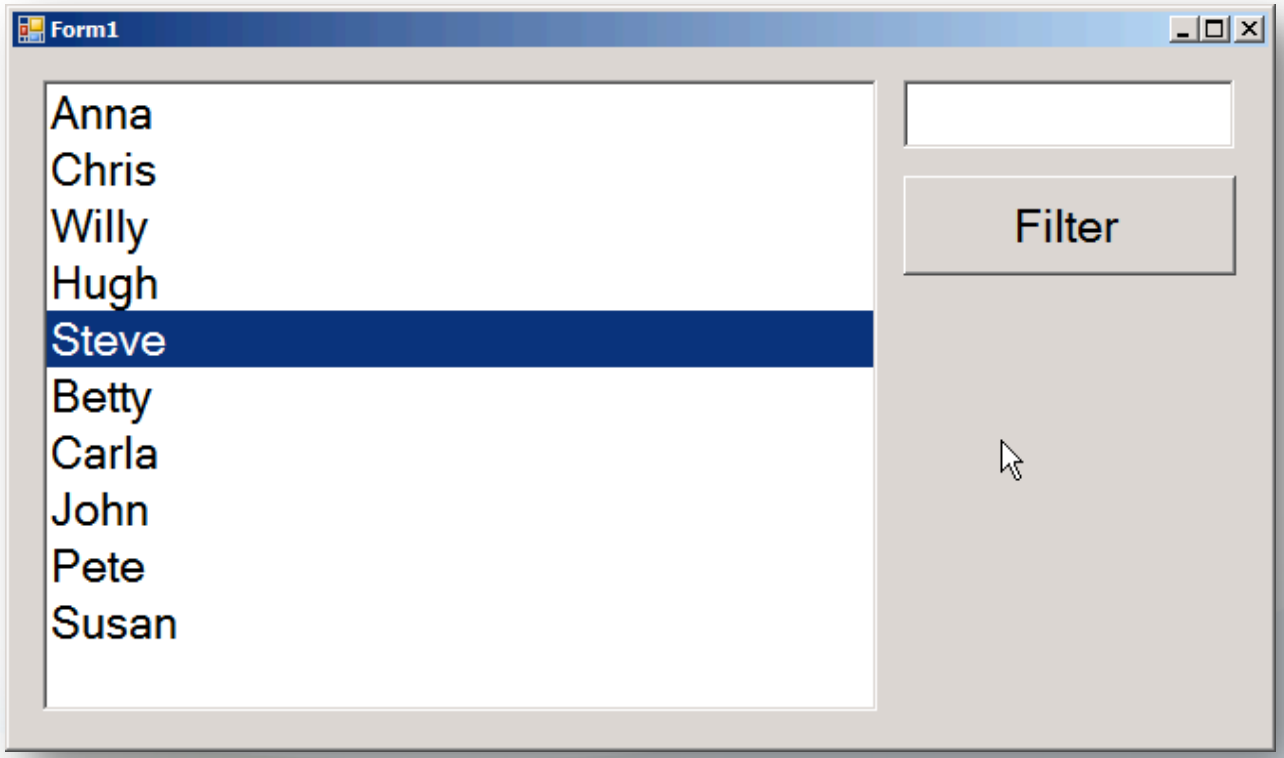
Težavnost 300

How this demo got better

- 11 fewer lines of code!
- Okay, seriously:
  - Simple functions with just a return or a statement
  - Easy to parallelize, since the algorithm is now encapsulated in one function
- But:
  - Data is regarded as immutable, but mutable data structures are still being used
  - One class-level field left

Demo

- Filtered list in the UI (supercharged functional version)



Težavnost 300

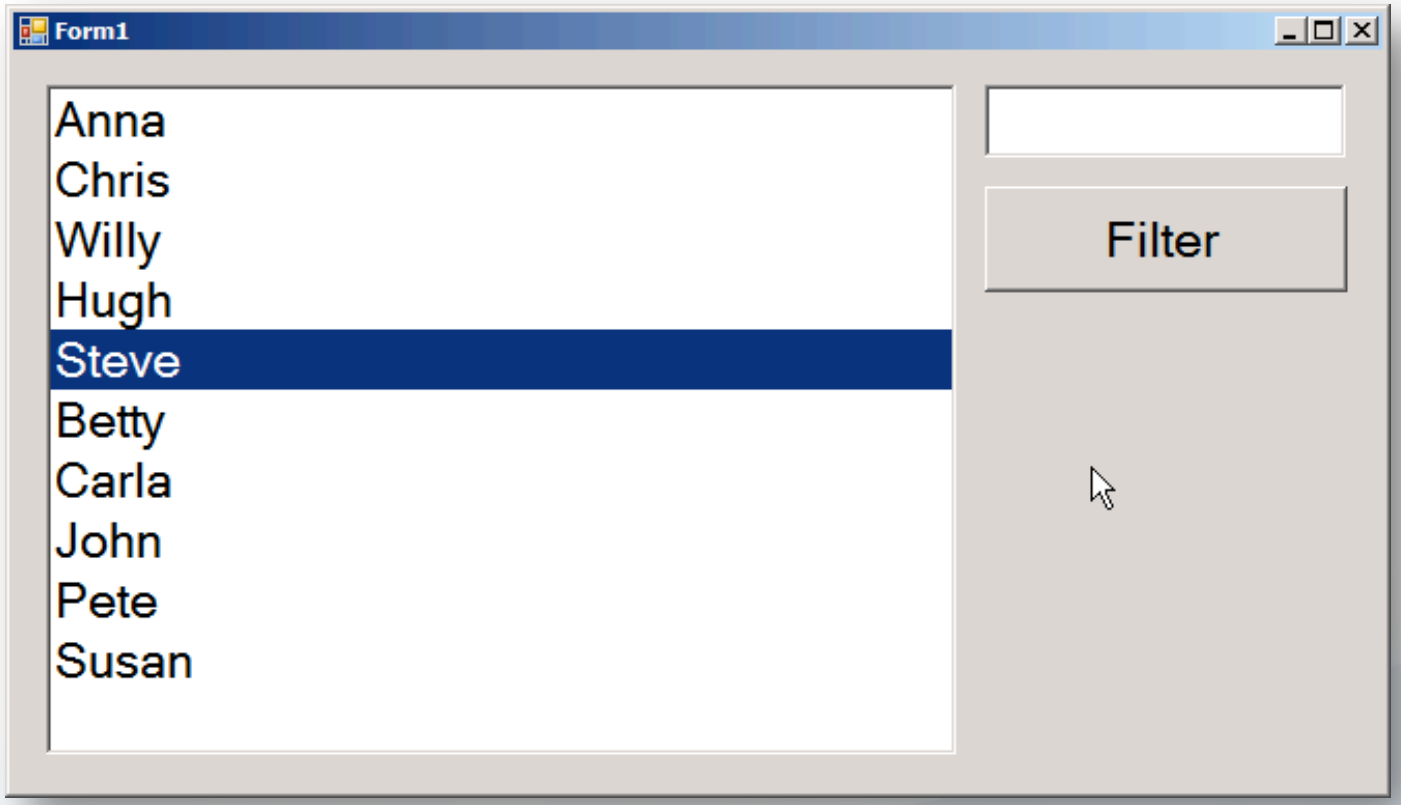
Microsoft NT konferenca 2009

What happened this time

- Yeah yeah… another 8 lines saved
- All functions are pure now
- Fully functional flow of data, no more storage outside functions
- But:
  - Event handlers are delegates that use closures, Windows Forms designer doesn't have a clue about this sort of thing

Demo

- Filtered list in the UI (Black Ninja version)



Težavnost 300

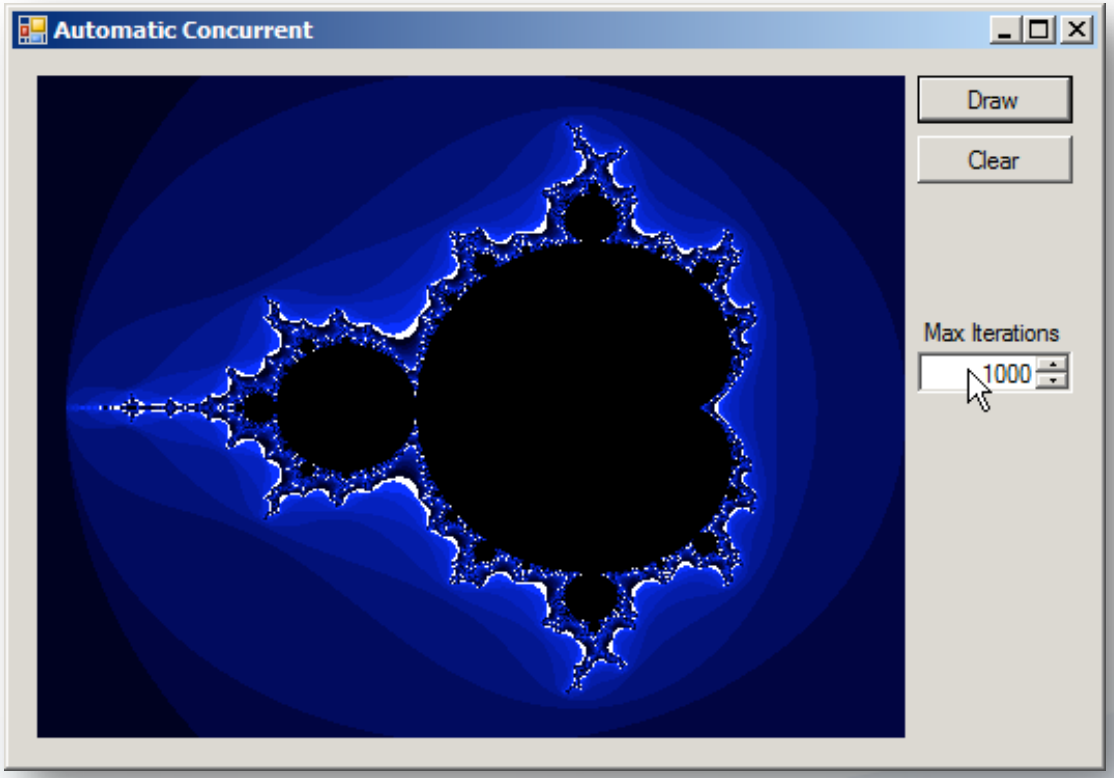
Microsoft NT konferenca 2009

Parallelization in place

- “Declarative Data Parallelism” is what PLINQ provides
- Changing the source of a LINQ query to IParallelEnumerable by calling AsParallel() on an IEnumerable takes care of concurrency automatically

Demo

- Drawing Mandelbrot Fractals



Težavnost 300

Microsoft NT konferenca 2009

Summary

- Frameworks like the .NET Parallel Extensions are great
- … but they don't do our structural work for us
- Functional approaches bring a useful discipline to the imperative C# language

Thank you

Please feel free to contact me about the content anytime.

oliver@sturmnet.org

Težavnost 300

Microsoft
NT konferenca 2009