**VisugXL event 20/11/2024**

**Session 1: Azure Static Web apps – Too good to be true! (Bart Wullems)**

https:/bartwullems.blogpost.com

20/07/1969 🡪 overal in the world, sitting in front of television because moonlanding

19/11/1969 🡪 niemand paid attention, third man sat foot on the moon, name not known, sad because as amazing as the first ones

08/10/2008 🡪 all streets empty, sitting in front of tv, Windows Azure was born at the pdc conference move tot he cloud

12/05/2021 🡪 niemand seems to care, Azure static web apps was announced

* Fanclub made : Peter Conrad and Static Web Apps fanclub
  + Demo webpage

Different ways to host static websites (without Azure static web apps)

* Server (Azure app service) 🡪 not cheap, not really for websites
* Blob Storage and apart API server for backend 🡪 very cheap
* Reverse Proxy (handles the two things of Blob storage (it’s what static webapps is)

Some examples of supported framworks

* Anger
* React
* Next
* …

Demo (most of today is free)

* Deploy fans site
* Framework used : Astro
* Different hosting plans ( **Free**/Standard/Dedicated)
* Deployment via Github Actions
* One of many ways to deploy a Static Web App
* Microsoft Orix

Configuration Management

* 2 kinds of configuration
  + Static content (file: staticwebapp.config.json)
    - Routing
    - Authentication
    - Authorization
    - ….
  + Backend part
    - Managed through Azure Portal, azure CLI or laC

Deployment

* You don’t need to do this

Environments

* Multiple supported environments:
  + Production (always supported free of payed)
  + Pull request (you can create a new environment for pull request)
  + Branch
  + You can make alsof very specific environments
* Gives lot of flexibility for your webapp

Security (very important)

* Very easy made by Azure Static web apps
* Built-in Authentication
  + Azure Active directory
  + Github credentials
  + You can also use a custom provider (can be anything)
* Authorization
  + 2 built in roles
    - Anonymous
    - Authenticated
  + Link users through
    - Invitations
    - Custom Azure Function

API Integration

* Managed functions
  + Free
  + Microsoft takes care of everithing for example
    - Integrated security
    - Seamless routing
* Manage my functions
  + Link it to another API integration if you don’t want to use the one from microsoft
* You cannot use them both at the same time

Database connections

SWA CLI

* Separate application
* It also simulates the Auzure Proxy
  + Talk to you backend
  + You can do the end to end

Enterprise-Grade Edge

Personal opinion

Interessant info but too much examples for the little time he had (1h), hij had in mijn gedachte de voorbeelden verminderd en extra info geven bij de verschillende delen.

**Session 2: A Season for Speed: Turning Puzzles into C# Performance Wins (**[**Michaël Hompus**](https://sessionize.com/api/v2/72593dpe/view/GridSmart)**)**

A season of Speed

How to turn puzzles into c# performance wins

Architect @ Info Support

Advent of code

* Created by Eric Wastl
* Older years are available to do
* Annual coding challenge held in december
  + Every day you get a unique input file
  + Puzzle composed of 2 parts
    - Part 2 unlocked when part 1 is correct
* Designed to improve programming skill through fun puzzles
* Open to programmers of all skill levels
* Worldwide use
* Adventofcode.com
* Top 100 (not big chance to make it tot he top 100)
* You can create private leaderbords to compete against collegues/friends

Ticks 🡪 1 tick = 100 nanoseconds

* Smallest time scale in .NET
* Not accurate enough if you want to messure

BenchemarkDotNet 🡪 helps you to transfrom methods (ask chatgpt)

It takes about 8min to run a benchemark

.NET 9 not really a lot of performance wins in vergelijking with .NET 8

Wel a lot of performance wins between .NET 8 and .NET 7

2020 – Day 1 – Part 1

Speed things up

* Linq
  + Going over the list twice 🡪 not fast enough
  + Using indexes 🡪 a little bit faster
* Foreach
  + Convert all the numbers at once
  + About the half faster than linq
* Foreach with range
  + It is about 10 times faster than without range
  + Memory wise 8 times bigger than foreach without range
* For
  + About the half faster than foreach
  + Memory wise 100 times smaller (under 1 kilobyte)
* Goto
  + Are they readeble? Not really good
  + 3 thousands nanoseconds faster than For
  + Same memory use
* Intermediate variables
  + Much faster for 2 reasons
    - Using the index ads a lot of codes
  + 300 nanosecons faster than Goto
* Initialize array
  + It is slower
  + 400 nanoseconds
  + In general it is faster but here not
  + Memory wise it is a little bit smaller
* ASCII (little info before next step)
  + A char in dotnet is stored as a ushort
  + By substracting 48, you get the integer equivalent
* Custom integer parsing
  + 25 procent faster than with initialize array
* Ushort
  + .NET is integer 🡪 every time you use a integer you need to convert to Ushort
  + Not really faster
  + Memory wise it is about 50% less
* Pointers
  + Performance = pointers?
  + It is not
  + 1.300 nanoseconds slower
* Array references
  + Not really readeble
  + Just as fast as pointers
  + The setup takes longer than just once
* Two pointers technique
  + 2 numbers (functioning as pointers)
  + 1.400 nanoseconds faster
* Hashing (hashset)
  + 650 nanoseconds faster than with 2 pointers
  + Memorywise it is too much
* BitArray
  + Only 622 nanoseconds
  + Little bit less memory almost not overhead

2020- day 1 – part 2

3 numbers 🡪 cannot do it with bits

* Two pointers just as fast than part 1
* The rest is a lot slower with just a little bit more code

Why faster than dotnet itself?

* Not general purpose
* No out-of-range checks needed
* Limited input set (ASCII)
* Tailored methods for specific input (unsigned, signed)

All kinds of stuff

Input parsing

* Regex vs String.split vs Span<T>
  + Regex slowest
  + String.Split a little bit faster but not much
  + Span<T> 🡪 the fastest

Test: Walk through int [512,512]

* For X, for y is slower than For y , for X
* Foreach is the fastest

Test: serialize char [512,512] to String

* Creating a new string is the fastest

More random tips & tricks

* Modulo 🡪 great for cycling through a list without concern for overflows
* Use the in argument to prevent copies of value types
* Goto is an easy way to jump out of multiple iterators at once
* Don’t initialize char arras with defaults; \0 is perfectly fine as comparison
* Use Range ^1 to get the last item of a String, or Array
* Initialize list types with the right capacity to prevent internal resizing
* If a char of a string uniquely identifies the whole string, use only the char

Don’t overthink it…

Useful algorithms to know

* Dijkstra’s
* A-star
* Manhatten Distance
* LCM
* GCD
* Gauss’s area formula
* Kargers algoritm for minimum cut

Personal opinion

Interessante inhoud over hoe je je code sneller kunt laten runnen aan de hand van een vb uit the advent calender, op het einde heeft hij ook een paar nuttige tips en tricks gegeven van hoe je je code sneller kan laten runnen.

**Session 3: Crafting modern CLI tools using .NET (Dante De Ruwe)**

Dotty

* Cli tool he build fort his talk
* Dotty greet VisugXl !
  + Responds with hello VisugXL!
* Dotty can also introduce itself
* Dotty convert
  + It can converts units
* Dotty weather Gent
  + Gives the wethear back fort his week in gent

Dantederuwe.com 🡪 some blogpost

Part 1 🡪 the cli

* Cli tools are used every day
* Why the cli?
  + Speed & efficiency
  + Automation & scripting
  + Resource-friendly
  + And so much more
    - Focus
    - Customizability
    - Acces via ssh
    - …
* CLI semantics

Part 2 🡪 building a CLI tool

* How would we do this using .NET 6
  + Mostly done by switch
* How would we do this using .NET 7 and higer
  + Switch with different cases under
  + Nicer way
  + You get the options already captured
  + Not the best experience
* Downsides
  + Manual detection of commands,arguments,options,…
  + Maual parsing / type validation
  + Manual defaults
  + Error prone
  + Build everything yourself
    - No build-in –help
    - Similar commands suggestion
    - …
  + No built-in IoC container
  + Does not look like .NET applications
* Generic host tot he rescue?
  + Familiar .NET skeleton
  + Dependecy injection

Nuget package fort his

* Cocona (used for dotty)
  + Familiar .NET skeleton
  + Dependecy injection
  + Better aproach than the switch commands
  + Can add option and default
  + Easily define commands and subcommands
  + Built-in basic option binding
  + Demo
    - Lotts of extra features
    - You can add descriptions, filters, … to your command
    - Can defined subcommands
      * You can add other commands to the group
* Spectre.Console
  + Is mainly a library that makes for pretty console output
  + Spectre.console.cli
  + Familiar .NET skeleton
  + No Dependecy injection
  + Easily define commands and subcommands
  + Built-in basic option binding

Part 3 testing

* Integration/e2e test your CLI
* How to easily test the CLI from the outside?
  + Testcontainers
    - Opensource library
    - Uses databases
    - Try all your logic
    - You can use it for testing and then delete it

Part 4 publish the CLI

* How?
  + You can not only share packages on nuget but also dotnet CLI Tools
    - Install using dotnet tool install
  + How to publish a cli tool to nuget
    - Dotnet pack (it takes al your files into one “pack”)
    - Dotnet nuget push
  + It’s live!

Part 5

The end

Dantederuwe.com/talks/cli-tools-dotnet