**VisugXL event**

Op woensdag 20 november heb ik 3 sessies bij het event VisugXL gevolgd. Hieronder vind u een verslag van de sessies dat ik gevolgd heb, alsook gaat u op het einde van elke sessieverslag mijn persoonlijke mening vinden over wat ik er van vond.

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

Tijdens de sessie werd uitgebreid ingegaan op de mogelijkheden en voordelen van Azure Static Web Apps. Deze service biedt een innovatieve oplossing om statische front-end frameworks eenvoudig te integreren met serverloze back-end API's via Azure Functions. Hieronder volgt een samenvatting van de belangrijkste onderwerpen, in samenhangende tekstvorm.

**Introductie**

De spreker begon de sessie met een aantal boeiende historische vergelijkingen. Hij herinnerde het publiek aan het moment op **20 juli 1969**, toen de hele wereld voor de televisie zat om de maanlanding te zien. Dit werd scherp gecontrasteerd met **19 november 1969**, toen de derde man voet op de maan zette, maar weinig aandacht kreeg, ondanks de indrukwekkende prestatie. Dezelfde lijn trok hij door naar de technologie: **8 oktober 2008** markeerde een mijlpaal met de introductie van Windows Azure op de PDC-conferentie, terwijl de aankondiging van Azure Static Web Apps op **12 mei 2021** nauwelijks aandacht kreeg, ondanks de innovatie die het met zich meebracht.

Om zijn punt kracht bij te zetten, richtte de spreker humoristisch een "fanclub" op voor Azure Static Web Apps, genaamd *Peter Conrad and Static Web Apps Fanclub*. Als onderdeel van de sessie werd een demo-webpagina gemaakt voor deze fictieve fanclub.

**Manieren om statische websites te hosten**

Vervolgens werden verschillende manieren besproken waarop statische websites gehost kunnen worden, waarbij Azure Static Web Apps als de meest geïntegreerde oplossing naar voren kwam. Traditionele servers, zoals Azure App Service, worden genoemd, maar blijken niet goedkoop en minder geschikt voor puur statische websites. Een alternatieve, voordeligere methode is het gebruik van Blob Storage met een aparte API-server voor de backend. Hoewel goedkoop, mist deze aanpak de integratie die Azure Static Web Apps wel biedt.

De spreker benadrukte het belang van een geïntegreerde oplossing en legde uit dat Azure Static Web Apps in feite werkt als een *reverse proxy*, wat de voordelen van beide bovenstaande methoden combineert.

**Ondersteunde frameworks**

Een ander groot voordeel van Azure Static Web Apps is de brede ondersteuning voor populaire front-end frameworks zoals Angular, React, Next.js en Astro. Voor de demo werd Astro gebruikt, een modern framework dat ideaal is voor het bouwen van statische sites. Deze flexibiliteit maakt de service aantrekkelijk voor ontwikkelaars met uiteenlopende voorkeuren.

**Demo en hostingplannen**

Tijdens de demo demonstreerde de spreker het proces van het deployen van een website met behulp van GitHub Actions. Dit toonde hoe eenvoudig het is om een Azure Static Web App te publiceren. De spreker lichtte ook de verschillende hostingplannen toe: een gratis plan dat ideaal is voor kleinere projecten, een standaardplan met meer mogelijkheden en een dedicated plan voor grootschalige toepassingen. Deze diversiteit maakt het platform toegankelijk voor zowel hobbyisten als bedrijven.

**Configuratiebeheer**

Configuratiebeheer binnen Azure Static Web Apps is overzichtelijk georganiseerd. Voor de statische content wordt een configuratiebestand (*staticwebapp.config.json*) gebruikt, waarin zaken zoals routing, authenticatie en autorisatie kunnen worden ingesteld. De backend-configuratie kan worden beheerd via de Azure Portal, Azure CLI of Infrastructure-as-Code (IaC). Deze scheiding maakt het beheer zowel flexibel als overzichtelijk.

**Omgevingen**

Een van de sterke punten van Azure Static Web Apps is de ondersteuning voor meerdere omgevingen. De productieomgeving is standaard beschikbaar, ongeacht het gekozen plan. Daarnaast kunnen ontwikkelaars specifieke omgevingen creëren voor bijvoorbeeld pull requests of individuele branches. Deze functionaliteit biedt enorme flexibiliteit en maakt het testen en implementeren van nieuwe functionaliteiten eenvoudig.

**Veiligheid**

Beveiliging is een cruciaal aspect van Azure Static Web Apps en is eenvoudig te beheren. Voor authenticatie zijn er standaardopties zoals Azure Active Directory en GitHub-inloggegevens, maar het platform ondersteunt ook aangepaste authenticatieproviders. Voor autorisatie zijn er twee ingebouwde rollen: *anonymous* en *authenticated*. Gebruikers kunnen gekoppeld worden via uitnodigingen of met behulp van aangepaste Azure Functions. Deze ingebouwde beveiligingsmaatregelen maken het platform geschikt voor zowel kleine projecten als grootschalige, zakelijke toepassingen.

**API-integratie**

Voor de integratie van backend-functionaliteit biedt Azure Static Web Apps twee opties. Ontwikkelaars kunnen kiezen voor door Microsoft beheerde functies, die gratis zijn en voorzien van geïntegreerde beveiliging en naadloze routing. Voor meer controle is het ook mogelijk om een eigen API te koppelen. De keuzevrijheid biedt flexibiliteit, hoewel het niet mogelijk is om beide methoden tegelijk te gebruiken.

**SWA CLI**

De sessie ging ook kort in op de Static Web Apps CLI, een tool die de Azure Proxy simuleert. Hiermee kunnen ontwikkelaars lokaal hun volledige webapplicatie testen, inclusief de interactie tussen front-end en back-end. Dit maakt de CLI een waardevolle toevoeging aan de ontwikkelworkflow.

**Persoonlijke mening**

De sessie bood interessante inzichten in Azure Static Web Apps en liet zien hoe krachtig en flexibel deze oplossing is. Wel vond ik dat de spreker te veel tijd besteedde aan voorbeelden, wat ten koste ging van verdieping in enkele onderwerpen. Een meer gebalanceerde aanpak had waarschijnlijk meer waarde opgeleverd binnen de beperkte tijd van één uur.

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

Deze sessie, gegeven door een architect van Info Support, richtte zich op het verbeteren van de snelheid van C#-code aan de hand van puzzels en uitdagingen uit de programmeerwedstrijd *Advent of Code*. De spreker gebruikte praktische voorbeelden om technieken en optimalisaties uit te leggen die de prestaties van C# aanzienlijk kunnen verbeteren. Hieronder volgt een samenvatting.

**Advent of Code**

De sessie begon met een introductie van *Advent of Code*, een jaarlijkse programmeerwedstrijd gecreëerd door Eric Wastl. Deze wedstrijd wordt elk jaar in december georganiseerd en biedt dagelijks een nieuwe programmeerpuzzel, bestaande uit twee delen. Zodra het eerste deel correct is opgelost, wordt het tweede deel ontgrendeld. De wedstrijd is toegankelijk voor programmeurs van alle niveaus en wordt wereldwijd gebruikt om programmeervaardigheden te verbeteren op een leuke en uitdagende manier.

De deelnemers krijgen unieke invoerbestanden voor elke puzzel en kunnen deelnemen aan openbare of privé-ranglijsten om te concurreren met collega’s en vrienden. Hoewel het uiterst moeilijk is om de top 100 te bereiken, blijft het doel van *Advent of Code* om plezier te hebben en te leren.

**Optimalisaties in C#: Voorbeelden en Technieken**

De spreker demonstreerde verschillende manieren om de prestaties van C#-code te verbeteren met voorbeelden uit de puzzel van dag 1 in 2020. Hieronder volgt een overzicht van de besproken technieken:

**1. Gebruik van verschillende iteratietechnieken**  
Bij het doorlopen van een lijst bleek dat sommige technieken efficiënter zijn dan andere:

* LINQ is handig, maar niet de snelste oplossing, omdat het een lijst tweemaal doorloopt.
* foreach is ongeveer tweemaal zo snel als LINQ.
* for-loops zijn nog sneller en gebruiken bovendien veel minder geheugen.
* Het gebruik van goto levert een minimale prestatieverbetering op, maar maakt de code minder leesbaar.

**2. Intermediate variabelen**  
Door tussenvariabelen te gebruiken, kun je de prestaties verbeteren. Dit komt doordat directe indexering vaak extra code genereert. Hierdoor wordt het werk sneller en eenvoudiger.

**3. Arrays en geheugenbeheer**  
Bij het initialiseren van arrays is het niet altijd nodig om standaardwaarden toe te kennen. Het gebruik van *BitArray* kan voordelig zijn vanwege de minimale geheugenkosten, maar sommige technieken, zoals pointers, bleken juist langzamer of onleesbaar.

**4. ASCII-waarden en aangepaste integer-parsing**  
De spreker legde uit dat een karakter in .NET wordt opgeslagen als een ushort. Door 48 af te trekken, kan dit eenvoudig worden omgezet naar de bijbehorende integer-waarde. Dit biedt aanzienlijke snelheidswinst bij het verwerken van grote hoeveelheden data.

**5. Hashing en de Two-Pointers techniek**  
Voor het vinden van getallenparen of -trio’s werden technieken zoals hashing (met een HashSet) en de Two-Pointers methode besproken. Hoewel hashing sneller is, gebruikt het veel geheugen. De Two-Pointers techniek biedt een goede balans tussen snelheid en geheugengebruik.

**Algemene prestatieverbeteringen in C#**

De sessie ging verder met bredere inzichten en tests voor het optimaliseren van C#-code. Enkele belangrijke lessen waren:

* Bij invoerparsing is Span<T> de snelste methode, vergeleken met bijvoorbeeld Regex of String.Split.
* Bij het doorlopen van een int[512,512]-array is het sneller om eerst de y-as en daarna de x-as te itereren, in plaats van andersom.
* Bij het serialiseren van een char[512,512] naar een string bleek het creëren van een nieuwe string de snelste aanpak.

Daarnaast gaf de spreker enkele algemene tips:

* Gebruik de *modulo*-operator om door een lijst te itereren zonder overflow.
* Het gebruik van de in-parameter voorkomt onnodige kopieën van waardetypen.
* goto kan nuttig zijn om uit meerdere geneste iteraties tegelijk te springen, maar dit gaat vaak ten koste van de leesbaarheid.
* Initialiseer lijsten met de juiste capaciteit om interne resizing te voorkomen.
* Als één karakter een string uniek identificeert, gebruik dan alleen dat karakter in plaats van de hele string.

**Nuttige algoritmen**

De spreker raadde enkele nuttige algoritmen aan om te leren voor het oplossen van complexe problemen:

* **Dijkstra’s algoritme** (kortste pad)
* *A-algoritme*\* (kortste pad met heuristieken)
* **Manhattan Distance**
* **Gauss’ oppervlakteformule**
* **LCM en GCD** (kleinste gemene veelvoud en grootste gemene deler)
* **Karger’s algoritme** (minimale snede in een grafiek)

**Persoonlijke mening**

De sessie bood veel interessante inzichten over het verbeteren van de prestaties van C#-code, geïllustreerd aan de hand van een voorbeeld uit *Advent of Code*. De afsluitende tips en tricks waren bijzonder nuttig en praktisch toepasbaar. Al met al was het een informatieve en inspirerende sessie, met een goede balans tussen theoretische uitleg en praktische toepassingen.

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

In deze sessie werd besproken hoe moderne Command Line Interface (CLI)-tools kunnen worden ontwikkeld met behulp van .NET. Hoewel console-applicaties soms als ouderwets worden beschouwd, benadrukte de spreker hun blijvende waarde voor interne tools, prototypes en zelfs grotere projecten. De sessie bood een overzicht van hoe CLI-tools efficiënter, gebruiksvriendelijker en krachtiger gemaakt kunnen worden met moderne technieken.

**Dotty: Een voorbeeld CLI-tool**

De spreker demonstreerde een zelfgebouwde CLI-tool genaamd *Dotty*. Deze tool werd ontwikkeld als voorbeeld voor de sessie en bevatte diverse functionaliteiten:

* **Dotty greet VisugXL!**: Geeft een begroeting terug zoals “Hello VisugXL!”
* **Dotty introduce**: Stelt zichzelf voor.
* **Dotty convert**: Kan eenheden converteren.
* **Dotty weather Gent**: Geeft het weer voor Gent weer.

Meer informatie en blogs over deze tool zijn te vinden op [dantederuwe.com](https://dantederuwe.com).

**Waarom een CLI-tool?**

CLI-tools worden dagelijks gebruikt door ontwikkelaars vanwege hun vele voordelen:

* **Snelheid en efficiëntie**: Handige hulpmiddelen voor snel gebruik zonder zware interfaces.
* **Automatisering en scripting**: Gemakkelijk te integreren in workflows.
* **Licht in gebruik**: CLI-tools zijn minder resource-intensief dan grafische toepassingen.
* **Andere voordelen**:
  + Focus en aanpasbaarheid.
  + Toegang via SSH.
  + Geschikt voor specifieke ontwikkelprocessen.

**Het bouwen van een CLI-tool in .NET**

De sessie ging dieper in op het bouwen van een CLI-tool met .NET en de uitdagingen die daarbij komen kijken.

**1. Werken met .NET 6**  
Bij het bouwen van een CLI-tool in .NET 6 worden vaak switch-statements gebruikt om commando's en argumenten te verwerken. Hoewel dit werkt, leidt het tot een minder intuïtieve ervaring en veel handmatige stappen, zoals:

* Detectie van commando’s, argumenten en opties.
* Handmatige parsing en type-validatie.
* Zelf foutmeldingen en standaardwaarden definiëren.

**2. Verbeteringen in .NET 7 en hoger**  
Met .NET 7 worden commando-opties vooraf verwerkt, wat de code leesbaarder maakt. Toch blijven veel nadelen bestaan, zoals de afwezigheid van ingebouwde functies voor hulp (--help), foutafhandeling en Dependency Injection (DI).

**3. Gebruik van een Generic Host**  
Een betere oplossing is het gebruik van een *Generic Host*, een bekend .NET-skelet dat functies zoals Dependency Injection mogelijk maakt. Dit verbetert de ontwikkelervaring aanzienlijk.

**NuGet-pakketten voor het bouwen van CLI-tools**

De spreker introduceerde twee NuGet-pakketten die het bouwen van CLI-tools met .NET vereenvoudigen:

**1. Cocona**  
Dit pakket, gebruikt voor de Dotty-tool, biedt een gestroomlijnde manier om CLI-tools te ontwikkelen. Het bevat:

* Dependency Injection.
* Eenvoudige definitie van commando’s en subcommando’s.
* Basisoptie-binding en standaardwaarden.
* Ingebouwde functies zoals beschrijvingen en filters.

**2. Spectre.Console**  
Dit pakket richt zich op het verbeteren van de visuele presentatie in de console. Daarnaast biedt het:

* Eenvoudige definitie van commando’s en subcommando’s.
* Basisoptie-binding.
* Geen ondersteuning voor Dependency Injection, maar wel een vergelijkbaar .NET-skelet.

**CLI-tools testen en publiceren**

**1. Testen van CLI-tools**  
De spreker besprak hoe je CLI-tools kunt testen, zowel via integratietests als end-to-end (E2E)-tests. Hierbij werd *Testcontainers* aanbevolen, een open-source bibliotheek die containers gebruikt om bijvoorbeeld databases te testen. Dit maakt het mogelijk om volledige logica te testen zonder sporen achter te laten.

**2. Publiceren van CLI-tools**  
Het publiceren van een CLI-tool kan via NuGet.

* Met dotnet pack worden alle bestanden samengevoegd in één pakket.
* Met dotnet nuget push wordt het pakket geüpload naar NuGet.
* De tool kan vervolgens worden geïnstalleerd met dotnet tool install.

**Afsluiting**

De sessie sloot af met een overzicht van de besproken onderwerpen en een verwijzing naar de website van de spreker ([dantederuwe.com/talks/cli-tools-dotnet](https://dantederuwe.com/talks/cli-tools-dotnet)) voor meer informatie.

**Persoonlijke mening**

Deze sessie bood een praktisch en duidelijk overzicht van hoe je moderne CLI-tools kunt bouwen met .NET. De uitleg over pakketten zoals Cocona en Spectre.Console gaf waardevolle inzichten, en de demonstratie van Dotty maakte het geheel tastbaar. Het stuk over testen en publiceren was een handige aanvulling, wat de sessie zeer informatief en bruikbaar maakte voor ontwikkelaars.

Ik vond dit een heel boeiende dag waar ik veel info heb bijgeleerd dat je op school niet snel zou horen, enige minpunt is dat het moeilijk volgen was terwijl dat ik notities moest nemen voor deze opdracht.