SDN MAGAZINE

SOFTWARE DEVELOPMENT NETWORK

ISSN: 2211-6486

IN DIT NUMMER O.A.:

Nummer 117 juni 2013 SDN Magazine verschijnt elk kwartaal en is een uitgave van Software Development Network

117

www.sdn.nl

Microsoft en Achmea: strategische partners bij de ontwikkeling van Visual Studio 2010

"Achmea wil razendsnel kunnen inspelen op ontwikkelingen in de markt"

Achmea is een waardevolle strategische partner van Microsoft. Dit blijkt onder meer uit het feit dat de verzekeraar heeft meegedacht bij de ontwikkeling van Microsoft Visual Studio 2010. "Microsoft beschouwt ons als een van de meest innovatieve verzekeraars ter wereld", zegt Achmea's principal architect Alex Thissen.

Foto: Hans Barten

Alex Thissen: "Microsoft luistert echt naar Achmea."

Afgelopen zomer was Achmea's Alex Thissen een week lang te gast bij het Windows Communication Foundation team op Microsofts hoofdkantoor in het Amerikaanse Redmond. "Er zijn op de hele wereld maar zes bedrijven die daarvoor werden uitgenodigd, dus ik beschouw het wel als een eer dat Achmea deel uitmaakt van dit selecte groepje. Microsoft luistert echt naar Achmea, ze beschouwen ons als een bedrijf dat voorop wil lopen. We spraken bijvoorbeeld over Microsofts ondersteuning van IBM Websphere MQ, waarvoor wij andere communicatiepatronen mogelijk wilden maken. Maar ook de koppeling tussen Sharepoint en SAP was onderwerp van gesprek."

Microsoft-topman Scott Guthrie, alias 'Scott Gu', was onlangs te gast op de Achmea-burelen. Thissen: "Ja, dat zijn inspirerende sessies. In mijn vakgebied heeft hij de status van een popster, dus je begrijpt dat het geweldig sparren is met zo'n man."

Mep je fit

Bij de ontwikkeling van Visual Studio Team System (bestaande uit ontwikkelomgeving Visual Studio 2010 en de 'Application Lifecycle Management omgeving' Team Foundation Server 2010) was Achmea in een vroeg stadium betrokken, vertelt Thissen. "We hebben ideeën en suggesties aangeleverd en Microsoft gewezen op bugs en verbeterpunten, zoals bijvoorbeeld bij het berekenen van de *burnrate* van code. Dat is nuttig voor ons én nuttig voor Microsoft."

Voor Achmea is Visual Studio Team System belangrijk bij het ontwikkelingen van software aan de voorkant, zoals webapplicaties en mobiele applicaties, en voor integratie-oplossingen met behulp van webservices. "Voor Achmea wordt het internet steeds belangrijker, niet alleen om schades te melden of verzekeringen af te sluiten, maar ook voor allerlei zaken op het gebied van preventie. Zo hebben we in opdracht van Zilveren Kruis Achmea de game 'Mep je fit' ontwikkeld voor Microsofts 'computertafel' Surface. Ook tijdens de Olympische Winterspelen hadden we in het Holland Heineken House een gezondheidsinstallatie staan; bezoekers konden daarop onder andere hun bloeddruk en body mass index meten en sportieve prestaties vergelijken met die van topsporters."

Agile

Daarnaast draagt de Microsoft-software bij aan de toenemende vraag naar transparantie en voorspelbaarheid, vindt Thissen. "Team Foundation Server houdt alle stappen in het ontwikkelproces bij, vastgelegd in hiërarchische workitems. Zo bieden wij onze klanten – de mensen in de business van Achmea – optimaal inzicht in het proces."

Last but not least is Thissen tevreden met MSF Agile, het processjabloon dat Visual Studio Team System voert. "Dat betekent dat we snel en flexibel kunnen bouwen, in korte iteraties met de klant. Dat is precies wat we nodig hebben, want Achmea wil razendsnel kunnen inspelen op nieuwe ontwikkelingen die zich voordoen in de markt."

achmea

Colofon

Uitgave:
Software Development Network
Eenentwintigste jaargang
No. 117 • juni 2013

Bestuur van SDN:
Remi Caron, voorzitter
Rob Suurland, penningmeester
Marcel Meijer, secretaris

Redactie:
Marcel Meijer
(redactie@sdn.nl)

Aan dit magazine werd meegewerkt door:
Roel Hans Bethlehem, Bob Swart, Maarten van Stam, Arjen Bos, Alexander Meijers, Remi Caron, Marcel Meijer en natuurlijk alle auteurs!

Listings:
Zie de website www.sdn.nl voor eventuele source files uit deze uitgave.

Contact:
Software Development Network
Postbus 506, 7100 AM Winterswijk
Tel. (085) 21 01 310
Fax (085) 21 01 311
E-mail: info@sdn.nl

Vormgeving en opmaak:
Reclamebureau Bij Dageraad, Winterswijk
www.bijdageraad.nl

Adverteerders

Adverteren?
Informatie over adverteren en de advertentietarieven kunt u vinden op www.sdn.nl onder de rubriek Magazine.

voorwoord

Beste SDN magazine lezer,

Het wil maar niet lukken met de lente. Het mooie weer wil maar niet komen. Maar dat geeft ons wel allemaal de mogelijkheid om heerlijk binnen bij de openhaard mooie nieuwe Mobile apps te maken of nieuwe technologieën uit te proberen. Als SDN magazine zijn wij op zoek naar jullie ervaringen met alle verschillende mobile platformen of nieuwe technologieën.

Ondanks het mindere weer is het wel warm met allerhande nieuwtjes. De laatste aankondigen van Windows Blue oftewel Windows 8.1, zodat we direct kunnen booten in de Desktop mode etc. Of de aankondigingen van XBox One, het multimedia device voor in de woonkamer. Ik vind het design wel erg lijken op de VCR recorder van Philips uit de jaren tachtig, maar dat mag de pret niet drukken. Helaas zullen we moeten wachten tot de E3 om de echte kwaliteit van de spellen te zien, maar dit beloofd wel een erg mooi platform te worden. Super, ook dit uitgebreide integratie met de verbeterde Kinect. De telefoon was al personal, mijn Surface is straks personal en mijn XBox straks gelukkig ook. Overigens biedt de XBox One straks meer mogelijkheden dan Televisie, Films en spellen. Ik denk dat het een heel breed toepassingsgebied heeft, misschien zelfs wel meer naar de zakelijk kant. Over bovenstaande nieuwtjes komen we in de volgende magazines of SDN events vast en zeker terug.

Als je dan bij de openhaard zit en niet aan het programmeren bent, dan is dit magazine voldoende gevuld voor een paar leerzame uurtjes. Met artikelen over Blocked Threads in Delphi, Knockout, Push communication, hoe te beginnen met Windows Azure, column over Agile, Mobile strings in Delphi, Active directory in de Cloud, een oplossing om je ASP.NET gridview scrollable te maken en een goed verhaal over SCRUM.

Uiteraard hopen wij, dat je dit magazine heerlijk in het zonnetje in je tuin, op je balkon of een gezellig terras kunt lezen. Maar hoe dan ook we wensen je een fijne zonnige zomer en heel veel leesplezier!

Groeten,
Marcel •

Agenda 2013

- SDE 2
  14 juni 2013
- TechEd Europe
  25-28 juni 2013
  Madrid
- Build Conference
  26 - 28 juni 2013
- SDE 3
  20 september 2013
- SDE+
  medio november 2013

Inhoud

Cary Jensen

Terminating a Blocked Thread

A good rule to live by, if you are a software developer, is do not add additional threads of execution to an application unless there are no acceptable alternatives. What this implies is that there are situations where it is necessary to add one or more additional threads of execution to an application. These additional threads of execution are often called worker threads, and that is the term that I am using in this article.

A common use for a worker thread is to wait until something happens. This something may be an operating system notification, a timeout on a synchronization object, the termination of another thread, or some other similar type of signal. While the thread is waiting for this something to happen, it is blocked.

Blocked Threads

Blocked threads pose a specific problem, and that is that they do not execute. To put this another way, while a thread is blocked it is dormant. In order to better understand this issue, let's consider how the TThread class does its work as well as how it terminates.

Each thread has a virtual method named Execute, and any code that appears in the overridden Execute method is performed in a worker thread. This work proceeds concurrently, and independently, of work being performed by your application's main thread of execution, as well as that performed by any other worker threads.

What is critical, as far as this article is concerned, is how a thread terminates. Simply put, a thread terminates when it exits its Execute method. For example, a thread may perform a specific task in its Execute method, after which it exits the Execute method. At this point, the thread has terminated.

This situation is fine when a thread has a short, specific task that it performs to completion, after which it terminates. In many cases, however, a thread doesn't just perform one task and then terminate. Instead, it performs a task over and over, until it is instructed to terminate. This instruction comes in the form of a call to its Terminate method, which sets the thread's Terminated property to True. Most often, it is the main thread of execution of your application that calls the thread's Terminate method.

What is often confusing is that calling a thread's Terminate method does not actually terminate a thread, since a thread only terminates when its Execute method is exited. As a result, for those threads that perform a task repeatedly, it is the responsibility of the thread to periodically test its Terminated property. When it discovers that its Terminated property is set to True the thread should respond by releasing any resources it has used and exit its Execute method as quickly and efficiently as possible.

This brings us back to the basic problem. A thread must execute code to test its Terminated property and respond by exiting its Execute method. However, if that thread is blocked, waiting for something to happen, it is not executing at all, which means that it cannot test its Terminated property.

Terminating Blocked Threads

At first glance, terminating a blocked thread sounds like an impossible task. Fortunately, there are solutions that are both sound and effective.

Let's begin by considering a task that is well suited for a worker thread. Imagine that you need to write a service that processes files created by another process. Each time that other process creates its output file, your service needs to read and parse that file's contents, adding the contained data into a relational database.

What your service can do is spawn a thread that blocks on a FILE_NOTIFY_CHANGE_FILE_NAME notification, which is a Windows operating system event. To do this the thread can create a the notification by calling the Windows API function FindFirstChangeNotification, passing a parameter indicating into which directory the external process will write the file. This function returns a handle to the notification, which you then block on using one of a handful of blocking functions, such as WaitForSingleObject.

Calling WaitForSingleObject blocks the thread, which will remain suspended until a file change occurs or the call times out. When a file change does occur in that directory, which is what will happen when a text file is written to that directory, the thread will become unblocked, at which time the thread can verify that the file change is associated with the creation of a text file that it must process. Either before or after the thread examines the file it might have to process it should also test its Terminated property, just in case it had been asked to terminate some time before the file was created. So long as the thread's Terminate method has not been called, the thread should loop back and call the WaitForSingleObject function again.

Unresponsive Threads

Unfortunately, there is a drawback to blocking a thread in this way. A blocked thread does not execute. As a result, it cannot terminate while it is being blocked. In other words, if the main thread of execution calls the thread's Terminate method, that thread will not even be able to test its Terminated property until a new file is created, or the call to WaitForSingleObject times out. The situation might seem even worse when you consider that in many cases you will pass a timeout parameter to WaitForSingleObject using the global variable INFINITE, in which case the thread will never time out.

So, the question is, how do you configure a thread that will be blocked for long periods of time to be responsive to a request to terminate. Take our text file processing application, for example. In most cases the text processing worker thread will be unblocked only when a change occurs in the directory of interest. But what if you need to terminate the process. If the process cannot terminate until the thread terminates, it might be hours before the thread is unblocked, at which time it can evaluate Terminated and exit.

Fortunately, there is a technique that permits you to create threads that are blocked for extended periods of time but can still be terminated on demand, such as when the application needs to shutdown.

The code for this example can be found in the ResponsiveThread project, which you can download using this URL.

http://www.jensendatasystems.com/responsivethreads.zip

This project includes a thread that processes text files whenever they appear in the same directory as the executable. To simulate a real process, the text files are read into a ClientDataSet. The ClientDataSet stores the contents of the file by name. If a new text file appears whose name already appears in the ClientDataSet, a blob field in the ClientDataSet is updated with the new file contents. If the filename does not already appear in the ClientDataSet, a new record is inserted, using that filename, and the contents of that file are stored in the associated blob field. Once the file's contents have been stored, the original file is deleted from the folder.
The code that performs this operation is found in the LoadTextFile procedure shown here.

```
procedure LoadTextFile(Filename: String);
begin
  CDS.Open;
  try
    CDS.IndexFieldNames := 'Filename';
    if not CDS.FindKey([UpperCase(FileName)]) then
    begin
      CDS.Insert;
      CDS.Fields[0].AsString := UpperCase(FileName);
      TBlobField(CDS.Fields[1]).LoadFromFile(FileName);
      CDS.Post;
    end
    else
    begin
      CDS.Edit;
      TBlobField(CDS.Fields[1]).LoadFromFile(FileName);
      CDS.Post;
    end;
  finally
    CDS.Close;
  end;
  DeleteFile(FileName)
end;
```

The worker thread that processes the file begins by setting up a notification associated with its executable's directory. This is shown here:

```
var
  HDirChange: THandle;
…
begin
  HDirChange := FindFirstChangeNotification(
                PChar(ExtractFilePath(ParamStr(0))),
                False, FILE_NOTIFY_CHANGE_FILE_NAME);
```

In a non-responsive application, the code that blocks on the notification might look something like this:

```
while WaitForSingleObject(HDirChange, INFINITE) do
  //check for appropriate text files
  if Terminated then exit;
  //do some work
  //...
  //loop back again
end;
```

The problem with this code is that a test for Terminated is only performed when a directory change occurs, which may be many hours apart (or may never happen at all).
One solution, but one with its own drawbacks, is to set a timeout. In the following example, the thread times out every minute (60000 milliseconds).

```
while True do
  WaitResult := WaitForSingleObject(HDirChange, 60000);
    if WaitResult := WAIT_TIMEOUT then
      if Terminated then exit
    else
    begin
      //check for appropriate text files
    end;
    //loop back again
end;
```

The drawback to this approach is two-fold. First, the thread will unblock every minute, whether or not a change has occurred in the directory. Second, the thread may be blocked for up to a minute even after the application has tried to terminate the thread.

Creating Responsive Blocked Threads

All of these problems can go away when you ask the thread to block on more than one event. One of these events is the event of interest, a change notification in this case. At least one more event is one that is controlled by the primary thread of execution in your application. When the application wants to shutdown, it uses that event to unblock the thread.

In the ResponsiveThread application the primary thread of execution manages a TEvent object. This TEvent is created as an unsignaled event from the OnCreate event handler of the main form. This event handler, which includes additional code to create the initial Client DataSet if it does not already exist, as well as the code to create the file processing worker thread, is shown here:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  AppDir := ExtractFilePath(ParamStr(0));
  CDS := TClientDataSet.Create(nil);
```

```
  CDS.FileName := AppDir + 'HoldText.cds';
  if not FileExists(CDS.FileName) then
  begin
    CDS.FieldDefs.Add('Filename', ftString, 127, True);
    CDS.FieldDefs.Add('File', ftBlob);
    CDS.CreateDataSet;
    CDS.SaveToFile;
    CDS.Close;
  end;
  Event := TEvent.Create(nil, False, False, '');
  TextFileProcessor := TTextFileProcessor.Create(True);
  TextFileProcessor.Start;
end;
```

In this code, Event is of the type TEvent, and it is declared in the main form's interface section, as shown here:

```
var
  Form1: TForm1;
  AppDir: String;
  CDS: TClientDataSet;
  Event: TEvent;
```

TextFileProcessor is a variable used to refer to the instance of TTextFileProcessor, which is a TThread descendant. That variable declaration is shown here:

```
var
  TextFileProcessor: TTextFileProcessor;
```

The following is the entire TTextFileProcessor declaration and implementation, minus some comment lines.

```
unit TextFileProcessu;

interface

uses
  Classes, mainformu, Windows, SysUtils;

type
  TTextFileProcessor = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
  end;

implementation

procedure TTextFileProcessor.Execute;
var
  HDirChange: THandle;
  WaitResult: Cardinal;
  sr: TSearchRec;
  ObjHandles : Array[0..1] of THandle;
begin
  HDirChange := FindFirstChangeNotification(
    PChar(AppDir),
    False, FILE_NOTIFY_CHANGE_FILE_NAME);
  ObjHandles[0] := HDirChange;
  ObjHandles[1] := Event.Handle;
  while True do
  begin
     WaitResult := WaitForMultipleObjects(2, @ObjHandles,
False, INFINITE);
    if WaitResult = 0 then  //the file notification was de-
tected. Process files
    begin
      if SysUtils.FindFirst('*.txt', faAnyFile, sr) = 0 then
      begin
        repeat
          Self.Synchronize(procedure
                      begin
                        LoadTextFile(AppDir + sr.Name);
                      end);
        until FindNext(sr) <> 0;
        FindClose(sr);
      end;
    end;
    if Terminated then exit;
    FindNextChangeNotification(HDirChange);
  end;
end;

end.
```

When the worker thread's Execute method is entered, the notification is created and the handle of the notification along with the handle of the TEvent are assigned to an array of handles. Next, the code enters an infinite loop and begins blocking on the events in the array using WaitForMultipleObjects.

The first parameter of WaitForMultipleObjects is the number of events that are being waited for. The second is the address of the handle array, and the third is whether one or all of the events must be signaled before the thread will no longer be blocked. In this case, False is passed so that the signaling of any one of the events will unblock the thread. The final parameter is the timeout, which is set to INFINITE, meaning that it will never timeout.

WaitForMultipleObjects returns a value. If you pass False in the third parameter, that value is an integer associated with the event that caused WaitForMultipleObjects to return. If the first event in the array triggered WaitForMultipleObjects, 0 is returned. If the second event is triggered, 1 is returned, and so forth.

At this stage, only two things will unblock the event. The first is the file change notification, which triggers the thread to look for text files, processing each one found before looping back to the call to WaitForMultipleObjects. (An anonymous method was used to call the file processing code in this example.) The second event is the signaling of the TEvent. This occurs when the application is being shutdown from the OnClose event handler of the main form, as shown here.

```
procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  TextFileProcessor.Terminate; //Set Terminated to True
  Event.SetEvent; //Trigger the event. This unblocks the
thread
  TextFileProcessor.WaitFor; //If the thread is processing
files, wait
  TextFileProcessor.Free;
  //Should be all done.
  CDS.Free;
end;
```

The result is an application that has very little overhead, only processing files when there is a possibility of files being present that need to be processed. In addition, the worker thread terminates almost

instantly upon application shutdown.

Creating responsive blocked threads is even simpler if your application is a polling application, which is something that you might need if there is no available notification. A polling application is one that performs some task periodically. For example, imagine that you need to read data from a database every five minutes, taking some sort of action when a particular pattern of data is detected. In this cases you will block the thread with a determinate timeout. Each time the timeout expires the thread performs its task, and then loops back to the blocking call (after also checking to see if it has been asked to terminate).
I didn't create a project that demonstrates this approach, but I offer this pseudo code which demonstrates it.

```
procedure TMonitorThread.Execute;
var
  WaitResult: TWaitResult;
begin
  //Timeout after a minute
  while True do
  begin
    WaitResult := Event.WaitFor(60000);  // Five minutes
    if WaitResult = wrSignaled then  //The Event was signaled, which only
    begin                            //happens after the thread had been
                                     //instructed to terminate
      if Self.Terminated then exit;  //Yes, this is redundant.
    end;
    // Place code here that executes each time the TEvent times out
    //
    // Test the Terminated property before looping back to the Wait call
    if Self.Terminated then exit;
    //ok, go back and wait again
  end;
end;
```

As in the previous example, the Event variable is a TEvent created and managed by the main thread of execution. When the application is being shut down, the main thread signals the TEvent, which unblocks the thread, causing it to terminate almost immediately. •

Cary Jensen

Cary Jensen is the bestselling author of more than 20 books on software development, including Delphi in Depth: ClientDataSets, and winner of the 2002 and 2003 Delphi Informant Reader's Choice Award for Best Training. A frequent speaker at conferences, workshops, and seminars throughout much of the world, he is widely regarded for his self-effacing humor and practical approaches to complex issues. Cary's company Web site is at: http://www.JensenDataSystems.com. Cary has a Ph.D. from Rice University in Human Factors Psychology, specializing in human-computer interaction.

with
93000 employees
working for
2500 clients
in
94 countries
and
52 years of experience

software development holds no mysteries for us

CSC

BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING

CSC delivers top-quality software to its clients. We are the world's leading independent information technology services company. Our experience across all industries enables us to harness the best ideas, practices and solutions from both the public and private sectors. We do amazing things. Find out more at csc.com.

Marco Kuiper

Knockout: JavaScript applicaties met een punch

Er is niet meer om heen te draaien: webapplicaties worden steeds belangrijker. Met de (op)komst van HTML5 en de mogelijkheden die dit ons biedt, wordt het voor ontwikkelaars ook steeds interessanter om dit platform verder te ontdekken. Hoewel de taal die gebruikt wordt voor deze applicaties (JavaScript) zeer krachtig is, kan je snel het gevoel bekruipen dat er zeer veel handmatig dient te worden geprogrammeerd. Knockout kan je hierin helpen, om zo sneller applicaties te ontwikkelen.

Knockout (ook wel KnockoutJS of knockout.js) is een van de vele JS libraries die op dit moment in ontwikkeling zijn. Wat Knockout speciaal maakt, is dat het sterk leunt op het MVVM-pattern waar de Microsoftgemeenschap erg bekend mee is (Silverlight & WPF), daar waar andere libraries MVC gebruiken. Door het gebruik van declaratieve bindings wordt het viewmodel makkelijk en duidelijk aan de UI gekoppeld.

De eerste versie van de library, geschreven door Microsoftontwikkelaar Steve Sanderson zag in juli 2010 het daglicht en draagt op moment van schrijven versienummer 2.2.1 (uitgebracht in januari 2013).

Waarom Knockout?

Veel Microsoftontwikkelaars zullen de stap naar Knockout als framework snel maken vanwege de Microsoftachtergrond van het project. In de ASP.NET & Web Tools 2012.2 update is IntelliSense ook toegevoegd voor Knockout in Visual Studio 2012. Ontwikkelaar Steve Sanderson geeft echter wel aan dat dit een persoonlijk open-source product is, en dus niet onder de vlag van Microsoft valt.

Knockout werkt als stand-alone framework, en is daarom dus niet afhankelijk van een andere JavaScript library. Echter, andere libraries zoals jQuery kunnen prima samenwerken met Knockout. Zo kunnen ontwikkelaars gebruik maken van de kracht van MVVM en declaratieve bindings (Knockout), maar tegelijkertijd werken met goede AJAX functionaliteit en animaties (jQuery).

Dit artikel is een introductie in de wereld van Knockout, en maakt onderweg enkele uitstapjes en vergelijkingen met de wereld van C# (Silverlight & WPF). Enkele key concepts worden toegelicht, waar men op moet letten als ontwikkelaar en een aantal geavanceerde features.

Hello Knockout

De architectuur van een typische Knockout-applicatie (Figuur 1) is een goed begin om kennis te maken met het framework.
Voor ontwikkelaars met een Silverlight/WPF-achtergrond, zal deze architectuur niet vernieuwend overkomen. De client maakt gebruik van een MVVM pattern, en haalt data op van (verschillende) web services aan de kant van de server. In plaats van SOAP/XML communicatie, levert de server JSON data terug die de client meteen kan verwerken.

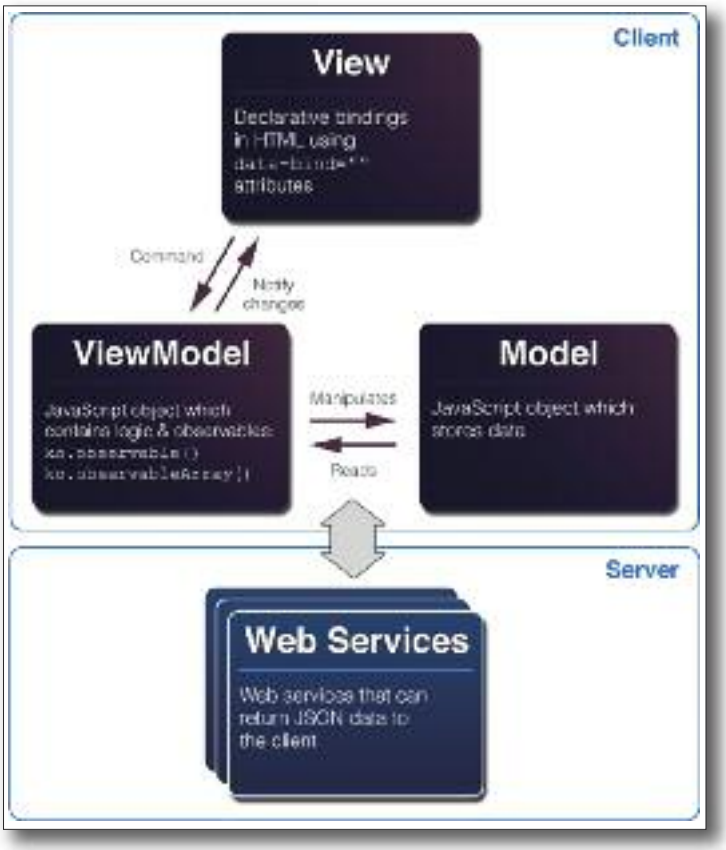


Fig. 1: Knockout-applicatie architectuur

In combinatie met REST kan dit een erg krachtig mechanisme worden.

Echter, de invulling van de client is vanzelfsprekend verschillend wanneer het aankomt op het gebruik van Knockout. Het ViewModel speelt hierin een erg grote rol, aangezien deze verantwoordelijk is voor de koppeling tussen het Model en de View.

Observables & data-binding

Het belangrijkste element van Knockout is het gebruik van Observables en de data-bind in de HTML. Een Observable is te vergelijken

met een Property op een ViewModel in C#, die een OnProperty Changed-event af kan laten gaan om de View aan te passen (Listing 1). In deze code is meteen te zien dat de Knockout-library wordt gebruikt om een Observable aan te maken: ko is de namespace voor Knockout.

```
//C#
private string _cookbookName = "Mijn kookboek";

public string CookbookName
{
  get { return _cookbookName; }
  set
  {
    _cookbookName = value;
    OnPropertyChanged("CookbookName");
  }
}

//JavaScript
this.cookbookName = ko.observable("Mijn kookboek");
```

Listing 1: Properties op het ViewModel in C# en JavaScript

De observable kan ook zonder variabele aangemaakt worden, maar door hem direct mee te geven wordt de initiële waarde van de observable aangegeven.

Vervolgens dienen de properties gekoppeld te worden aan de View. Hiervoor maakt Knockout handig gebruik van het HTML5 data-* attribuut. Dit attribuut stelt de ontwikkelaar in staat om custom attributen te maken, om daar extra data in op te slaan. Deze mag iedere naam hebben, zolang hij met data-* begint. Aangezien het hier om bindings gaat, werkt Knockout met een attribuut genaamd data-bind (Listing 2).

```
<!-- XAML -->
<TextBlock Text="{Binding CookbookName, Mode=TwoWay}" />

<!-- HTML -->
<p data-bind="text: CookbookName"></p>
```

Listing 2: Data binding in XAML en HTML

Wanneer de property op het ViewModel nu gemodificeerd wordt, zal de HTML automatisch aangepast worden. Het lezen en schrijven van deze properties dient middels een functie te gebeuren, aangezien niet alle browsers het gebruik van JavaScript properties ondersteunen (Listing 3).

Door gebruik te maken van de .subscribe() functie op een observable, kan de ontwikkelaar zich expliciet abonneren op wijzigingen die doorgevoerd worden op de observable (Listing 3). Wanneer de observable aangepast wordt, zal deze functie afgaan en de nieuwe waarde als parameter meegeven. Intern maakt Knockout ook gebruik van deze mogelijkheid om afhankelijkheden op de hoogte te brengen wanneer observables zijn gewijzigd.

Net zoals in C# kent Knockout een type lijst waarvan de state wordt bijgehouden. Over deze lijst kan geïtereerd worden om een UI op te bouwen. In C# is dit de ObservableCollection<T>, en bij Knockout de observableArray(). Wat belangrijk is om te weten, is dat een observableArray bijhoudt welke objecten in de lijst staan, niet de state van die objecten. Op een property van het type ko.observableArray kan je vervolgens verschillende functies aanroepen om de inhoud te muteren. Denk hierbij aan het toevoegen, verwijderen en sorteren van de objecten (Listing 3).

Het is belangrijk om aanpassingen op de observables door te voeren via de functies die daarvoor bedoeld zijn. Denk hierbij aan de get* en set* methoden die onder water door een C# compiler gecreëerd worden bij object-properties, en bij Java expliciet uitgeschreven dienen te worden. Overschrijf de observable dus niet door de property opnieuw te declareren, en voeg items aan de observableArray toe en niet aan de (onderliggende) JavaScript array. Wanneer dit gebeurt, zal het observable-mechanisme van Knockout niet afgaan en zal de UI niet aangepast worden (Listing 3).

```
  this.cookbookName(); // Read property
  this.cookbookName("Jouw kookboek"); // Write property
  this.cookbookName = "Jouw kookboek"; // Wrong: Will remove
the observable
  this.cookbookName.subscribe(function(newValue) {
    // Explicit subscribe to an observable
  });

  this.recipes = ko.observableArray() // Empty array
  this.recipes(); // Returns JavaScript array
  this.recipes().push("Book"); // Wrong: Will add to JS
array, but not update the UI
  this.recipes; // Knockout array to e.g. .push() objects
```

Listing 3: Gebruik van properties op het ViewModel

JavaScript ondersteunt wel getters en setters, wat dus betekent dat observables geen functies zouden hoeven zijn. Echter, vanwege de cross-browser ondersteuning die Knockout wil bieden, en het niet ondersteunen van getters en setters in oudere versies van IE, is toch gekozen om observables functies te laten zijn.

Hoe zien we deze observableArray() nu terug in de HTML? Middels de text binding wordt de tekstuele representatie nu in de HTML getoond (vergelijkbaar met de toString() methode). Dit is echter niet altijd wenselijk en om deze reden zijn er vele verschillende soorten bindings gedefinieerd.

Bindings

Bindings zorgen voor de koppeling tussen de HTML (de View) en het onderliggende ViewModel, en zijn daarom cruciaal voor het werken van de applicatie. De koppeling tussen de View en het ViewModel wordt middels de ko.applyBindings() methode gemaakt (Figuur 2).

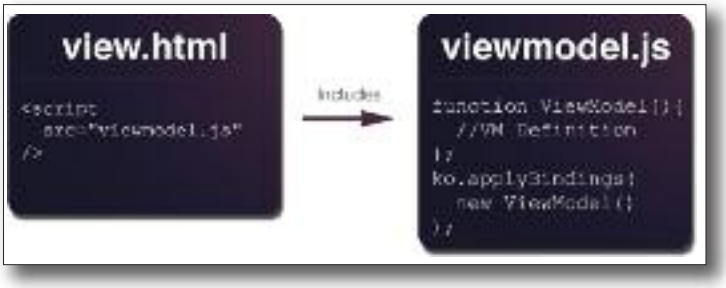


Fig. 2: Activatie van Knockout bindings

Op deze manier wordt de context van het ViewModel uit viewmodel.js gebruikt over de gehele pagina van de view.html. Het is echter ook

mogelijk om meerdere ViewModels op een pagina te gebruiken, mocht dit nodig zijn. Dit kan door een tweede parameter op te geven in de ko.applyBindings() methode, die het HTML element mee geeft waar het ViewModel aan gekoppeld wordt (Listing 4).

```
<!-- HTML -->
<div id="cookbook">
  <p data-bind="text: title"></p>
</div>
<div id="novel">
  <p data-bind="text: title"></p>
</div>

//JavaScript
function CookbookViewModel() {
  this.title = ko.observable("Mijn kookboek");
}

function NovelViewModel() {
  this.title = ko.observable("Mijn roman");
}

ko.applyBindings( new CookbookViewModel(),
  document.getElementById("cookbook") );
ko.applyBindings( new NovelViewModel (),
  document.getElementById("novel") );
```

Listing 4: Gebruik van ko.applyBindings() met meerdere ViewModels

Nu de View en het ViewModel aan elkaar gekoppeld zijn, kunnen de verschillende bindings worden gebruikt om de diverse scenario's waar je als ontwikkelaar tegen aan kunt lopen te realiseren. Deze bindings zijn te verdelen in verschillende categorieën, met voor elke binding een eigen functie.

Binding	Beschrijving
Text and appearance	
visible	Het wel of niet weergeven van het DOM element waar deze binding op geplaatst is. Dit kan gebind worden aan een boolean op het ViewModel, maar er kan ook een expressie aan worden meegegeven.
text	De text-binding zorgt ervoor dat er een textnode wordt aangemaakt binnen het DOM element van de binding, en daar de tekst in plaatst van de property op het ViewModel.
css	De css-binding dient te worden gebruikt om een CSS klasse wel of niet toe te voegen aan het DOM element van de binding. De klasse kan bepaald worden door een observable binnen het ViewModel, maar ook rechtstreeks in de HTML samen met een expressie. In het laatste geval zal de binding er als volgt uit zien: data-bind="css: { outOfStock : amount() < 0 }".
attr	Om de attributen op het element te kunnen modificeren, dient gebruik te worden gemaakt van de attr-binding. Op deze manier krijgt de ontwikkelaar de controle over de attributen zoals href, alt, src etc.
Control flow	
foreach	Om over een observableArray te itereren in de DOM, kan gebruik gemaakt worden van de foreach-binding. De binding context binnen de loop verwijst naar het element waar op dat moment over geïtereerd wordt (Listing 5).
if	Hoewel de if-binding erg lijkt op de visible-binding (beide laten DOM elementen wel of niet zien), is er een belangrijk verschil tussen beide. De visible-binding plaatst alleen de CSS property display:none op het element, terwijl de if-binding de inhoud volledig uit de DOM haalt. De ifnot-binding werkt, zoals de naam al doet vermoeden, tegenovergesteld ten opzichte van de if-binding.
with	De with-binding zorgt voor een nieuwe binding context: alles binnen het element kan vervolgens direct aangesproken worden. Kortom, in plaats van Parent.Child en Parent.Child2 aan te geven in de bindings, kan men middels de with:Parent simpelweg Child en Child2 gebruiken.
Form fields	
click	Het JavaScript click-event wordt middels de click-binding afgehandeld. Het onthoudelaars hiermee om moeten gaan, en welke events nog meer worden ondersteund staat verderop beschreven.
options	De options binding werkt alleen samen met een <select> element, en zorgt ervoor dat een observableArray direct gekoppeld kan worden aan de dropdown list.
enable	Om ontwikkelaars de mogelijkheid te geven om form-elementen in en uit te schakelen, kan gebruik worden gemaakt van de enable binding. Hier dient een boolean-value aan te worden gekoppeld die aangeeft of het element gebruikt kan worden of niet.
value	De value-binding koppelt een (geselecteerd) form-element aan een observable. Dit verschilt per element: voor een <select> element wordt deze binding gekoppeld aan de geselecteerde waarde, maar bij een <input> element aan de waarde van de tekst.

Tabel 1: Een aantal veel gebruikte Knockout bindings

Door de technieken van de voorgaande hoofdstukken met elkaar te combineren, kan zeer snel een kleine, maar flexibele webpagina gemaakt worden (Listing 5).

```
// Pseudo JavaScript (Properties should be observables)
function ViewModel() {
  this.owner = {
    firstname: "", lastname: ""
  };

  this.books = [
    {
      title: "", summary: "",
      rating:
      {
        description: "", grade: 8
      }
    } // More books
  ];
}

<!-- HTML -->
<div data-bind="visible: books.length > 3">
  <p data-bind="with: owner">
    Owner:
    <strong data-bind="firstname"></strong>
    <strong data-bind="lastname"></strong>
  </p>
  <div data-bind="foreach: books">
    <h2 data-bind="text: title"></h2>
    <div data-bind="if: rating">
      <img data-bind="attr: { src: 'ratings/'+
rating.grade +'.jpg' }" />
      <p data-bind="rating.description"></p>
    </div>
    <p data-bind="text: summary"></p>
  </div>
</div>
```

Listing 5: Gebruik van verschillende bindings

De basiselementen van Knockout zijn nu behandeld, en een zeer simpele single-page applicatie kan met de voorgaande technieken gemaakt worden. Echter, Knockout biedt ook een meer geavanceerde functionaliteit aan, die nodig kan zijn voor de applicatie.

Computed observables

Computed observables zijn onderdeel hiervan. Dit type observable (eerder "dependent observables" genoemd) stelt ontwikkelaars in staat om een observable aan te maken dat wordt opgebouwd uit data van verschillende andere observables. Wanneer één van deze observables wordt aangepast, zal de computed observable zijn nieuwe waarde berekenen, en aanpassen (Listing 6).

```
// JavaScript
this.title = ko.observable("Mijn kookboek");
this.isbn = ko.observable(1782161147);

this.libraryIdentifier = ko.computed(function() {
  return this.isbn() + " (" + this.title() + ")";
}, this);
```

Listing 6: Simpel computed observable voorbeeld

Om het syteem van een computed observable te laten werken, maakt Knockout gebruik van een simpele maar doeltreffende techniek.

1. Wanneer een computed observable is aangemaakt, zal Knockout de functie direct uitvoeren om te controleren wat de initiële waarde is van de observable.
2. Tijdens het uitvoeren van die functie zal Knockout een lijst bijhouden welke (computed) observables op het viewmodel worden aangeroepen. Knockout zal nu automatisch de computed observable aanpassen wanneer één of meerdere van deze observables wordt aangepast.

Wanneer de computed observable een tak bevat die bij de initiële uitvoering van de functie niet wordt geraakt (denk aan een if/else structuur), dan kan dit later voor problemen zorgen. Knockout heeft immers niet alle observables in zijn interne lijst staan van observables die bijgehouden moeten worden voor de computed observable.

Een computed observable kan meer dan alleen maar simpelweg een waarde teruggeven op basis van de informatie uit andere observables. Een computed observable kan ook een waarde van de View ontvangen, en hierop actie ondernemen (Listing 7).

```
// JavaScript
this.libraryIdentifier = ko.computed({
  read: function () {
    return this.isbn() + " (" + this.title() + ")";
  },
  write: function (value) {
    // Needs validation, but just as an example
    this.isbn(value.substring(0, 10));
    this.title(value.substring(11, value.length));
  },
  owner: this
});
```

Listing 7: Computed observable met read en write functionaliteit

Deze functionaliteit is erg krachtig, en biedt een uitstekend alternatief voor het missen van de getters en setters die wel in C# aanwezig zijn. De gewone observable kan immers afgeschermd worden door een computed observable met logica, zodat kan worden voorkomen dat de waarde van de observable corrupt raakt (input validatie van de gebruiker).

Events

Hoewel niet extreem complex of geavanceerd, events zijn een belangrijk onderdeel van een goede applicatie. Ontwikkelaars kunnen zo inhaken op acties van de gebruiker, zoals een click, keyup, mouseleave en de overige JavaScript-events. Events kunnen worden toegevoegd op ieder DOM-element, en ook worden gecombineerd (Listing 8).

```
<!-- HTML -->
<ul data-bind="foreach: books">
  <li data-bind="text: title,
    event: { mouseover: $parent.displayCover,
      mouseout: $parent.hideCover,
      click : $parent.addToCart }">
  </li>
</ul>

// JavaScript
var self = this;

this.cart = ko.observableArray();
this.books = ko.observableArray([
  { title : "Mijn kookboek", coverImg : "kookboek.jpg" },
  { title : "Mijn roman", coverImg : "roman.jpg" }
]);

this.displayCover = function(book, e) {
  // Update the view using "book.coverImg"
};

this.hideCover = function(book, e) { };

this.addToCart = function(book, e) {
  // We can't use "this" here
  self.cart.push(book);
};
```

Listing 8: Gebruik van events

Er zijn een aantal dingen waar ontwikkelaars rekening mee dienen te houden bij het gebruik van events.

- Bij het gebruik van een with of foreach-binding wordt de context van het object meegegeven waar op dat moment over wordt geïtereerd. Aangezien de event-functies vaak op een hoger niveau zitten (bijvoorbeeld het ViewModel) dient in de HTML middels $parent of $root de juiste methode aangeroepen te worden.
- Knockout geeft het object waar de event-actie aan is gekoppeld als eerste parameter mee aan de functie. De tweede parameter is het event zelf, zoals een MouseEvent.
- Events zijn asynchroon. Daarom is het handig om een variabele naar het ViewModel aan te maken (vaak “self” of “vm” genoemd), aangezien de this-verwijzing naar het verkeerde object wijst. In het geval van het voorbeeld wijst het naar het book-object waar op dat moment op is gedrukt.

Templating

Binnen Knockout is het ook mogelijk om gebruik te maken van Templating. Dit kan erg krachtig zijn, aangezien (delen van) HTML hergebruikt kan worden. Over de verschillende Views heen hoeft dus geen code dubbel voor te komen. Ook dit werd al eerder getoond bij het gebruik van een module loader in frameworks zoals Prism voor C#.

Standaard heeft Knockout twee mogelijkheden voor templating:
1. Built-in template engine van Knockout.
2. Ondersteuning voor externe template engines, zoals jquery.tmpl en Underscore.

Het gebruik van templates is verder vrij simpel: Middels een binding en een id dat aangeeft welk template gebruikt moet worden kan er gemakkelijker een stuk herbruikbare HTML gemaakt worden. De container van de template zal niet worden vervangen, maar alleen inhoud wordt hieraan toegevoegd. Eventuele styling middels CSS en overige bindings blijven daardoor dus intact (Listing 9).

```
<!-- Template -->
<script type="text/html" id="tmpl-book">
  <h2 data-bind="text: title"></h2>
  <p>Author: <span data-bind="text: author"></span></p>
```

```
  </script>

  <!-- HTML -->
  <div data-bind="template: { name: 'tmpl-book', foreach:
books }"></div>

  <!-- Will be rendered as -->
  <div data-bind="template: { name: 'tmpl-book', foreach:
books }">
    <h2 data-bind="text: title">[Title]</h2>
    <p>Author: <span data-bind="text:
author">[Author]</span></p>
    <!-- More books using the template will follow here -->
  </div>
```

Listing 9: Gebruik van build-in templating

Het nadeel van Knockout is dat het geen support biedt voor externe templates. Zodoende dienen de templates middels de <script>-tag te worden opgenomen binnen eenzelfde HTML bestand. Voor ontwikkelaars zou het echter prettig zijn om daadwerkelijk externe HTML bestanden in te laden voor meer loskoppeling.

Ontwikkelaar Jim Cowart heeft om deze reden een Knockout External Template Engine-project uitgebracht. Middels deze Knockout plugin worden de externe HTML views asynchroon geladen. Op deze manier kunnen ontwikkelaars voor ieder ViewModel een aparte HTML pagina aanmaken (Listing 10).

```
  <!-- HTML -->
   <div data-bind="template: { name: module().name(),
     data: module().data() }"></div>

   // JavaScript
   function ModuleViewModel(name, data) {
     this.name = ko.observable(name);
     this.data = ko.observable(data);
   };

   function MainViewModel() {
     var vm = this;
     this.module = ko.observable();

     this.loadFirstModule = function() {
   // This will load "module.html" and bind the data from
the ModuleViewModel
       vm.module(new ModuleViewModel("module", new Module-
ViewModel()));
     };
   };
```

Listing 10: Gebruik van Knockout External Template Engine en verschillende ViewModels

In combinatie met een JavaScript-loader zoals RequireJS (die ook eventueel los de CSS kan inladen) kunnen ontwikkelaars volledig modulair en onafhankelijk hun applicatie in elkaar zetten. De Views (HTML), ViewModels en Models (en evt. de CSS) worden dan pas ingeladen op het moment dat de gebruiker ze nodig heeft (Figuur 3).

Mapping plugin

De mapping plugin is niet direct onderdeel van de Knockout-core, en dient dus als losse library te worden toegevoegd aan het Knockout-project. Deze plugin maakt het mogelijk om automatisch JavaScript-

Het gebruik van de External Template Engine brengt ook problemen met zich mee. Aangezien de HTML asynchroon wordt ingeladen, kan het zijn dat bepaalde elementen nog niet aanwezig zijn direct na het uitvoeren van de code. Dit kan ervoor zorgen dat stukken code niet werken, denk aan een id-selector van jQuery.

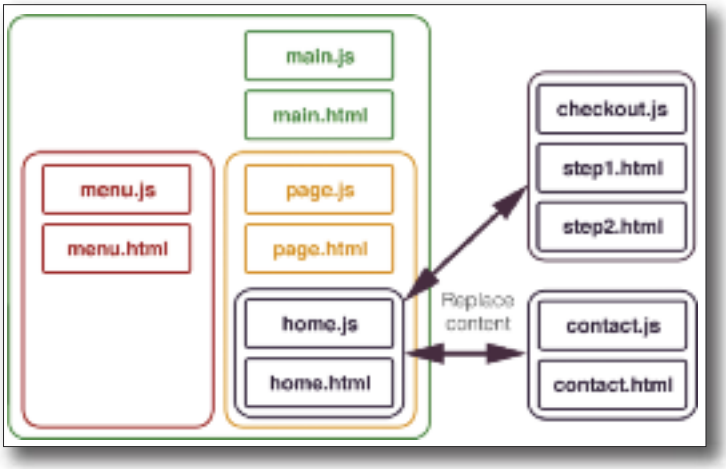


Fig. 3: Een Knockout-applicatie die volledig modulair opgezet is met externe bestanden

objecten om te zetten naar ViewModels, inclusief het aanmaken van observables, zonder dat de ontwikkelaar daar al te veel moeite voor hoeft te doen. Dit is bijzonder handig wanneer deze objecten als JSON objecten ontvangen worden (Figuur 4).

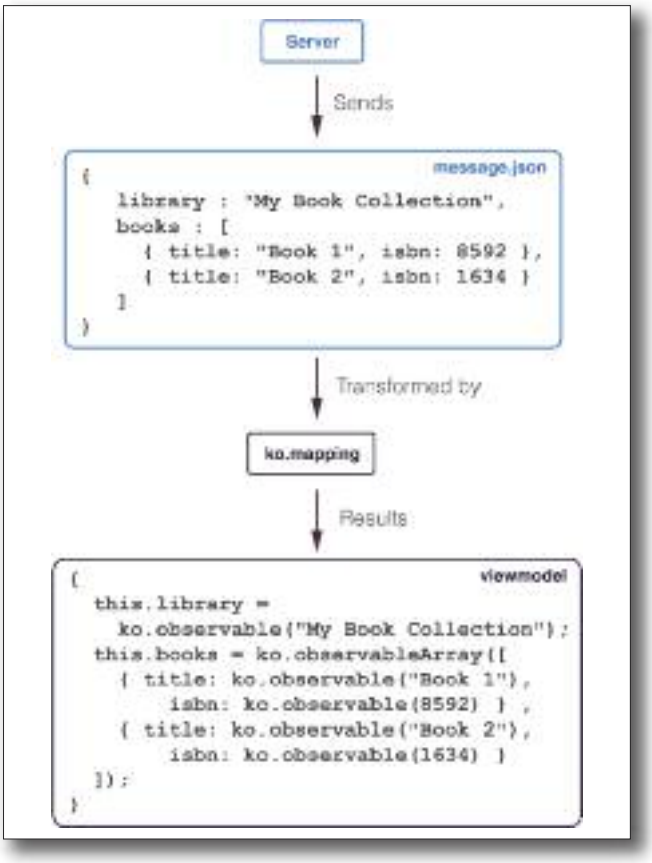


Fig. 4: Werking van mapping plugin

De mapping.fromJS() methode zorgt dus voor de omzetting naar een viewmodel met observable objecten. De mapping plugin kan dit ook de andere kant op (terug naar een unmapped, “plat” JSON-object) middels de mapping.toJS() methode. Dit mechanisme is enorm sterk in combinatie met een REST server, waarbij objecten middels JSON over en weer worden verstuurd en zo automatisch gekoppeld worden aan de view.

Hoewel automatische mapping een oplossing kan zijn om snel objecten van de server te consumeren binnen Knockout, dient er vaak nog extra logica te worden toegevoegd om tot het gewenste eindresultaat te komen. Ook hiervoor biedt de mapping plugin de nodige functionaliteit, door mapping options te specificeren. Een voorbeeld hiervan is het toevoegen van logica aan het binnenkomende object (Listing 11).

```
// JavaScript
var author = {
  gender : "male",
  name : "Shakespeare"
}

var mapping = {
  "name" : {
    update: function(options) {
      if(options.parent.gender() === "male") {
        return "Mr. " + options.data;
      } else {
        return "Ms. " + options.data;
      }
    }
  }
};

var authorVm = ko.mapping.fromJS(author, mapping);
// authorVm.name() will now return "Mr. Shakespeare"
```

Listing 11: Toevoegen van logica wanneer deze wordt gemapt

Naast de update-functie die de mapping plugin aanbiedt, zijn er nog een aantal andere mogelijkheden om de mapping functionaliteit te beïnvloeden. Enkele voorbeelden zijn dat ontwikkelaars properties uitsluiten van het mappen (ignore), extra properties toevoegen (include), het specificeren van een ander target en meer.

De update-functie gaat gelijk af wanneer de property wordt gemapt. Dit heeft een directe impact op het options.parent object, aangezien deze nog niet hoeft te garanderen dat alle properties al zijn gemapt. De mapping plugin gaat simpelweg de properties van boven naar beneden af zoals ze zijn gedeclareerd. M.a.w.: Het voorbeeld uit Listing 11 zou niet werken wanneer de declaratie van de gender en name properties uit het author-object omgedraaid zouden zijn.
De update-functie van name zou al afgaan, maar op dat moment bestaat options.parent.gender() nog niet.

Ondanks dat op deze manier sneller een applicatie kan worden ontwikkeld (de applicatie kan immers direct de objecten van de server consumen), dient er wel rekening te worden gehouden met een nadeel die deze methode met zich meebrengt. De properties van het JSON-object worden immers direct gekoppeld aan de view middels de bindings. De view dient dus te worden aangepast wanneer de structuur van de objecten wordt aangepast aan de kant van de server. Foutmeldingen hierover worden pas getoond nadat Knockout probeert de observables te binden: dat is op het moment wanneer de view wordt getoond.

Drawbacks

Zoals aangegeven in het begin van dit artikel, kan Knockout het leven van de ontwikkelaar een stuk aangenamer maken. Maar tegen welke prijs? Er zijn enkele drawbacks aan het framework, die voor verschillende ontwikkelaars doorslaggevend kunnen zijn om niet voor dit framework te kiezen. Een van deze redenen kan zijn vanwege de vele bindings die geplaatst moeten worden in de HTML (Listing 5). Dit zorgt ervoor dat de HTML zeer onoverzichtelijk kan worden. Ook is het -vanuit een zeer puristisch oogpunt - niet de bedoeling om functionele handlers te plaatsen in de HTML, deze is immers alleen bedoeld voor het opslaan van data. De wereld van HTML en JS overlappen elkaar nu nog sterker, iets wat ontwikkelaars juist uit elkaar willen houden.
Behalve dat de HTML onoverzichtelijk kan worden door de bindings, is het voor Knockout ook nodig om soms HTML-elementen toe te voegen, zodat hier bijvoorbeeld tekst aan kan worden ge-bind. Er zullen dus loze HTML-elementen zoals span, strong etc. worden toegevoegd aan de DOM, wat de HTML nog meer zal vertroebelen (Listing 12).

Iets wat niet is behandeld in de vorige hoofdstukken, maar Knockout wel krachtig kan maken, is het gebruik van bindings middels HTML commentaar (Listing 12). Door commentaar toe te voegen, en daar de prefix “ko” aan mee te geven, zal Knockout dit herkennen en hier iets mee doen. In het voorbeeld wordt over een lijst geïtereerd. Echter, het eerste item in de HTML dient altijd een bepaalde tekst te bevatten die niet in de lijst voorkomt. Dit kan ook handig zijn voor een tabel, waarbij de eerste regel er anders uit dient te zien dan de rest.
Wat je je wel dient te realiseren, is dat commentaar opeens functioneel is geworden. De code kan onbruikbaar worden door commentaarregels uit de HTML te halen, iets wat zeer vreemd is. Ook HTML minifiers – tools die ervoor zorgen dat je HTML zo klein mogelijk wordt – zullen commentaar weghalen, waardoor de code onbruikbaar wordt.

```
<!-- HTML element for binding only -->
<h2>
  Hello <span data-bind="text: name"></span>!
</h2>

<!-- Binding using HTML comments -->
<ol>
  <li>All authors:</li>
  <!-- ko foreach: books -->
  <li data-bind="text: author"></li>
  <!-- /ko -->
</ol>
```

Listing 12: Verschillende HTML drawbacks

Een ander aspect wat ook zeker belicht dient te worden, is de ondersteuning van SEO (Search Engine Optimization) en de browser-history. Out-of-the-box ondersteunt Knockout deze functionaliteit namelijk niet: alles is één pagina. Zoekmachines zoals Google en Bing zullen niet direct alle content herkennen die wordt ingeladen middels

de viewmodels, en dit wordt helemaal lastig wanneer de data dynamisch wordt geladen middels AJAX. Dit is sowieso een probleem voor websites die AJAX gebruiken om hun content te laden, maar Knockout biedt hier dus zelf geen ondersteuning voor en dient door ontwikkelaars zelf toegevoegd te worden. Dit kan bijvoorbeeld middels de HTML5 PushState API.
Om het gebruik van de browser-history (de "back- en forwardbuttons") mogelijk te maken, dient de ontwikkelaar dit zelf te bouwen, of gebruik te maken van een routing framework zoals Sammy.js. Omdat dit niet vanuit Knockout zelf wordt ondersteund, kan dit een reden zijn voor de ontwikkelaar om niet snel voor het framework te kiezen.

Ontwikkelaar Kevin Malakoff heeft daarom een nieuw framework gemaakt, genaamd Knockback. Deze brengt de sterke punten van Knockout en Backbone - een ander populair JavaScript framework - samen, met daarin onder andere ondersteuning voor routing. Echter, dit framework draagt op moment van schrijven versie 0.16.8 en wordt maar (actief) onderhouden door één ontwikkelaar. Ondanks dat Knockout van zichzelf cross-browser wordt ondersteund, dient er toch goed rekening gehouden te worden met oudere browsers. Wanneer een viewmodel een collectie bevat met enorm veel complexe objecten, en de view maakt gebruik van verschillende complexe bindings afhankelijk van die objecten, dan kunnen oudere browsers nog wel eens blijven hangen. Hoewel dit niet zozeer met Knockout te maken heeft, maar meer met de browser, is het systeem van data-binding wel de oorzaak en dient daarom ook in het achterhoofd gehouden te worden.

Conclusie

Zoals bij iedere techniek, framework en methode is Knockout niet de silver bullet voor alle problemen die webontwikkelaars tegenkomen. Echter, voor C# ontwikkelaars die het MVVM-pattern goed onder de knie hebben, zal dit framework ervoor kunnen zorgen dat ze sneller de overstap kunnen maken naar de wereld van webdevelopment en JavaScript. De declaratieve bindings en het systeem van automatische UI-updates helpen hier enorm bij.
Hoewel dit artikel de belangrijkste en meest fundamentele aspecten behandelt om een eerste goede Knockout applicatie te maken, zijn er nog enkele onderdelen niet belicht. Denk hierbij aan het bouwen van custom bindings, het gebruik van de utility classes uit de ko.utils namespace, het toevoegen van custom functies etc. De documentatie van Knockout is hierin erg uitgebreid en enthousiaste ontwikkelaars zullen hiervoor dan ook geen stap terug doen.
Knockout is volwassen genoeg om volwaardige applicaties mee te maken. Door de updates die blijven komen, de support van de community, en Microsoft die achter het project staat zal het framework een lange levensduur hebben. Genoeg tijd om een mooie webapplicatie met een punch te maken!

Referenties

Knockout Home	http://knockoutjs.com/
External Template Engine	http://tinyurl.com/ko-external
Knockback.js	http://tinyurl.com/ko-knockback

Marco Kuiper

Marco Kuiper is een .NET ontwikkelaar bij Info Support. Hij heeft een enorme voorliefde voor de front-end van applicaties, en een passie voor alle ontwikkelingen rondom webdevelopment en open web standaarden. Deze informatie deelt hij dan ook erg graag op zijn blog met de rest van de wereld.

CoreDispatcher en Tasks in Windows 8

By Dave Smit (Win8 Developer Sparked)

In een eerdere blogposts heeft Dave Smits laten zien wat het verschil is tussen tasks en threads, en eventuele niet voor de hand liggende gevolgen bij verkeerd gebruik: http://www.familie-smits.com/post/2012/11/07/Threads-versus-Tasks.aspx
Onlangs was hij met de Dispatcher bezig om weer onderdelen terug de UI thread te invoken en liep hij tegen precies het zelfde aan. Neem het volgende stuk code:

```
        Debug.WriteLine("step 1");
           await Dispatcher.RunAsync(
                  CoreDispatcherPriority.Normal, async () =>
           {
               Debug.WriteLine("step 2");
               await Task.Delay(1000);

               Debug.WriteLine("step 3");
           });
           Debug.WriteLine("step 4");
```

Verwacht zou zijn dat in de Debug window het volgende resultaat verschijnt:

```
step 1
step 2
step 3
step 4
```

Helaas is dat niet geval en zal 4 eerder worden uitgevoerd dan 3. Dat is het zelfde als we zagen met de threadpool. Om dit probleem op te lossen heb ik een Extension method geschreven op de CoreDispatcher die dit probleem oplost.

```
        public static class DispatcherExtensions
    {
        public static Task RunTaskAsync(
                  this CoreDispatcher dispather,
                  Windows.UI.Core.CoreDispatcherPriority
                  priority,
                  Func<Task> task)
        {
            TaskCompletionSource<object> tsc =
                   new TaskCompletionSource<object>();
            var t = dispather.RunAsync(priority, async () =>
            {
                try
                {
                    await task();
                    tsc.SetResult(null);
                }
                catch (Exception e)
                {
                    tsc.SetException(e);
                }
            });
            return tsc.Task;
        }
    }
```

Wat als volgt gebruikt kan worden

```
        Debug.WriteLine("step 1");
            await Dispatcher.RunTaskAsync(CoreDispatcher
                  Priority.Normal, async () =>
            {
                Debug.WriteLine("step 2");
                await Task.Delay(1000);

                Debug.WriteLine("step 3");
            });
            Debug.WriteLine("step 4");
```

Microsoft

Mijn innovatieve geest denkt in Windows Azure.

Windows Azure is het ontwikkelplatform in de cloud dat developers vrij baan geeft in hun denken. Bouw en draai applicaties in de cloud. Lanceer applicaties in minuten, in plaats van uren. Programmeer in meerdere talen en technologieën, zoals .NET, PHP en Java. Begin vandaag nog met innoveren. Ongehinderd door redundantie, bandbreedte of de beperkingen van uw serverconfiguratie.
Dat is Cloud Power.

Start vandaag nog met Windows Azure.
Probeer* Windows Azure gratis. Ga naar microsoft.nl/azure

Download de gratis app op
http://gettag.mobi

Windows Azure™

* Voor het uitproberen zijn creditcardgegevens nodig. Voor gebruik boven het introductieniveau betaalt u het standaardtarief.

Edwin van Wijk en Jeroen Hendricksen

Real-time Push Communicatie

Het web 2.0 tijdperk heeft ervoor gezorgd dat web applicaties veel interactiever en dynamischer zijn geworden. Steeds vaker is 'real-time' communicatie van server naar client ("push" communicatie) gewenst. Momenteel wordt daarvoor nog vaak gebruik gemaakt van achterhaalde technieken zoals (AJAX) Long Polling en Forever Frame, maar nieuwe technologieën bieden ons nieuwe mogelijkheden om eenvoudiger en sneller berichten naar al onze clients te kunnen distribueren. Twee van deze technologieën zullen we in dit artikel beschrijven, te weten SignalR (door Jeroen Hendricksen) en Azure Servicebus Push Notification Hubs (door Edwin van Wijk).

Deze technologie is zoals gezegd met name zinvol in situaties waarin we berichten zo snel mogelijk naar een (grote) groep van aangesloten clients willen versturen. Het grote voordeel hierbij is dat de server het initiatief kan nemen om iets direct te "pushen" naar de client i.p.v. dat de client actief moet vragen om informatie.

Praktijkcase : App2000

Afgezien van leren hoe deze technologie werkt, wilden we het ook testen in een praktijksituatie. We hebben hiervoor gekeken naar een toepassing die gebruik maakt van het landelijke P2000-systeem. Het P2000-systeem (tegenhanger van het niet publieke C2000-systeem) dient om berichten die via de meldkamers van de verschillende hulpdiensten (brandweer, ambulance en KNRM) zijn uitgegeven via een netwerk van zendmasten de ether in te sturen. Bij de hulpdienst waar het bericht voor bedoeld is gaat vervolgens de "pieper" af. Het is al lange tijd mogelijk om als burger deze berichten te ontvangen. Naast het gebruik van zogenaamde scanners zijn er ook verschillende websites die de P2000 berichten publiceren.

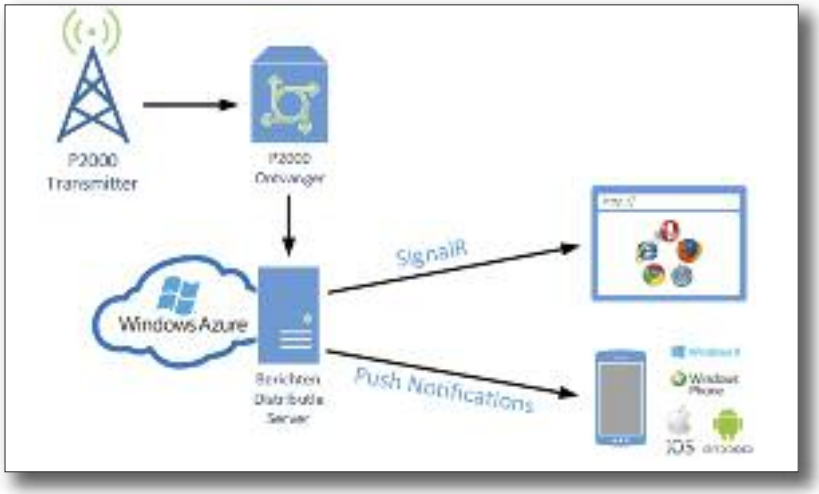


Fig. 1: App2000 architectuur

De praktische toepassing die we hebben bedacht is om deze berichten middels real-time communicatie te "pushen" naar verschillende devices. Dit zorgt ervoor dat burgers de berichten binnenkrijgen zonder dat ze hier actief naar hoeven luisteren. We hebben deze toepassing "App2000" genoemd.

Voordat we inzoomen op respectievelijk SignalR en Azure Servicebus Push Notifications geven we aan waar ze binnen onze praktijkcase van toepassing zijn (zie figuur 1).

Allereerst is er een P2000 ontvanger die de P2000 meldingen ontvangt van een P2000 transmitter. Deze ontvanger is al langer in gebruik en draait ergens op een door ons gehoste server. Deze stuurt de ontvangen meldingen die relevant zijn zonder vertraging naar onze berichten distributie server in de cloud (gehost in Windows Azure). Alle berichten die binnenkomen moeten vervolgens zo snel mogelijk naar aangesloten clients verstuurd worden. Wat betreft clients hebben we te maken met gebruikers die via een web applicatie berichten willen ontvangen, maar daarnaast ook gebruikers die een push notificatie op hun (mobiele) device willen ontvangen.

We willen zoveel mogelijk verschillende browsers en (mobiele) devices ondersteunen, zonder dat we daar in de implementatie heel specifiek rekening mee hoeven te houden. Ook willen we dat berichten die we naar de web clients pushen gebruik maken van de nieuwe WebSockets standaard die onderdeel is van de HTML5 standaard. Dit is waar het SignalR framework ons gaat helpen.

SignalR

The basics

De beste manier om met SignalR bekend te raken is door ermee aan de slag te gaan. Zoals gezegd, hebben we dit gedaan door een simpele voorbeeldapplicatie te bouwen waarmee we meldingen uit het P2000-systeem kunnen distribueren naar verschillende clients (zie ook figuur 1).

We zullen nu de stappen beschrijven die we hebben genomen om deze applicatie te realiseren. We beginnen daarbij met het opzetten van de berichten distributie server, die we hebben geïmplementeerd als een ASP.NET MVC4 applicatie.

Start Visual Studio 2012 en maak een nieuwe MVC4 web application aan.

Open de NuGet Package Manager Console en installeer SignalR door het volgende commando uit te voeren: “Install-Package Microsoft.AspNet.SignalR”.

Na wat geduld is ons project nu voorzien van references naar de benodigde assemblies om het SignalR framework te kunnen gebruiken. Wat opvalt is dat het package dat we installeren in de “Microsoft.AspNet” namespace valt. Dit is geen toeval, aangezien SignalR tegenwoordig onderdeel is geworden van het Microsoft ASP.NET platform. Daarnaast is het ook beschikbaar op GitHub onder een open-source licentie.

SignalR biedt verschillende niveaus van communicatie. Wij hebben gekozen voor de meest eenvoudige manier d.m.v. een SignalR “hub”. Een SignalR hub is een class die overerft van de “Hub” class uit het SignalR framework en eigenlijk de interface van de server met de aan te roepen operaties definieert.

Maak een nieuwe folder “Hubs” aan in het web project. Maak in deze folder een nieuwe public class die overerft van “Microsoft.AspNet.SignalR.Hub”. Voeg in deze hub de methode ”distributeMelding” toe. In figuur 2 is een voorbeeld van de class te vinden.

```
using Microsoft.AspNet.SignalR;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace App2000.Webserver.Hubs
{
    public class DistributionHub : Hub
    {
        public void distributeMelding(string melding)
        {
            Clients.All.broadcastMessage(DateTime.Now.ToString(
                "HH:mm:ss.fff") + " - " + melding);
        }
    }
}
```

Fig. 2: Hub class

De regel “Clients.All.broadcastMessage(data);” zorgt ervoor dat de melding die binnenkomt via de operatie “distributeMelding” verstuurd wordt naar alle verbonden clients. Omdat de “All” property op de ”Clients” class een DynamicObject (.NET 4.5) oplevert, kunnen we de parameters die we willen meegeven met de methode vrijelijk definiëren. De door SignalR gegenereerde client code zal vervolgens automatisch deze operatie met parameters beschikbaar maken. Hier komen we later nog op terug.

De Hub is nu aangemaakt, maar om deze geactiveerd te krijgen als we de webserver starten moeten we de hub eerst nog registreren bij ASP.NET. Dit doen we door in de “Global.asax.cs” file de volgende regel toe te voegen in de “Application_Start()” methode: “Route Table.Routes.MapHubs();”.

Omdat we te maken hebben met een MVC-applicatie is het belangrijk om de operatie aan te roepen net vóórdat “RouteConfig.RegisterRoutes” wordt aangeroepen, anders wordt onze Hub niet correct geregistreerd en ook niet opgestart bij het starten van de web applicatie. Het uiteindelijke resultaat vinden we terug in figuur 3.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Routing;

namespace App2000.Webserver
{
    public class WebApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {

            AreaRegistration.RegisterAllAreas();
            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

            // Register the default hubs route: ~/signalr
            RouteTable.Routes.MapHubs();

            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
}
```

Fig. 3: Hub registratie in de route table

Om te verifiëren dat we tot nu toe alles goed hebben gedaan, starten we het web project vanuit Visual Studio. In de standaard browser wordt nu een url geopend als “http://localhost:poort”. Achter deze url plakken we “/signalr/hubs”, zodat we uitkomen op “http://localhost:poort/signalr/hubs”. De browser zou nu een javascript bestand moeten retourneren, wat betekent dat het SignalR deel voor wat betreft de server nu correct werkt.

De web-client

Nu we de server gereed hebben gaan we de web-client maken. Deze client moet de meldingen die de server distribueert op het scherm tonen.

Maak een nieuwe webpagina aan. Om ons voorbeeld zo elementair mogelijk te houden kiezen we voor een HTML pagina genaamd “Default.html” in de root van het web application project.

In dit HTML bestand plaatsen we in de header een script-tag die wijst naar “/signalr/hubs”. Bij het opvragen van dit javascript bestand in de browser, wordt door SignalR dynamisch een script gegenereerd waarin onze eerder aangemaakte hub ”distributionHub” en de operatie ”distributeMelding” aanwezig zijn.

Ook voegen we een referentie toe naar JQuery en JQuery.SignalR, een tweetal vereiste componenten die geïnstalleerd worden met SignalR en waar we in de gegenereerde client een afhankelijkheid mee hebben. De code is weergegeven in figuur 4.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>App2000 meldingen</title>
    <script src="Scripts/jquery-1.6.4.min.js"></script>
    <script src="Scripts/jquery.signalR-1.0.1.min.js"></script>
    <script src="/signalr/hubs"></script>

</head>
<body>

</body>
</html>
```

Fig. 4 : SignalR javascript referenties

Controleer altijd of de versienummers in de referenties naar de javascript bestanden nog overeenkomen met de versie van SignalR die

is geïnstalleerd en pas deze aan indien nodig.
Nu de referenties naar de benodigde javascript files zijn toegevoegd, voegen we op de pagina een knop toe waarmee we berichten kunnen verzenden (als simulatie van de back-end). Daarnaast voegen we een sectie toe waarin meldingen in een lijst gerepresenteerd kunnen worden. De HTML die we hiervoor nodig hebben is weergegeven in figuur 5. Dit voegen we toe binnen de nog lege body van de HTML pagina.

```
<div>
    <input type="button" id="btnVerzendBericht" value="Verzend een bericht" />
</div>
<div>
    <ul id="meldingen">
    </ul>
</div>
```

Fig. 5: HTML code

```
<script type="text/javascript">
    $(function () {
        // Proxy created on the fly
        var hub = $.connection.distributionHub;

        // Declare a function on the chat hub so the server can invoke it
        hub.client.broadcastMessage = function (melding) {
            $('#meldingen').append('<li>' + $('<div/>').text(melding).html() +
                '</li>');
        };

        // Start the connection
        $.connection.hub.start().done(function () {
            hub.server.distributeMelding("De eerste melding");
            $("#btnVerzendBericht").click(function () {
                hub.server.distributeMelding("Bericht van de server " +
                    Math.floor(Math.random() * 1001));
            });
        });
    });
</script>
```

Fig. 6: Javascript met client code

In de nu volgende stap gaan we ervoor zorgen dat meldingen die via het SignalR framework binnenkomen ook daadwerkelijk op de pagina getoond worden. Daarom voegen we de javascript code die is weergegeven in figuur 6 toe aan het script-block in de header van de HTML-pagina, na de referentie naar "/signalr/hubs".

SignalR stelt ons in de client een instantie van de hub beschikbaar, welke toegankelijk is via "$.connection.distributionHub". De naam "distributionHub" komt overeen met de naam van de hub class in onze server code is, maar dan in camelCasing stijl. Op deze hub zit een client property met een event "broadcastMessage" dat afgaat als de server een bericht naar ons pusht. Hierop abonneren we ons door een function te registreren die dezelfde methode signatuur heeft als de "broadcastMessage" operatie die in onze DistributionHub wordt aangeroepen. De implementatie hiervan is dat een binnenkomend bericht toegevoegd wordt in de unordered list op de pagina.

Om nu de verbinding naar de hub te starten, roepen we "$.connection.hub.start()" aan. Zodra de verbinding is gemaakt (".done" callback), versturen we voor testdoeleinden een bericht met als inhoud de tekst "De eerste melding" zodat we direct kunnen verifiëren dat we een succesvolle verbinding hebben. Als laatste koppelen we een functie aan de click handler van de "Verzend een bericht" knop. De implementatie hiervan verstuurt een bericht met een willekeurig nummer naar de hub. Onze client is nu gereed om meldingen te ontvangen en te dienen als simulator van de back-end middels de "Verzend een bericht" knop.

Start de web applicatie (zodat de server gestart wordt) en navigeer naar "http://localhost:poort/Default.html". Na het openen van de pagina en het succesvol starten van de hub wordt automatisch het bericht "De eerste melding" verzonden naar de client. Open nu meerdere browsers of tabs naar dezelfde url om meerdere clients te simuleren.

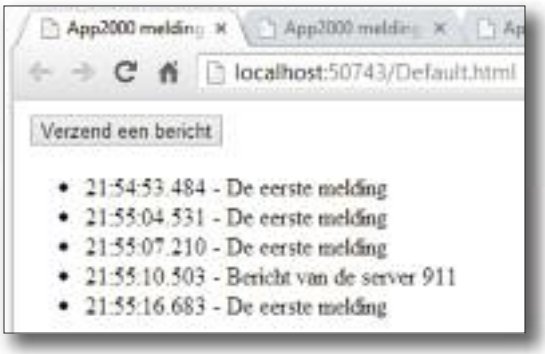


Fig. 7: SignalR in Chrome

Voor elke client zal een "De eerste melding" bericht worden verzonden. Als we nu op de "Verzend een bericht" knop drukken, dan wordt er wederom een bericht verzonden naar alle verbonden clients. Dit bericht gaat van de client naar de server en van daaruit wordt het gedistribueerd naar alle verbonden clients. Een voorbeeld van het resultaat is weergegeven in figuur 7.
In onze voorbeeldapplicatie is het uiteraard de P2000 ontvanger die berichten naar de server stuurt en de server die vervolgens het bericht naar alle verbonden clients doorstuurt.

WebSockets en Chrome

De meest efficiënte manier van communicatie tussen de client en server is door gebruik te maken van het WebSockets protocol. SignalR gebruikt dit protocol als het ondersteund wordt door de browser, maar zal automatisch terugvallen op alternatieven als Server Sent Events, Forever Frame en Long polling als WebSockets niet worden ondersteund. Het is interessant voor ons om te controleren of daadwerkelijk gebruik wordt gemaakt van WebSockets. In Chrome is dit eenvoudig te achterhalen. Ga hiervoor naar de developer view van de Default.html pagina uit het vorige hoofdstuk. Ga dan naar het tabje "Network", vernieuw de webpagina (F5) als het tabje nog leeg mocht zijn en selecteer de verbinding die begint met de naam "connect?transport=xxx". Als er een WebSockets verbinding mogelijk is met de server dan bestaat er nu een tabje "Frames" waar de berichten (frames) die over de lijn zijn gegaan worden weergegeven. Dit zien we in Figuur 8. Het bewijs dat daadwerkelijk gebruik wordt gemaakt van het WebSockets protocol is bij deze geleverd.

Fig. 8: SignalR debugging in Chrome

Op het moment van schrijven ondersteunen de laatste stabiele versies van de meest gangbare browsers als Chrome, Firefox en Opera de WebSockets standaard. Zie voor een uitgebreide lijst van browsers die het WebSockets protocol ondersteunen de website “http://caniuse.com/websockets”. Deze lijst wordt regelmatig bijgewerkt.

Wrap-up

We hebben gezien hoe we SignalR in kunnen zetten om real-time berichten te pushen naar meerdere verbonden clients door niet veel meer te doen dan het installeren van het SignalR-framework middels NuGet, het aanmaken van een eenvoudige Hub class en een webpagina met relatief weinig HTML en javascript. Ook hebben we gezien hoe we kunnen controleren dat SignalR functioneert en of er gebruik wordt gemaakt van het WebSockets protocol. Wij denken dat de kracht en de eenvoud van SignalR het een zeer breed inzetbaar raamwerk maakt voor verschillende toepassingen.

Push notifications

Naast de gebruikers via de website op de hoogte houden m.b.v. SignalR, wilden we gebruikers tevens op de hoogte brengen van P2000 meldingen op hun mobiele device m.b.v. push notifications. Bij mobiele devices denken we aan smartphones en tablets maar ook een Windows 8 PC kan push notifications ontvangen.

Push notifications zijn berichten die op initiatief van een server (back-end) naar een mobiel device worden verstuurd. De gebruiker krijgt dan - zonder dat een client applicatie actief hoeft te zijn op het device - een melding in beeld. In dit artikel ligt de focus op het versturen van tekst, maar naast tekst kan een push notification ook een afbeelding of een geluidsfragment bevatten.

Naast verschillende typen data kunnen push notifications ook verschillende weergaven hebben op een device. Een “alert” of “toast” is een melding die de gebruiker in beeld krijgt onafhankelijk van de actieve applicatie. Een push notification kan ook de informatie op een icoon (of “tile” in Windows RT / Phone termen) bijwerken. Denk bijvoorbeeld aan het aantal ongelezen e-mails op het icoon van een e-mail app. Dit wordt ook wel een “badge” genoemd.

Infrastructuur

Voor het versturen van push notifications heeft elk platform een eigen Push Notifications Service (PNS). Zo biedt Apple de “Apple Push Notifications Service” (APNS), Microsoft de “Windows push Notifications Service” (WNS) en Google “Google Cloud Messaging” (GCM). Al deze platformen werken wel nagenoeg volgens hetzelfde interactiepatroon (zie figuur 9).

Fig. 9: PNS infrastructuur

1. De client app op het device haalt bij de platform specifieke PNS zijn device-handle op.
2. De client app registreert deze device-handle bij de back-end.
3. De back-end stuurt (als gevolg van een bepaalde gebeurtenis) een push notification naar de PNS, vergezeld van de device-handle.
4. De PNS stuurt de push notification naar het desbetreffende device.

Uitdagingen

Bij het gebruik van push notifications zijn een aantal uitdagingen te onderkennen. De eerste uitdaging is platform afhankelijkheid. De back-end moet zelf functionaliteit bieden om device-handles op te slaan voor alle platformen die ondersteund worden door de app. Daarnaast moet de server in staat zijn om push notifications in het specifieke formaat van de ondersteunde platformen te versturen. Dit betekent dat de ontwikkelaars kennis moeten hebben van verschillende API’s en berichtformaten. Dit leidt af van het bouwen van de uiteindelijke functionaliteit van de app.

De tweede uitdaging is schaalbaarheid. Als een app succesvol is en bijvoorbeeld één miljoen gebruikers heeft, zal de server in staat moeten zijn om in korte tijd één miljoen berichten in het juiste formaat te versturen. Dit vergt veel resources voor de back-end en infrastructuur.

Een wat meer functionele uitdaging is routering (of filteren) van berichten. Vaak bestaat de behoefte om niet alleen berichten naar alle geregistreerde clients te kunnen sturen (broadcast), maar ook onderscheid te kunnen maken tussen verschillende groepen clients. Denk daarbij aan een nieuws app. Het zou mooi zijn als de gebruiker zelf kan aangeven in welke nieuwscategorieën hij of zij is geïnteresseerd en dan alleen nieuwsberichten die in deze categorieën vallen ontvangt. De back-end zal zelf functionaliteit moeten bieden om dit publish / subscribe mechanisme en het beheer ervan door de clients te ondersteunen.

Oplossingen

Om het hoofd te kunnen bieden de bovengenoemde uitdagingen, biedt het Windows Azure platform verschillende oplossingen. Het is bijvoorbeeld mogelijk om push notifications te versturen vanuit een Windows Azure Mobile Service. Ondanks dat deze oplossing een aantal uitdagingen wegneemt, blijft nog wel aardig wat werk voor de ontwikkelaar aan de server kant nodig. Een andere onderdeel van Azure dat een oplossing biedt voor alle bovengenoemde uitdagingen, is Windows Azure Servicebus Push Notification Hubs. Hierop focussen we in dit artikel en leggen uit hoe het werkt. Het is enigszins verwarrend, maar als we in dit deel van het artikel spreken over een “hub”, dan bedoelen we dus een push notification hub en niet een SignalR hub.

De push notification hubs feature van Windows Azure is in januari 2013 als gratis preview beschikbaar gekomen. Op dit moment (maart 2013) is er alleen ondersteuning beschikbaar voor het iOS (Apple) platform en het Windows RT (Microsoft) platform. Ondersteuning voor Android en Windows Phone komt later. De inschatting is dat de push notification hubs feature ergens medio 2013 de productie status zal krijgen. Dan zal ook bekend worden wat de kosten van het gebruik zullen zijn.

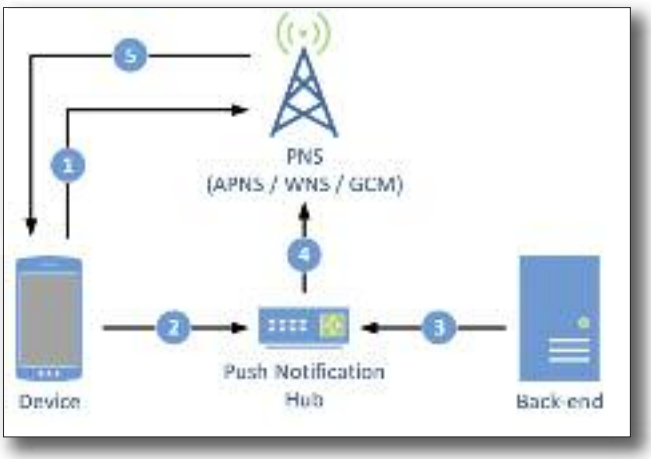


Fig. 10: Infrastructuur met push notification hub

Push notification hubs maken het mogelijk om platform onafhankelijk push notifications te versturen. Dat wil zeggen dat wanneer de back-end een push notification wil versturen naar alle clients, deze slechts één bericht hoeft te verzenden naar de hub. De hub zorgt ervoor dat dit bericht bij alle geregistreerde devices wordt afgeleverd, rekening houdend met het specifieke platform van elk geregistreerd device. Dit betekent dat de ontwikkelaar niet meer hoeft te weten hoe te communiceren met de verschillende platformen, maar slechts de API en het formaat van de hub hoeft te kennen. In figuur 10 is dit schematisch weergegeven.

1. De client app op het device haalt bij de platform specifieke PNS zijn device-handle op.
2. De client app registreert deze device-handle bij de hub.
3. De back-end stuurt (als gevolg van een bepaalde gebeurtenis) een push notification naar de hub.
4. De hub stuurt voor elk geregistreerd device de push notification door naar de PNS van het desbetreffende platform.
5. De PNS stuurt de push notification naar het device.

De push notification hubs infrastructuur is gebouwd bovenop de Azure Servicebus. De Azure Servicebus is gebouwd voor schaalbaarheid en kan dan ook miljoenen clients bedienen. Dit betekent dat een app die miljoenen clients heeft, niet de administratie van al deze miljoenen clients voor zijn rekening hoeft te nemen. Daarnaast hoeft - zoals eerder genoemd - bij een gebeurtenis dus slechts één bericht naar de hub verzonden te worden door de back-end en zorgt de hub dat alle (miljoenen) clients snel een platform specifieke notificatie ontvangen.

Push notification hubs bieden ook zogenaamde “Tags” om notificaties te kunnen routeren. Elke client kan optioneel één of meerdere tags meegeven bij het registreren. Zodra een notificatie voorzien van een tag wordt verzonden via de hub, zullen alleen die clients die zich met deze tag hebben geregistreerd deze notificatie ontvangen. We zien dit schematisch weergegeven in figuur 11.

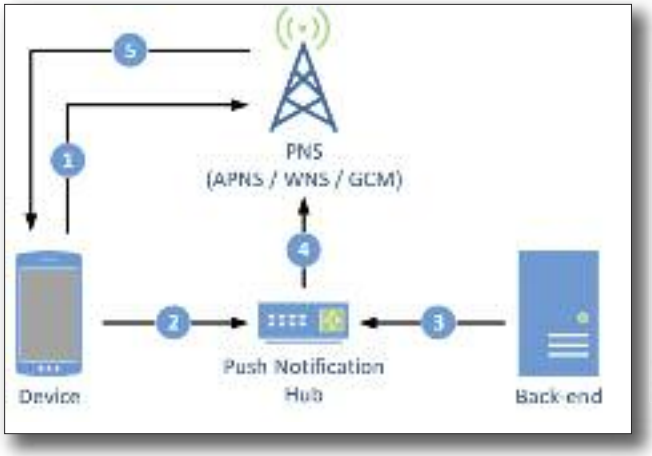


Fig. 11: tags

Device 1 heeft zich geregistreerd met de interesses (tags) “Sport” en “Economie”. Device 2 heeft zich geregistreerd met interesse “Technologie”. Een push notification met tag “Sport” (a) wordt alleen bij device 1 afgeleverd. Een push notification met tag “Technologie” (c) wordt alleen bij device 2 afgeleverd. Een push notification zonder tag (b), zal bij alle devices worden afgeleverd onafhankelijk van de tags waarmee clients zich eventueel hebben geregistreerd. De mogelijkheid om een broadcast te sturen naar alle clients blijft dus bestaan ondanks het gebruik van tags.

Praktijkcase : App2000

Binnen de App2000 toepassing, hebben we tags gebruikt om berichten gericht per regio te kunnen versturen. Een mobiel device update regelmatig de registratie bij de App2000 hub met een tag die de regio representeert waar het device zich op dat moment bevindt (op basis van GPS positie). Als er een melding verschijnt, bepaalt de berichten distributie server (zie figuur 1) op basis van de beschikbare gegevens in de melding de regio waar de melding betrekking op heeft. Vervolgens wordt er een push notification naar de hub gestuurd voorzien van een regio tag. Alleen die devices in de desbetreffende regio ontvangen vervolgens deze melding.

Platform onafhankelijk vs. “native”

Zoals eerder aangegeven bieden push notification hubs de mogelijkheid om platform onafhankelijk push notifications te versturen. Het is echter ook mogelijk om platform specifiek (ofwel “native”) te werken. Het verschil zit hem in de plek waar de platform specifieke inhoud van de push notification wordt opgesteld. Wanneer de back-end native push notifications wil versturen, wordt de inhoud van de push notification door de back-end in het platform specifieke formaat opgebouwd. Een push notification naar een Windows device bevat bijvoorbeeld een stuk XML terwijl een push notification naar een Apple (iOS) device een stuk JSON bevat. In figuur 12 is een voorbeeld van een Windows RT push notification bericht weergegeven.

```
<toast>
  <visual>
    <binding template="ToastText01">
      <text id="1">Dit is een test push notification!</text>
    </binding>
  </visual>
</toast>
```

Fig. 12: Windows RT push notification

Push notification hubs bieden ons ook de mogelijkheid om te werken met “templates” (sjablonen). Elke client moet dan bij het registreren bij de hub een template opgeven. Deze template is in feite een platform specifiek push notification bericht, waarbij middels expressies op bepaalde plekken in het bericht wordt aangegeven welke gegevens moeten worden ingevuld. In figuur 13 is een voorbeeld van een Windows RT template weergegeven.

```
<toast>
  <visual>
    <binding template="ToastText01">
      <text id="1">$(msg)</text>
    </binding>
  </visual>
</toast>
```

Fig. 13: Windows RT push notification

In de template is te zien dat middels de expressie “$(msg)” de waarde van een property met de naam “msg” moet worden ingevuld. Binnen de SDK zijn klassen beschikbaar om eenvoudig alle verschillende templates aan te kunnen maken. We zullen hier een voorbeeld van zien als we gaan kijken naar de code later in dit artikel.
Wanneer gebruikgemaakt wordt van templates, hoeft de back-end maar één push notification te versturen naar de hub, waarbij een set van naam-waarde-paren (property-bag) moet worden meegegeven. In het voorbeeld zal deze property-bag een property met de naam “msg” bevatten. De hub zal per geregistreerde client de bijbehorende template bepalen, de waarde van de “msg” property uit de property-bag in de template invullen en het ingevulde platform specifieke bericht doorsturen naar de desbetreffende PNS. Dit is uiteraard voor de back-end de meest eenvoudige manier van het versturen van push notifications.

Security

Om veilig push notifications te kunnen versturen, zijn een aantal maatregelen genomen. Allereerst moet de app die zich gaat registreren bij een hub bekend zijn bij de PNS van het desbetreffende platform. Als onderdeel van de registratie van een app bij de PNS (dit gaat via de developer portal van het desbetreffende platform), wordt voor de app een geheim gegeven gegenereerd. Voor een Windows RT app is dit bijvoorbeeld een “package security id (SID)” en een “client secret”. Voor een iOS app is dit een X509 certificaat. Dit gegeven wordt vervolgens bij de hub vastgelegd zodat een koppeling tussen de app en de hub is gelegd. Vanaf dit moment is de app geautoriseerd om zich te registreren bij de hub en is de hub geautoriseerd om push notifications te versturen naar de PNS.

De hub zelf is ook beveiligd tegen ongeoorloofd gebruik. Bij de creatie van een hub worden een tweetal “shared access signatures” (SAS) gegenereerd. Met de eerste SAS wordt alleen het “Listen” recht verkregen. Dit recht is nodig om te kunnen registreren bij de hub en push notifications te kunnen ontvangen en wordt dus gebruikt door de app op het client device. De tweede SAS biedt “Listen”, “Manage” en “Send” rechten. De back-end gebruikt deze SAS om push notifications te kunnen versturen naar de hub.

Hoe te gebruiken?

We hebben het tot nu toe slechts over de concepten van push notification hubs gehad. In dit onderdeel laten we zien hoe gebruikgemaakt kan worden van push notification hubs. Hiervoor zijn een aantal zaken nodig. In dit artikel ligt de focus op push notifications naar een Windows RT app en daarvoor zijn nodig:

- een Windows Store developer account,
- een Windows Azure subscription,
- de Azure Servicebus managed SDK for Windows RT,
- de Azure Servicebus .NET Preview SDK en
- een editie van Visual Studio 2012
  (alle edities zijn mogelijk (ook Express))

Om een Windows Store app te kunnen ontwikkelen die gebruikmaakt van push notifications, is een Windows Store developer account nodig (zie https://appdev.microsoft.com/StorePortals/en-us/Account/Signup/Start).

Om push notification hubs te kunnen gebruiken is een Windows Azure subscription nodig. (zie http://www.windowsazure.com/nl-nl).

De Azure Servicebus managed SDK for Windows RT biedt de API om vanuit een Windows RT app gebruik te kunnen maken van de Azure Servicebus features (te downloaden van: http://go.microsoft.com/fwlink/?LinkID=277160). Deze SDK bevat een assembly “Microsoft.WindowsAzure.Messaging.Managed.dll”.

De Azure Servicebus .NET preview SDK is een NuGet package (http://nuget.org/packages/ServiceBus.Preview). Als dit package wordt geïnstalleerd, zal een reference worden toegevoegd naar de assembly “Microsoft.ServiceBus.Preview.dll”. Deze SDK biedt de API om gebruik te kunnen maken van de push notification hubs.

Mini tutorial

In de volgende paar stappen laten we zien hoe een eenvoudige app kan worden gemaakt die push notifications kan ontvangen van een push notification hub. Tevens beschrijven we een eenvoudige test back-end om push notifications te kunnen versturen.

Maak om te beginnen een nieuw Windows Store Grid App project aan in Visual Studio 2012. Voeg een reference toe naar de bovengenoemde Azure Servicebus managed SDK assembly (“Microsoft.WindowsAzure.Messaging.Managed.dll”).

Klik rechts op de project file en kies de optie “Manage NuGet Packages...”. Zoek online naar het package “Servicebus.Preview” en installeer dit package (zie figuur 14).

Fig. 14: Installeer NuGet package

Vervolgens moeten we de app gaan koppelen aan de Windows Store (hiervoor moet een Windows Store developer account beschikbaar zijn). Klik hiervoor rechts op de project file en kies de optie “Store, Associate App with the store...”. Er zal gevraagd worden om in te loggen met het Microsoft account waarmee het Windows Store developer account aangemaakt is. Na te zijn ingelogd wordt een dialoog getoond waarin een app kan worden geselecteerd (zie figuur 15).

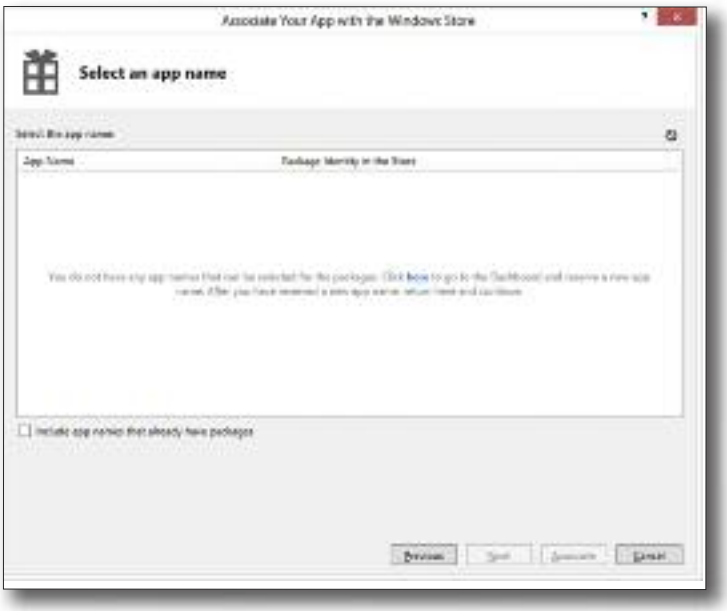


Fig. 15: selecteren app

Als er nog geen apps zijn geregistreerd is deze dialoog leeg. We gaan nu eerst de app bekend maken in de store. Klik op de link die in het dialoogscherm wordt getoond om een naam voor de app te reserveren. De Windows Store developer portal zal worden getoond met de mogelijkheid een naam voor de app te reserveren. Vul hier de gewenste (unieke!) naam van de app in en bevestig de naam. In het

voorbeeld hebben we gekozen voor “App2000ClientRT”. De naam zal worden gereserveerd (zie figuur 16).

Fig. 16: App naam gereserveerd

Klik nu op de “Advanced features” knop. Klik vervolgens op de link “Push notifications and live services info”. Er zal een nieuw scherm getoond worden. Klik in de “Push notifications” sectie op “Authenticating your service”. In het volgende scherm worden de “package security identifier” (SID) en een “client secret” getoond. Bewaar deze gegevens goed (copy/paste).

```
<VisualElements
  DisplayName="App2000ClientRT" Logo="Assets\Logo.png"
  SmallLogo="Assets\SmallLogo.png" Description="App2000Client"
  ForegroundText="light" BackgroundColor="#464646"
  ToastCapable="true" >
```

Fig. 17: Package.appxmanifest

Ga nu terug naar Visual Studio. Het dialoogscherm zal automatisch verversen en de zojuist gereserveerde naam van de app zal getoond worden. Klik op de naam van de app en klik op de “Next” knop. Er zal een samenvatting gegeven worden van de registratie. Klik vervolgens op de “Associate” knop. De app is nu aan de Windows Store gekoppeld.

Om push notifications te kunnen ontvangen, moeten we dit configureren in de app. Klik hiervoor rechts op het “Package.appxmanifest” bestand in het project en kies “View Code”. De inhoud (XML) zal getoond worden. Voeg aan het “VisualElements” element het attribuut “ToastCapable” toe met waarde “true” (zie figuur 17) en sla het bestand op.Nu moet een Azure Servicebus Push Notification Hub aangemaakt worden. Ga hiervoor naar de Azure Management Portal (https://manage.windowsazure.com) en log in.

Fig. 18: Nieuwe hub aanmaken

Fig. 19: Hub configuratie

Klik op de “Service Bus” feature in het menu links. Maak een nieuwe Servicebus namespace aan door onderaan het scherm op de “+ (Create)” knop te klikken. Vul het dialoogscherm in (kies een unieke namespace naam) en bevestig deze gegevens. Wacht totdat de namespace is geactiveerd. Klik nu op de nieuwe Servicebus namespace om deze te selecteren. Er wordt een dashboard getoond. Klik bovenaan het scherm op de optie “Push Notification Hubs”. Klik op de link “Create a new notification hub”. Vul alle benodigde gegevens in en bevestig deze (zie figuur 18).

Klik - nadat de hub is aangemaakt - op de hub om deze te selecteren. Er wordt een dashboard getoond. Klik vervolgens bovenaan het scherm op de “Configure” link. Vul op de pagina die dan wordt getoond, de “package security identifier” en “client secret” in die we tijdens de registratie van de app in de Windows Store developer portal hebben bewaard (zie figuur 19).

Nu hebben we de push notification hub en de app aan elkaar gekoppeld. De hub is nu geautoriseerd om push notifications te versturen via de Microsoft PNS (WNS) naar de app.
We moeten nu nog één ding doen voordat we de code in kunnen duiken. We hebben namelijk een “Shared Access Signature” van de hub nodig om als client te mogen registreren bij de hub en push notifications te mogen ontvangen. Klik hiervoor op de “View SAS key” knop onderaan het scherm. Er wordt een dialoog scherm getoond met een tweetal SAS keys. Bewaar voor de client de eerste (“DefaultListenSharedAccessSignature”) sleutel.

We zijn nu gereed om code te gaan schrijven en onze client app met push notifications uit te rusten. Om gebruik te kunnen maken van de hub, moeten we eerst een connectie opzetten naar de hub.

```
sealed partial class App : Application
{
    NotificationHub _notificationHub;

    public App()
    {
        var cn = ConnectionString.CreateUsingSharedAccessSecretWithListenAccess(
            "sb://app2000ps.servicebus.windows.net/", "WT9.....................0w==");
        _notificationHub = new NotificationHub("hub001", cn);

        this.InitializeComponent();
        this.Suspending += OnSuspending;
    }

    // ...
```

Fig. 20: Initialisatie notificationhub

In de “App.xaml.cs” file (zeg maar de “Main” van de Windows RT applicatie), voegen we een globale variabele van het type “NotificationHub” (in namespace “Microsoft.WindowsAzure.Messaging”) toe. Vervolgens initialiseren we deze variabele in de constructor van de class (zie figuur 20).

```
protected override async void OnLaunched(LaunchActivatedEventArgs args)
{
    await InitializeNotificationsAsync();

    // ...
}

protected async override void OnActivated(IActivatedEventArgs args)
{
    base.OnActivated(args);
    await InitializeNotificationsAsync();
}
```

Fig. 21: Initialiseer registratie in OnLaunched en OnActivated

We gebruiken de “_notificationHub” variabele in de rest van de klasse om met de hub te communiceren. In de methoden “OnLaunched” en “OnActivated” voegen we een aanroep naar de methode “InitialiseerPushNotificationsAsync” toe (zie figuur 21).

In de methode “InitialiseerPushNotificationsAsync” wordt eerst een aanroep naar “RefreshRegistrationsAsync” gedaan. Dit is altijd nodig voordat er iets met een registratie wordt gedaan. Daarna wordt gecontroleerd of er een registratie bestaat en als dat niet het geval is wordt een nieuwe (template) registratie gecreëerd (zie figuur 22).

```
async Task InitialiseerPushNotificationsAsync()
{
    await _notificationHub.RefreshRegistrationsAsync();

    if (!await _notificationHub.RegistrationExistsForApplicationAsync("MijnWTToastRegistratie"))
    {
        var template = NotificationXmlBuilder.CreateToastText01Xml("${msg}");
        await _notificationHub.CreateTemplateRegistrationForApplicationAsync(
            template, "MijnWTToastRegistratie");
    }
}
```

Fig. 22: Initialiseer client registratie

De te gebruiken template wordt gecreëerd met behulp van één van de methoden op de “NotificationXmlBuilder” klasse. Deze klasse bevat voor alle verschillende soorten templates een specifieke methode, waarbij afhankelijk van het aantal invulvelden in de template een aantal parameters kan worden meegegeven (zie figuur 23).

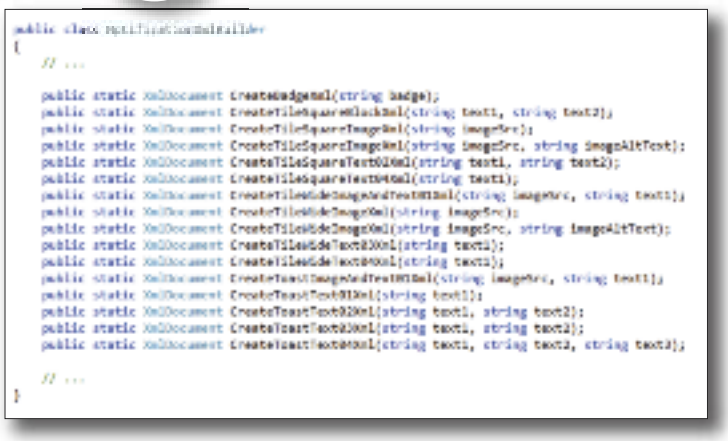


Fig. 23: een selectie van de methoden op de NotificationXmlBuilder

Fig. 24: back-end simulatie

Wanneer we nu de Windows Store app starten, zal deze zich registreren bij de push notification hub en gereed zijn om push notifications te ontvangen.

Om dit te testen maken we een eenvoudige test applicatie die de back-end simuleert. Maak hiervoor een nieuw “Console Application” project aan in Visual Studio 2012. Voeg het NuGet package “Service-Bus.Preview” toe aan het project. In de “Main” methode voegen we code toe om een connectie te maken met de hub en een toast notification te versturen (zie figuur 24).

Omdat de Window RT app zich met een template heeft geregistreerd bij de hub, gebruiken we in de back-end de “SendTemplateNotification” methode. We geven hier alleen een collectie van naam-waarde paren (property-bag) aan mee. De waarde van de property genaamd “msg” zal ingevuld worden in de template en worden getoond in de push notification (zie figuur 25).

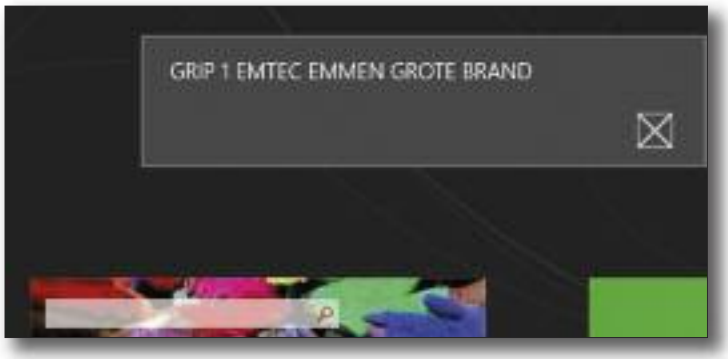


Fig. 25: push notification in Windows 8

Wrap-up

In dit artikel hebben we een klein deel van de mogelijkheden laten zien van de ServiceBus Push Notification Hubs feature van Windows Azure. Deze technologie biedt ons de mogelijkheid om op een eenvoudige wijze ondersteuning voor push notifications voor verschillende (mobiele) platformen toe te voegen aan onze apps. Daarbij is het platform schaalbaar. Het dus geen probleem als die killer app met ondersteuning voor push notifications die u in gedachten heeft, binnen 2 weken na publicatie wereldwijd 2 miljoen downloads heeft bereikt.

Conclusie

De twee technologieën die in dit artikel zijn beschreven bieden ons een krachtige en eenvoudige manier om een grote groep gebruikers / devices notificaties te kunnen sturen, zonder dat we ons druk hoeven te maken over het type device en wat voor software erop draait. Dit zorgt ervoor dat we als ontwikkelaars onze aandacht volledig kunnen richten op de functionaliteit van onze applicaties. In dit artikel hebben we slechts een deel van de mogelijkheden van de verschillende technologieën laten zien. We adviseren iedereen die in deze technologie geïnteresseerd is om het Internet op te gaan voor meer informatie (zie ook onderstaande links) en er zelf mee aan de slag te gaan.

Nuttige links:

SignalR:

- http://www.hanselman.com/blog/AsynchronousScalableWebApplicationsWithRealtimePersistentLongrunningConnectionsWithSignalR.aspx
- http://stackoverflow.com/questions/tagged/signalr

Push notification hubs:

- http://msdn.microsoft.com/en-us/library/jj891130.aspx
- http://channel9.msdn.com/Blogs/Subscribe

Edwin van Wijk

Edwin van Wijk is sinds 1999 werkzaam in de IT en werkt als IT Architect bij Info Support, met als specialisme het Microsoft platform. Hij wordt hoofdzakelijk ingezet op projecten bij klanten voor advies op het gebied van architectuur en .NET development. Daarnaast is hij vaak betrokken bij de invoering van de Endeavour ontwikkelstraat van Info Support bij klanten. Edwin is trekker van de Windows Azure focusgroup binnen het Info Support competence center voor Microsoft Application Development. Zijn expertise ligt vooral op het gebied van: Architectuur, SOA, EAI, ESB, WCF, WF, en het Windows Azure platform.

Jeroen Hendricksen

Jeroen werkt als software ontwikkelaar en heeft zich gespecialiseerd in .NET. Momenteel is hij werkzaam voor klanten in de zorg- en overheidssector. Verder heeft hij een bovengemiddelde interesse voor alles dat met privacy en security te maken heeft.

Marcel Meijer

Windows Azure Free Trail

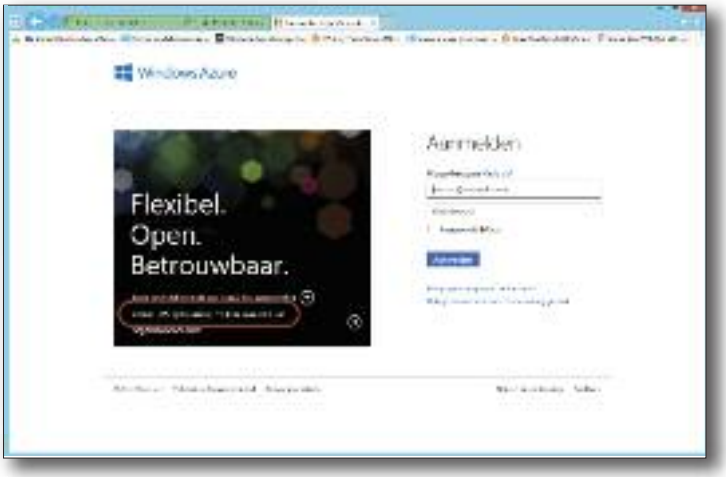
Omdat ik een Windows Azure account wilde koppelen aan mijn bestaande Office 365 account, heb ik een nieuw Windows Azure account gestart. In dit artikel neem ik jullie stap voor stap door het aanmeld proces heen. Je kunt zonder risico starten met Windows Azure, als je over het gratis gebruik heen gaat dan stopt het allemaal vanzelf.

De eerste stap die je moet nemen, ga naar: https://www.windowsazure.com/nl-nl/pricing/free-trial/.

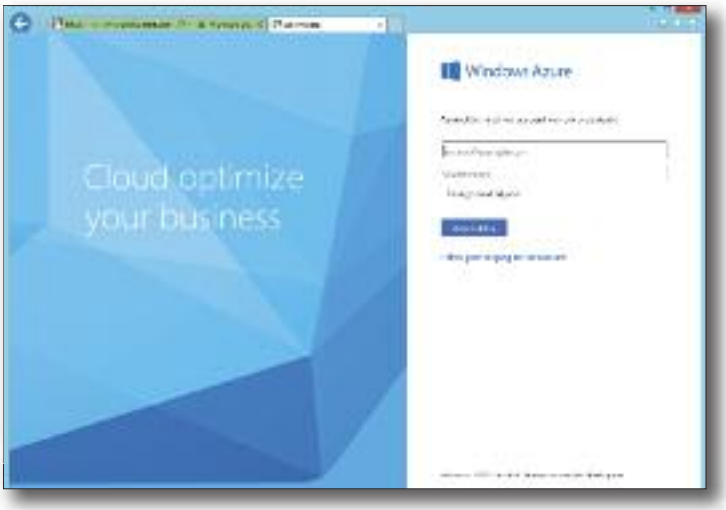
Als je op “nu kopen” klikt, dan krijg je dit scherm.

Ik ga even van Prive gebruik uit. Dat betekent, dat we eventueel later willen betalen voor wat we echt gebruiken en niet dat we vooraf capaciteit inkopen. Het inkopen van Windows Azure capaciteit kan erg interessant zijn voor bedrijven of startups, die weten hoeveel capaciteit ze minimaal verstoken. Je krijgt dan wel een korting. Het principe is gelijk aan wat Telecom aanbieders doen. Als je een bel-bundel koopt, dan zijn de tikken binnen je bel-bundel goedkoper dan er buiten.

Je moet je dan aanmelden met je Microsoft Account. Aangezien ik mijn Windows Azure Free Trial wilde koppelen aan mijn Office 365 account, kies ik daar niet voor. Het vervolg na deze stap is voor beide gelijk.



Ik krijg nu het inlog scherm van Office 365.



Als ik dan ingelogd heb met mijn Office 365 account, dan krijg ik het volgende te zien. Ik heb inderdaad nog geen abonnementen, dat is het doel van deze actie ☺. Maar als je al een Free Trail actief hebt, dan kun je er niet nog een aanvragen. Je zult dan of je Free Trail moeten stop zetten of subscription kopen.

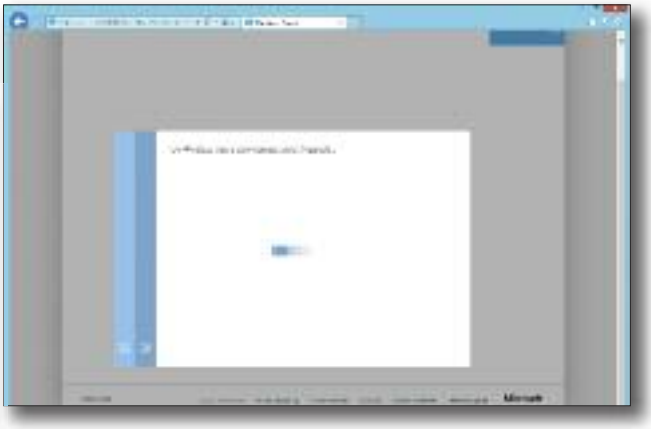
Na het klikken op de “Meld u aan” link, gaan we lekker verder.

We moeten een mobiel nummer opgeven. Daarmee wordt gecontroleerd dat er niet een vreemde robot of zo allerhande free trails gaat aanmaken.

Je krijgt dan een SMS met een code. Deze moet je op de site invoeren.

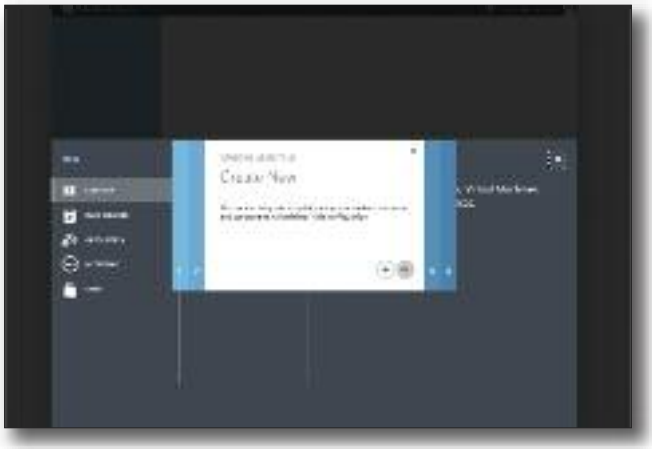
Daarna zullen we Credit card gegevens en adresgegevens moeten opvoeren. Via een faktuur betalen kan tegenwoordig ook, maar dan alleen als je kiest voor de bundel varianten.

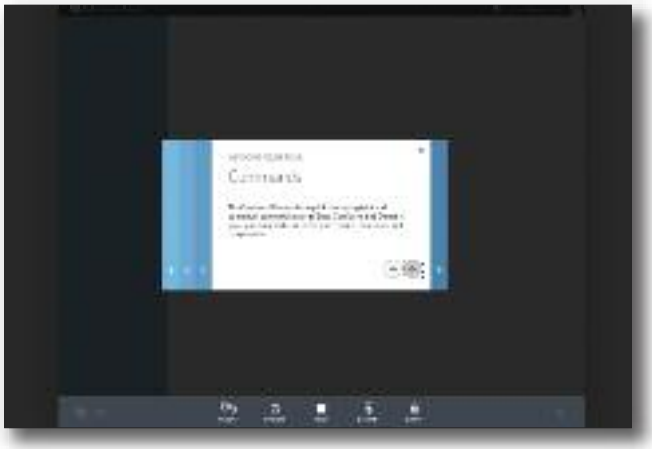
Na het klikken op volgende, wordt je subscription aangemaakt.

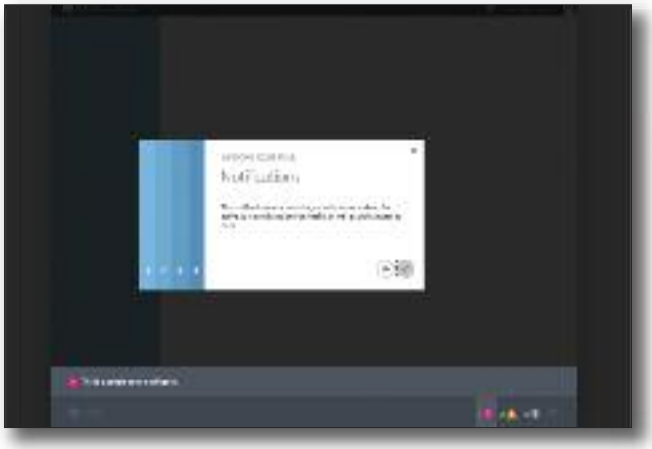


En je bent de gelukkige eigenaar van een mooie Windows Azure subscription. Anders gezegd je computerkracht en diskruimte is nu onbeperkt!

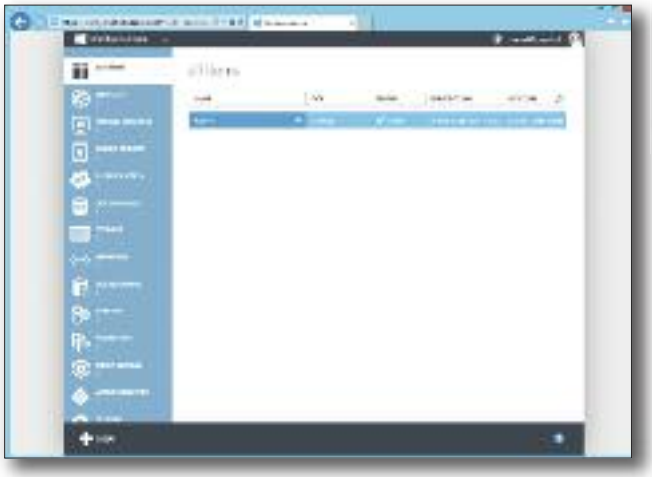
Als je dan op knop Rechtsboven (Portal) klikt, dan ga je naar de Windows Azure managment portal. Je doorloopt dan de welkoms wizard, die bekend maakt met alle vernuftige dingen in de portal.



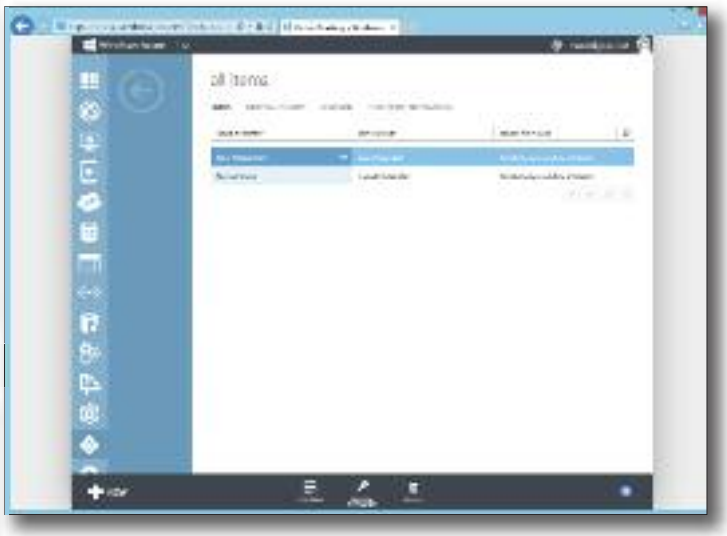




En we krijgen de Management portal te zien!

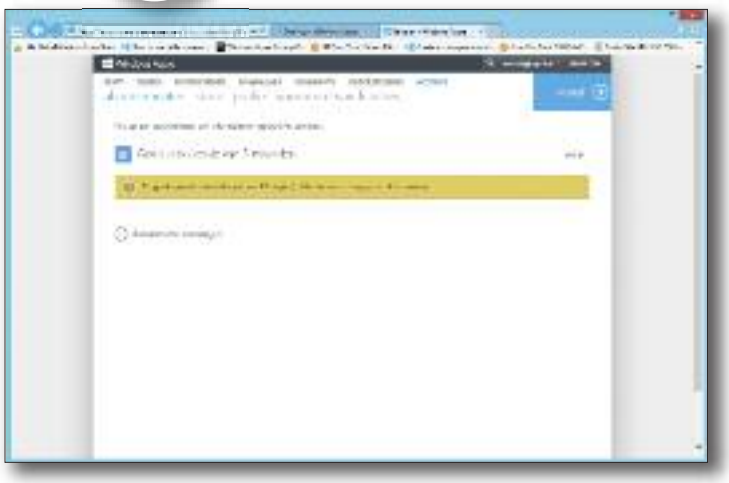


Zoals je ziet is de Active Directory van je Office 365 account gelijk toegevoegd. Met alle gebruikers van het Office 365 account.



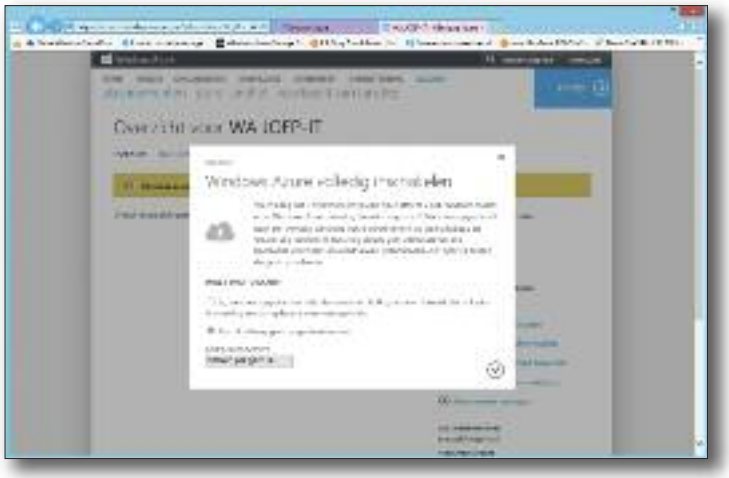
Starten met Windows Azure is FAST and EASY

Daarover later nog veel meer.
Als je nu naar het Billing menu gaat. Klik daarvoor op je e-mail adres bovenaan en kies voor “Show my bill”. Dan zie je dat je subscription een vreemde naam heeft. Deze kun je aanpassen.



Klik dan op de witte regel met “Gratis proefversie van 3 maanden”. Er verschijnt dan een overzicht van je verbruik en er staan een aantal opties. Een daarvan is “Uw abonnement bewerken”.

Mocht je niet meer van je subscription af willen (wat ik mij prima kan voorstellen ☺), dan klik je op gele regel.



Heel veel plezier met je Windows Azure subscription. Laat je gedachten de vrije loop en creëer mooie nieuwe toepassingen op het Windows Azure platform. ●

Marcel Meijer

Al meer dan 20 jaar begeeft deze 44-jarige zich in de wereld van de ICT. Op dit moment houdt hij zich voornamelijk bezig met Azure, Cloud, C#, Software Ontwikkeling, Architectuur in het algemeen en Windows Phone 7. Ook is hij bezig met Biztalk en Sharepoint. Hij werkt als Senior Solution Architect bij Prodware. In zijn vrije tijd is hij .NET track owner, eindredacteur en bestuurslid van de SDN. Binnen de SDN is hij verantwoordelijk voor het regelen van sprekers voor de SDN Events (SDE), het regelen/redigeren van artikelen voor het SDN Magazine, mede verantwoordelijk voor de eindredactie van de hardcopy en digitale magazines en de inhoud van de SDN Conferences. Op 1 oktober 2010 werd hij MVP.

Voor ons magazine zijn we altijd op zoek naar interessante artikelen. Wat heb je ontdekt, waar heb je een mening over, waar heb je ervaring mee opgedaan en waar wil je over vertellen?

Neem contact op met de redactie redactie@sdn.nl

NB: De deadline voor magazine 118 is op 29 juni a.s.

Sander Hoogendoorn

The changing interpretation of agile

For as long as I can remember I have been evangelizing, promoting, practicing, coaching, and training agile. For me as a developer the goals for applying agile approaches and techniques are pretty clear. I want to make better software. Higher quality, better suited for use, and possibly also faster. And from my own empirical evidence I can certainly state agile helps.

Gone with the wind

So you would say the current raging popularity of everything agile would bring greater joy to my personal life. Unfortunately it doesn't. Agile has become the new magical keyword for improving just about anything thinkable. There's an agile mindset. The word agile is dropped on a daily basis in advertisements. There's agile cars, agile games, and even agile washing powder. And on the fly agile seems to inspire people, communities and even movements to change the world. Great.

Yes, I know agile is characterized in dictionaries by quickness, lightness, and ease of movement or even nimble. So yes, I totally agree agile can be applied to much more than software development. But to quote Clark Gable in Gone With The Wind: Frankly, my dear, I don't give a damn. I don't care much for changing the world, or even changing organizations. I want to write software. Better. Faster.

Let me give you some examples where agile is going. At a recent agile conference delegates put up a board of interesting books. It contained highly agile books, such as Your Brain At Work, Tribal Leadership and Fearless Change. Without any doubt very interesting and worthwhile books. But what happened to the classics? I didn't spot Design Patterns, Framework Design Guidelines, Code Complete, Clean Code, or Patterns Of Enterprise Application Architecture?

These days, agile is used as a adapter for all kinds of coaching and personal improvement. From Getting Things Done through neuro-linguistic programming to gamification. There is even an agile game conference in the planning. Or what about personal energy management, product owner "light bulbs" or agile kitchens? All very nice, but very little to do with programming. As said, agile is becoming an accepted synonym for any form of personal development.

Half-full or half-empty?

And it's getting worse. In many organizations I've noticed that agile is both used to illustrate failure of things that have absolutely nothing to do with agile, as well as an explanation for things that go well. Also having nothing to do with agile. Let me also give some examples of this interesting use of the word agile.

First of all, agile seems to become synonymous for anything that is done just-in-time. For instance, some account manager, barely making the deadline for submitting a proposal to a client, commented on the process stating: "Great, we did this really agile," suggesting that delivering just-in-time is really agile. Nor the proposal, nor the way it was written had anything to do with agile whatsoever. At another occasion, two people showed up late for a meeting. The organizer of this meeting, which was neither a kick-off, a retrospective nor a stand-up meeting, just smiled and said: "Well, that's agile, isn't it?" Well, it isn't. So agile also has become a synonym for things are run over a deadline. The glass can be half-full or half-empty, depending how you look at it.

I suppose it is due to the liberal definition of agile that the word, and in its slipstream words such as Scrum, Kanban and Lean have become daily speech patterns. I am quite aware that we as software developers and testers have no right to claim the word agile for ourselves. And I for one certainly wouldn't want to deprive anyone from having fun with it. Everybody is entitled to personal development, or to chasing his or her dream to change the world or even an organization, but sometimes I wish we could just go back to what we developers and testers do best: write code and test functionality. ●

Bob Swart

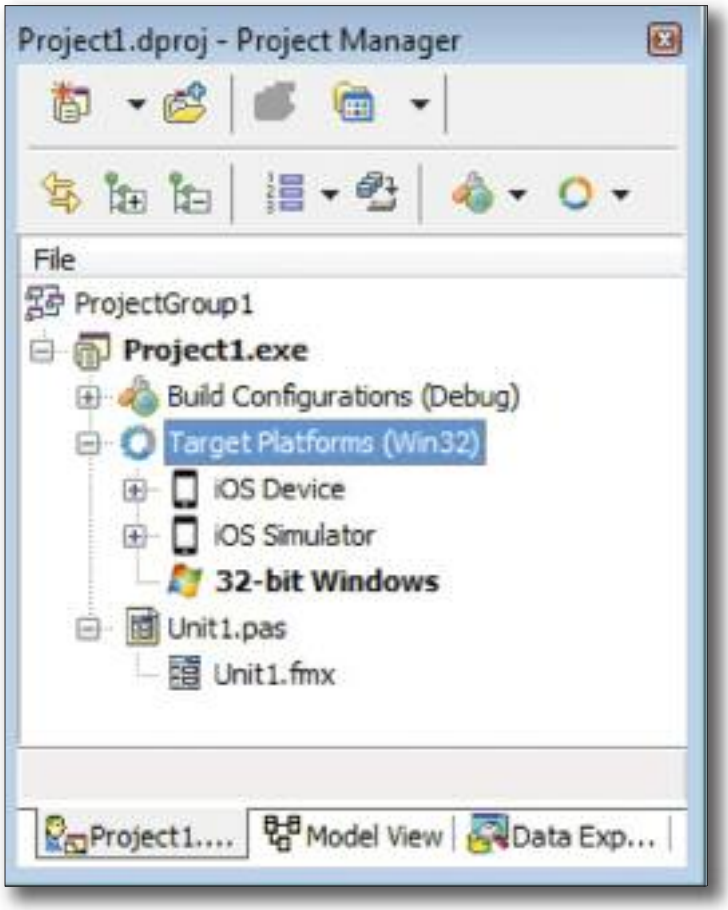
Delphi XE4 Mobile Strings

Op het moment van schrijven is het nog in beta (dus enkele zaken die ik nu opschrijf kunnen veranderd zijn), maar tegen de tijd dat jullie het SDN Magazine lezen zou Delphi XE4 beschikbaar moeten zijn. Het is weer een hele speciale versie: er zit nu namelijk een nieuwe compiler in. Delphi is niet langer alleen de snelle desktop compiler, maar er is nu ook een speciale mobile compiler. En dat heeft consequenties, met name voor Strings...

Delphi XE4 Mobile Compilers

OK, eigenlijk zijn het er twee: er zijn twee nieuwe mobile compilers die bij Delphi XE4 zitten. Nee, niet voor Android. Maar voor iOS. De ene is voor de iOS Simulator (die draait nog steeds onder OS X op Intel hardware), en de andere is voor het echte iOS device (en dat is ARM hardware). Beide compilers zijn gebaseerd op de LLVM technologie, wat helaas wel betekent dat ze een stuk langzamer zijn dan de desktop compiler. Door van LLVM gebruik te maken is er maar één front-end nodig, en verschillende LLVM back-ends voor de verschillende targets (zoals iOS Simulator op Intel, en iOS op ARM).

In onderstaande screenshot zijn drie verschillende targets te zien: iOS Device, iOS Simulator en 32-bit Windows, die dus in feite alle drie een verschillende compiler gebruiken voor verschillende binary output.



De back-ends vind ik zelf niet zo boeiend, maar de front-end is dat wel, want er zijn een aantal kleine en minder kleine wijzigingen in de Delphi Mobile Compilers vergeleken met de Delphi Desktop compiler...

Strings

Iedereen die langer dan een paar weken met Delphi heeft gewerkt, weet dat Delphi een aantal verschillende string types kent: vanaf Delphi 1 hebben we al de ShortString (met AnsiChars in, maximaal 255 tekens lang, en de lengte staat in het 0e byte). Daarnaast hebben we de AnsiString (ook met AnsiChars, maar kunnen veel langer worden), en de String (is sinds Delphi 2009 een Unicode String met WideChars). En nog de WideString voor communicatie met COM (en managed door Windows, dus erg langzaam vergeleken met een gewone Unicode String).

Al deze strings hebben één ding gemeen: het eerste teken staat op positie 1. En dat terwijl voor alle andere (dynamische) array types het eerste element op positie 0 begint. Net als bij alle andere talen die ik ken: daar begint een string ook altijd op positie 0.

Mobile Strings

Met de nieuwe Delphi Mobile Compilers wil Embarcadero ook een nieuwe "lichting" van Delphi ontwikkelaars aantrekken. Maar liefst zonder die te vermoeien met een half dozijn verschillende strings die te pas en te onpas gebruikt kunnen worden.

Dus ondersteunen de Delphi Mobile Compilers nog maar één String, met Unicode tekens erin, en het eerste teken staat op... positie 0.

Er gaan zelfs geruchten dat op termijn deze string "immutable" zou worden; oftewel: read-only, en niet aan te passen. Dus dan zou een stukje code als volgt niet meer werken:

```
var
  S: String;
begin
  S := 'Delphi';
  S := S + ' Development Network?';
  S[Length(S)-1] := '!';
  ShowMessage(S);
```

Als je deze code compileert met Delphi XE4 kun je inderdaad een warning krijgen:

```
W1068 Modifying strings in place may not be supported
in the future
```

Overigens is bovenstaande code niet eens echt cross-platform, want hij wijzigt het teken op positie Lengte min één. Dat is voor een string die op positie 0 begint correct, en daar zal het vraagteken (laatste teken) vervangen worden door een uitroepteken. Maar voor de "oude" Delphi Desktop compiler zal het een ander effect hebben: daar zal de "k" worden aangepast, waardoor de string niet echt meer correct is.

$IFDEF NEXTGEN

Dit is een potentieel probleem dat we op verschillende manieren kunnen oplossen. Aangenomen natuurlijk dat we dezelfde code onder zowel Windows en/of Mac OS X (met de Desktop compiler) als voor Mobile Devices (met de Mobile compiler) willen laten werken.

Delphi Desktop Compiler kan strings als 0-based strings behandelen

Een oplossing die voor de hand ligt is een IFDEF. Voor de nieuwe compiler kunnen we bijvoorbeeld {$IFDEF NEXTGEN} gebruiken, als volgt:

```
var
  S: String;
begin
  S := 'Delphi';
  S := S + ' Development Network?';
  {$IFDEF NEXTGEN}
  S[Length(S)-1] := '!';
  {$ELSE}
  S[Length(S)] := '!';
  {$ENDIF}
  ShowMessage(S);
```

Dit heeft inderdaad het gewenste effect, maar betekent helaas ook dat we dan stukken code die echt met strings weken in twee delen moeten schrijven (of in tweeën moeten knippen). Dat is natuurlijk niet echt handig. Om niet te zeggen: gewoon onhandig, en gevoelig voor fouten. Op die manier kun je de Win32 versie van de toepassing niet echt gebruiken als test (omdat je de andere code voor het mobiele platform dan juist niet aan het werk ziet).

$ZEROBASEDSTRINGS ON

Gelukkig is er nog een alternatieve mogelijkheid: het is namelijk mogelijk om de Delphi Desktop Compiler te vertellen dat deze strings ook als 0-based strings moet behandelen. Dat is funest voor de meeste bestaande code, maar wel ideaal voor code die we vanaf nu schrijven (voor cross-platform gebruik).
Het betreft de ZEROBASEDSTRINGS compiler directive, die we op ON of OFF kunnen zetten. Voor bovenstaande code, kunnen we deze single source compatible maken met zowel Win32 als Mobile targets door overal 0-based strings aan te zetten:

```
{$ZEROBASEDSTRINGS ON}
var
  S: String;
begin
  S := 'Delphi';
  S := S + ' Development Network?';
  S[Length(S)-1] := '!';
  ShowMessage(S);
```

Deze code zal nu zowel onder Win32 als de iOS Simulator en het echte iOS Device hetzelfde werken.

Wacht eens even, hoor ik de oplettende lezer al roepen: maar als we de compiler kunnen vertellen om gewone strings als 0-based strings te zien, dan kan dat ook andersom. Dus door {$ZEROBASED-STRINGS OFF} te zetten. Ja, in theorie kan dat. Maar het wordt afgeraden om dat te doen.

Pos vs. IndexOf

Wat natuurlijk gevaarlijk lijkt, is het gebruik van bestaande code die bijvoorbeeld nog de Pos aanroept. Pos levert altijd de positie op van een teken op basis van het feit dat het een 1-based string zou zijn. Zelfs als je ZEROBASEDSTRINGS ON hebt staan. Dus de volgende code levert "6" op, voor de positie van de "i" in "Delphi". En dat onafhankelijk van de waarde van ZEROBASEDSTRINGS:

```
var
  S: String;
begin
  S := 'Delphi';
  ShowMessage(IntToStr(Pos('i',S)));
```

Het gevaar is echter wel, dat we de daadwerkelijke positie niet moeten gebruiken in combinatie met de [en] array haken als ZEROBASEDSTRINGS op ON staat. Dan gaat het mis.
Bij gebruik van ZEROBASEDSTRINGS ON moeten we IndexOf gebruiken. Deze gaat namelijk uit van 0-based strings, en zal voor de "i" de positie 5 teruggeven...

```
var
  S: String;
begin
  S := 'Delphi';
  ShowMessage(IntToStr(Pos('i',S)));
  ShowMessage(IntToStr(S.IndexOf('i')));
```

Wederom onafhankelijk van de waarde van ZEROBASEDSTRINGS. Het enige dat wel effect heeft zijn de [en] haken. Daar moet je dus uitgaan van een start bij 1 of 0 afhankelijk van de compiler en de waarde van ZEROBASEDSTRINGS.

Er is nog een andere truck die we kunnen gebruiken: de waarde van Low op een 1-based string is 1, en op een 0-based string natuurlijk 0. Dus als Low(S) gebruiken kunnen we daar S.IndexOf bij optellen (of van Pos(S) de waarde van 1-Low(S) aftrekken). Op die manier zou je code kunnen schrijven die onafhankelijk is van het gebruik van de ZEROBASEDSTRINGS compiler directive.

TStringBuilder

Zoals bekend zit er ook een TStringBuilder class in Delphi, en deze werkt op een vergelijkbare manier als in .NET bijvoorbeeld. Er wordt vanuit de Delphi documentatie aangeraden om vooral naar de TString-Builder te gaan als je bestaande code (voor 1-based strings) wilt omzetten, of als je veel kleine stukjes string wilt samenvoegen.
De TStringBuilder class is zeer efficiënt, alhoewel je onder Windows en de iOS Simulator weinig performance verschil zal merken. Voor het uitvoeren van string operaties op een echt iOS device geldt dat een TStringBuilder ongeveer twee keer zo snel is. Wel moet ik daar meteen bij zeggen dat operaties op een echt iOS device ongeveer 10 tot 100 keer langzamer kunnen (en zullen) zijn dan op de gemiddelde Windows of Mac OS X computer (met de iOS Simulator), dus pas op als je tests gaat doen: de performance op een echt iOS device kan vies tegenvallen.

Conclusie

Ik hoop dat ik in dit korte artikeltje een beetje een overzicht heb gegeven van de nieuwe "Mobile" Strings die in Delphi XE4 zullen zitten voor gebruik met de nieuwe Delphi Mobile Compilers - zowel voor de iOS Simulator als de iOS Devices. Het verdient de aanbeveling om weinig met IFDEFs te werken, vooral voor wie geen Mac beschikbaar heeft om meteen alle code te testen (en om voor het hele ontwikkelteam nu iedereen een Mac te geven alleen om te testen lijkt me ook wat ver gaan). Met de ZEROBASEDSTRINGS komen we een heel eind, en ik vraag me sowieso af of er heel veel bestaande code zal zijn die we in een mobile toepassing willen hergebruiken, dus wellicht is het toch beter om dan al vanaf het begin af aan met 0-based strings aan de slag te gaan.

Delphi Mobile Compilers trekt een nieuwe "lichting" van Delphi ontwikkelaars aan

Tijdens de Delphi Developer Days 2013 (eind mei in Frankfurt en begin juni in Amsterdam) zullen Cary Jensen en ik nog dieper op dit onderwerp ingaan. Zie http://www.DelphiDeveloperDays.com voor meer details. Wellicht tot ziens! •

Bob Swart

Bob Swart is werkzaam in de IT sinds 1983 en heeft daarbij een voorliefde voor (Turbo) Pascal en Delphi. Bob spreekt regelmatig op (internationale) conferenties over Delphi en heeft honderden artikelen geschreven, alsmede zijn eigen Delphi cursusmateriaal voor Delphi trainingen en workshops. Behalve voor het geven van trainingen, is Bob ook beschikbaar voor consultancy, coaching, ontwerp- en bouwwerkzaamheden, of andere ondersteuning op het gebied van software ontwikkeling met Delphi – voor zowel Win32 als .NET. Sinds de zomer van 2007 is Bob ook reseller van CodeGear producten zoals Delphi en RAD Studio. Bob Swart Training & Consultancy is gevestigd in Helmond Brandevoort, en beschikt over een eigen trainingsruimte van ruim 42 vierkante meter, inclusief een testlab voor alle mogelijke toepassingen. De voorliefde van Bob voor Pascal en Delphi heeft hij ook tot uiting laten komen in de namen van zijn kinderen Erik Mark Pascal en Natasha Louise Delphine.

SDN > Update

Global Windows Azure Bootcamp NL

Op zaterdag 27 april vond wereldwijd het Global Windows Azure Bootcamp event plaats. In Nederland werd deze door de WAZUG en SDN georganiseerd.

Het programma in Nederland bestond uit een Windows Azure introduction door Marcel Meijer en na de lunch een sessie over HD Insight door Dennis Mulder. Tussendoor was er tijd om zelf aan de slag te gaan met de Hands-on labs van de Windows Azure trainingskit. Patriek van Dorp, Dennis Mulder, Edwin van Wijk en Marcel Meijer waren in de buurt om bij eventuele vragen of problemen te helpen.

Aan het einde van het event werd er wereldwijd een heel mooi experiment uitgevoerd. Alle landen deden mee met het Kinect en Windows Azure WorkerRoles van Alan Smith. Nederland deed mooi mee! In 21 landen werd mee gedaan aan de Render Farm voor een periode van 24 uur. In die 24 uur werden **724059 frames** gerenderd door **9793 WorkerRole instances**.

Dat had anders 4.5 jaar geduurd. Normaal gesproken had dit $249.955,- gekost en nu koste dat bijna niks, bijna iedereen gebruikte zijn MSDN vrije compute uren of een gratis Windows Azure trail.

Het was weer een perfect event en we danken de lokale sponsoren (Info Support, Ineta en Prodware) voor hun support.

Marcel Meijer

Windows Azure Active Directory (WAAD)

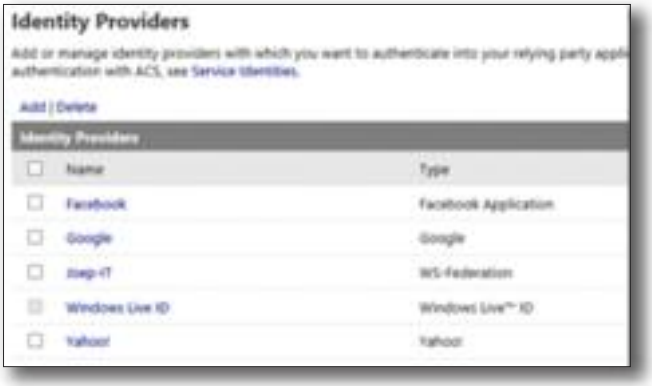
Het grote nadeel van Webapplicaties in de Cloud ten opzichte van Intranet applicaties is de beschikbaarheid van Active directory. Een van de oplossingen is natuurlijk om gebruik te maken van social networks (zoals Google, FaceBook, Twitter, Yahoo, Windows Live) om de gebruiker te valideren.

Maar dan weet je wel dat degene die zich aanmeldt degene is die hij zegt dat ie is, maar autorisatie moet je nog steeds zelf beheren. Dat is lastig, want zo heb je nog steeds geen echte controle. Zeker als je gebruikers medewerkers van je bedrijf zijn, dan wil je dat ze inloggen met een corporate account natuurlijk.
Maar als modern bedrijf maak je natuurlijk gebruik van Office 365 (Microsoft's SAAS oplossing voor Office, Exchange en SharePoint in de Cloud). De medewerkers hebben dan een account en zou het niet mooi zijn om dat account te kunnen gebruiken in je Webapplicatie.

Op de Windows Azure portal hebben we al een Active Directory menu item. Daarachter zit het reeds bekende Windows Azure Access Control. Via dit mechanisme kun je via de bekende social networks (Google, FaceBook, Yahoo en Windows Live) mensen authentiseren. Meer info heb ik al eerder beschreven op mijn blog: http://blog.marcelmeijer.net/2011/07/06/windows-azure-appfabric-acs-met-meerdere-instanties/ en http://blog.marcelmeijer.net/2012/05/04/windows- azure-wif-access-control-acs/.

Mijn eigen test site http://cloudtest.marcelmeijer.net maakt hier gebruik van.

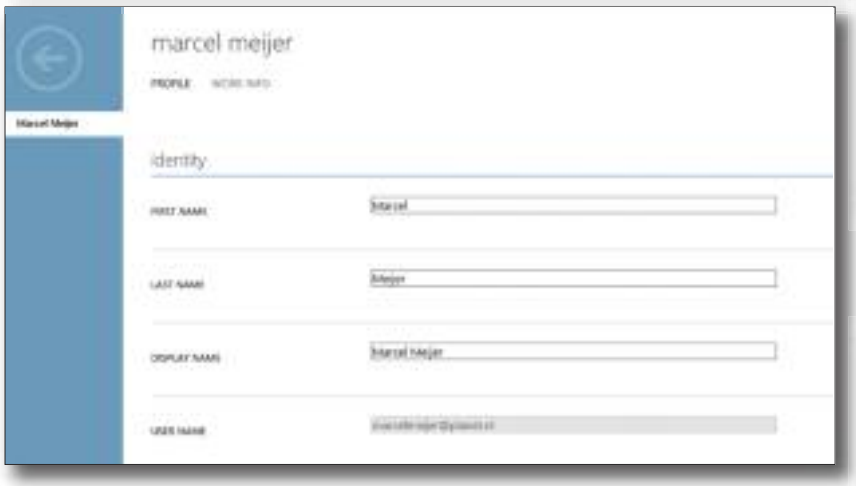
Maar waar deze site ook gebruik van maakt, is Office 365 als authenticatie provider. Met mijn Office 365 account op het Joep-IT domein kan ik inloggen op de site.



Via het Claims mechanisme van ACS krijgen we dan enkele gegevens terug. Die kunnen we dan weer gebruiken in onze applicaties etc.

Dit klinkt allemaal wel goed, maar dat is nog steeds niet het echte Active directory. Bij Active directory willen we users maken en deze users gegevens en rollen geven. Dat willen we dan gebruiken.

Sinds kort is een op Office 365 gebaseerde Active Directory beschikbaar gekomen. We kunnen een Directory maken, op dit moment alleen nog op een nieuwe <name>.onmicrosoft.com Office 365 account. Het is de bedoeling dat ik hier later ook mijn bestaande Joep-IT Office 365 account kan gebruiken.



En via de SDK kun je graph en dus de gegevens van deze Active directory opvragen en gebruiken. Super!

En ik kan gebruikers toevoegen.

De gebruiker krijgt dan een mailtje met een tijdelijk wachtwoord.

Maar wat als je nu een on-premise Active directory hebt, moet je dan dingen dubbel doen? Voor ACS hebben we al AD-FS (Active Directory Federation Services). Daarmee kun je de gebruikers van je lokale AD naar de Cloud halen.

Deze oplossing is niet helemaal optimaal. De 'nieuwe' Active directory biedtje mogelijkheden om je on-premise AD te syncen met je Cloud AD.

Aan de Cloud AD kun je dan applicaties toevoegen.

Dit is helemaal top. Ik kom later terug met een meer uitgewerkt voorbeeld van de werkelijke toepassing. ●

Herre Kuijpers

Extending ASP.Net Controls: a Scrollable GridView

Although ASP.Net comes with a large set of customizable web controls, you may wish in some cases to extend the functionality and behavior of a web control. Fortunately ASP.Net controls have various extension points:

1. Customizable properties
2. Style and css-classes
3. Templating
4. Overriding

Although the first 3 options can be configured at runtime, the 4th option will require some coding. Overriding is required when you wish to change the control's behavior or layout. In such cases it is typical to create a subclass for the web control in which the overridden methods are implemented.

Fortunately ASP.Net controls have various extension points

This article presents an example for extending the standard ASP.NET GridView control with the option to display scrollbars (a.k.a the scrollable table with fixed header problem), instead of using the standard 'paging', allowing to display all list items at once. This solution is tested with Internet explorer, Firefox and Chrome.

Paging and scrollbars

Web standards prescribe 'paging' functionality when browsing through large lists. The idea is that large lists will take a 'longer' time to download, so when using paging users only need to download a small set of rows just enough to be displayed on a single page.

The standard ASP.Net GridView supports paging by default. Strangely enough the GridView does not optimize loading time by much, since the list that is bound to the gridview is stored in the ViewState by default, and therefore still generates a 'large' page resulting a in longer download time.

So one might as well just choose to ignore paging and show the complete list as a whole. This has the advantage that the page does not need to be reloaded each time the user chooses a different page in the grid.

Obviously the list can become too large to fit on a single webpage, and hence using scrollbars would be a sufficient solution. Something that is quite common in desktop applications. I found however that adding a scrollbar to the GridView is not entirely trivial: the goal is to only have the list items scrollable, while leaving the gridview header fixed.

I've looked for solutions for this issue, however some of the proposed solutions are only compatible for some browsers (e.g. CSS based), others were quite complicated or worked only in certain situations. As far as I've seen no standard solution is available for the GridView control. The solution presented here is compatible with various common browsers and can be incorporated into any ASP.Net solution with ease. Also the final rendered HTML is as compact as possible.

I found however that adding a scrollbar to the GridView is not entirely trivial

Enabling 'scrollability'

The approach to the solution is not new. Basically the header row is displayed in a table in a header DIV, while the body rows are displayed in a separate scrollable body DIV element. This approach poses the following issues to solve.

Extracting the header from the body

By default the GridView renders the first row as the header row in the table. The trick is to extract the first row and render it as a separate table. This requires a bit of tinkering to get it working. It requires an override of the default RenderChildren method of the GridView control. The RenderChildren method is responsible for generating the output which is written to the response object. Instead of generating the

default GridView html element, I rearranged and inserted some html elements of my own.

The GridView can be broken down and build back up again in 10 small steps:

1. Create a new header DIV html element. Assign it a special css-class so it can be accessed from JQuery. Set the width of container DIV as specified for the gridview.
2. Create a new Table object and insert it into the header DIV. Make sure the correct styling is applied. Assign it a unique id for reference from javascript. Maximize the size of the inner header table to 100% for a tight fit within the DIV.
3. Select the header row in the GridView and insert it into the the newly created table. Note that this will automatically remove the row from the original table!
4. Render the header DIV (this will automatically also render its children elements).
5. Create the body DIV.
6. Set the style attributes of the DIV and make sure it is scrollable vertically, set the width and height explicitly for the scrollable body DIV.
7. Insert the header row back into the original table at position 0 (at the top).
8. Insert the GridView table inside the body DIV and render the body DIV and render it.
9. Insert the table back into the GridView. This is an essential step, as it will restore the GridView into its original state. Forgetting to do so will crash your application.
10. Render any other remaining controls contained in the GridView (e.g. the footer)

```
// step 1
HtmlGenericControl divhead = new HtmlGenericControl("DIV");
divhead.Attributes["class"]    =    "GridViewHeaderContainer";
divhead.Style.Add("overflow", "hidden");
if (this.Width != Unit.Empty)
        divhead.Style.Add("width", this.Width.ToString());

// step 2
Table tablehead = new Table();
tablehead.ApplyStyle(this.HeaderStyle);
tablehead.Style.Add("width", "100%");
tablehead.ID = ctrl.UniqueID + "$Header";
divhead.Controls.Add(tablehead);

// step 3
WebControl ctrl = (WebControl)this.Controls[0];
Control ctrlrow = ctrl.Controls[0];
tablehead.Controls.Add(ctrlrow);

// step 4
divhead.RenderControl(writer);

// step 5
HtmlGenericControl divbody = new HtmlGenericControl("DIV");

// step 6
divbody.Attributes["class"] = "GridViewBodyContainer";
if (this.Width != Unit.Empty)
    divbody.Style.Add("width", this.Width.ToString());
if (this.Height != Unit.Empty)
    divbody.Style.Add("height", this.Height.ToString());
divbody.Style.Add("margin", "0px");
divbody.Style.Add("overflow", "hidden");
divbody.Style.Add("overflow-y", "auto");

// step 7
ctrl.Controls.AddAt(0, ctrlrow);

// step 8
divbody.Controls.AddAt(0, ctrl);
divbody.RenderControl(writer);

// step 9
this.Controls.AddAt(0, ctrl);

// step 10
for (int i = 1; i < this.Controls.Count; i++)
{
    ctrl = (WebControl)this.Controls[i];
    ctrl.RenderControl(writer);
}
```

Listing 1: Extracting the header from the body in the RenderChildren method

Aligning the columns of the header row with the columns in the body table.

The problem with using 2 tables, one for displaying the header row and the other for displaying the body, is that the column sizes cannot be aligned easily. Apparently the sizing of the columns is defined by the browser, and may different per browser version.

The problem with using 2 tables is column sizes cannot be aligned easily

To make sure the columns of both tables align, some javascript/JQuery is used. The script contains a function called UpdateGrid() which is called after the gridview has completely rendered. At this time the column widths are known and re-aligned at runtime. It will look up the DIV's based on their css-class and loop through all rows and columns to determine the width. Then it will explicitly set the width for each table cell.

Once finished, it will make the first row in the body (which is the original header row) invisible. Because this is done after rendering in the browser, sometimes this may result in the display of 2 headers for a split second. That's the only drawback from this approach.

```
$(document).ready(function () {
    UpdateGrid();
});

function UpdateGrid() {
  for (var gid = 0; gid <
$(".GridViewHeaderContainer").length; gid++) {
     var headerdiv = $(".GridViewHeaderContainer")[gid];
     var bodydiv = $(".GridViewBodyContainer")[gid];
     var headerrow = headerdiv.children[0].children[0].chil-
dren[0];

     var tbody = bodydiv.children[0].children[0];
     var bodyheaderrow = tbody.children[0];

     $(headerdiv).height($(tbody.children[0]).height());
     for (var i = 1; i < tbody.children.length; i++) {
```

```
            var tr = tbody.children[i];
            for (var c = 0; c < tr.children.length; c++) {
                var td = tr.children[c];

 $(td).width($(bodyheaderrow.children[c]).width());
            }
        }
        for (var c = 0; c < headerrow.children.length; c++) {
            var td = headerrow.children[c];
            $(td).width($(bodyheaderrow.children[c]).width() -
1);
        }
        $(bodyheaderrow).css("display", "none");
    }
}
```

Listing 2: javascript function to synchronize table columns using JQuery

Using the grid in an UpdatePanel

In order to support updates of the Grid from within an UpdatePanel, one extra bit of javascript is required to make sure the Grid refreshes and realigns properly after each (partial) postback to the server. Every time the UpdatePanel updates its content it must call the UpdateGrid() javascript function. To do this the following handler must be declared in the webpage right after the ScriptManager declaration:

```
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<script type="text/javascript">
    Sys.Application.add_load(function () { UpdateGrid(); });
</script>
```

Listing 3: Synchronizing the columns right after a partial postback

Conclusion

Once the code-behind and javascript is plugged into the webpage, the GridView becomes scrollable!
Although .Net web controls allow for a high degree of customization including changing their behavior by overriding the RenderChildren method, it still requires quite some analysis and tinkering to find out the internals of the webcontrol and its sub-controls and how they are structured.

You are allowed to break this structure apart and render it in a different order, including adding your own elements. However remember to always restore the web control's internals to their original structure and state after rendering, otherwise the control will break and crash your application.

The source code from this article can be downloaded from: http://www.codeproject.com/Articles/573044/Extending-ASP-Net-Controls-a-Scrollable-GridView

Herre Kuijpers

Herre Kuijpers works as solution architect at Capgemini focused on applying Microsoft technologies.

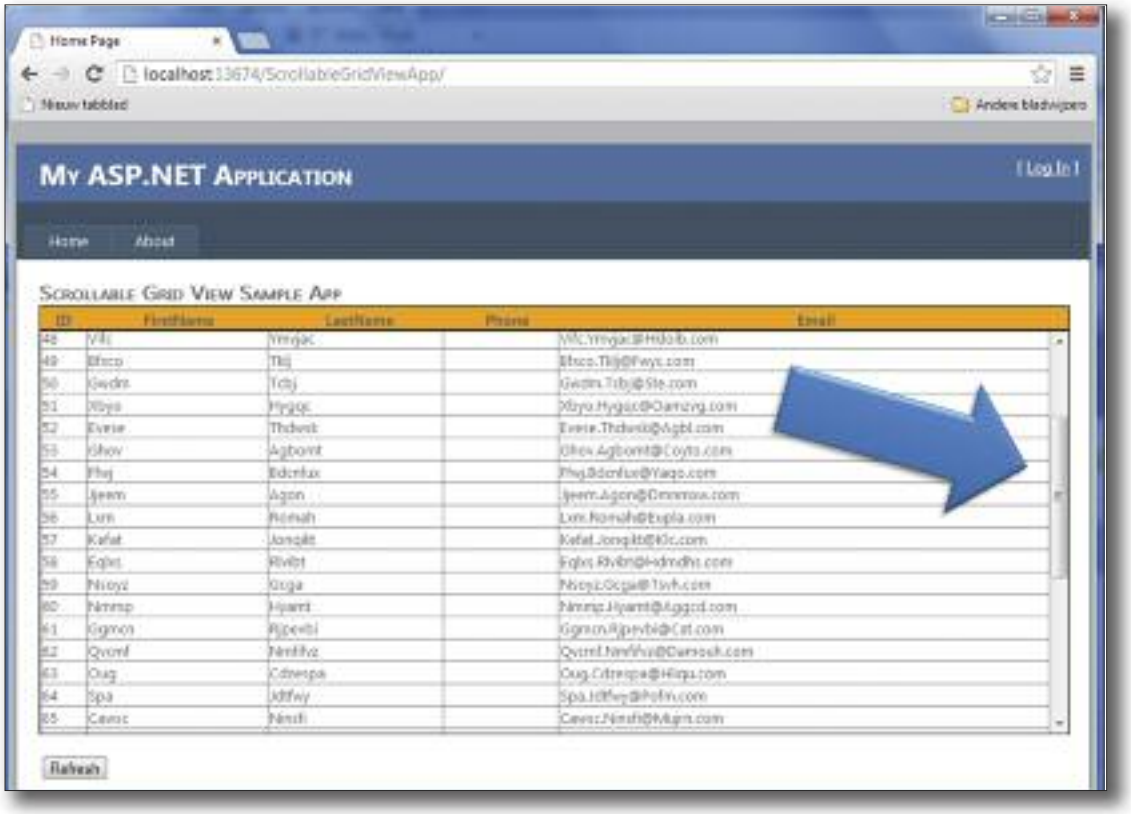


Figure 1: a Scrollable GridView with fixed header!

Dennis Vroegop

SCRUM: het blijft mensenwerk

Scrum. Het wonderwoord. De oplossing voor alle problemen met planningen, deadlines, glijdende specificaties en andere ICT project ellende. Toch?

Ik neem even aan dat je al wat langer in het vak zit. Als dat zo is weet je dat Scrum niet de 'silver bullet' is die mensen als Jeff Sutherland ons willen doen geloven. Ja, met Scrum kun je veel problemen voorkomen maar zoals altijd met oplossingen heeft ook deze oplossing als nadeel dat het zijn eigen reeks problemen heeft.

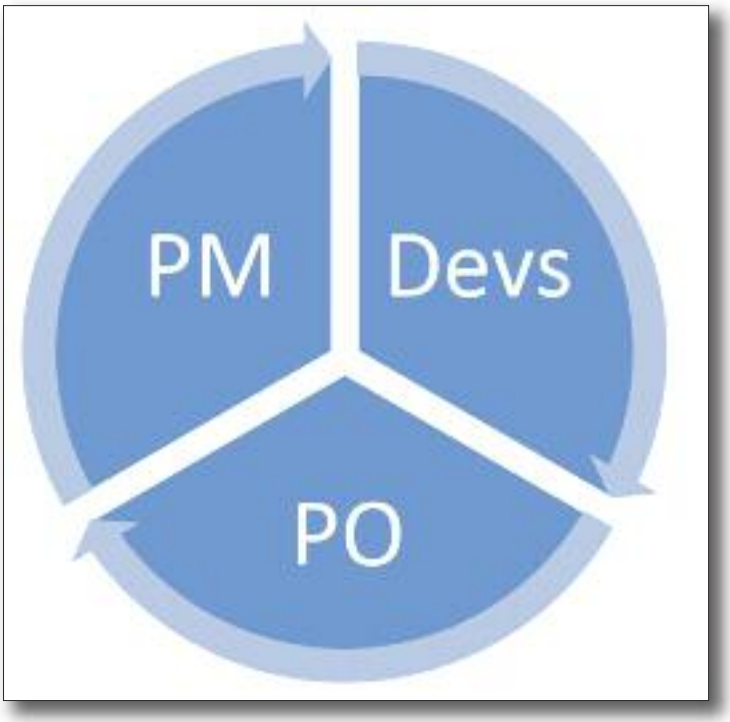
Eén van de grootste problemen met Scrum in mijn ogen is dat het een ogenschijnlijke technische oplossing is voor een mensenprobleem. Maar voor ik uitleg wat ik daar nu mee bedoel, wil ik eerst nog een keer kort uitleggen wat scrum inhoudt. Ondanks alle boeken, seminars, trainingen en nog meer is het eigenlijk heel simpel.

Scrum gaat uit van iteraties (sprints genaamd). Aan het eind van iedere iteratie hebben we een werkend geheel, iets wat we op kunnen leveren. Daarmee is één probleem van de klassieke methodes opgelost: de vraag van wanneer het product af is: na iedere iteratie is het product 'af'. Het is weliswaar niet compleet maar dat is een product nooit. Het is aan de opdrachtgeven om te bepalen wanneer een product af genoeg is om het project te stoppen.
Alles wat je nog meer denkt te weten over scrum is afgeleid van bovenstaande. Alle andere "regels" en procedures zijn er om bovenstaande mogelijk te maken. Daily Standups, backlog meetings, retrospective, sprint planning meetings: allemaal hulpmiddelen om de iteratie zo soepel mogelijk te laten verlopen met het doel aan het einde ervan iets werkends te hebben.

Cargo Cult Syndroom
Ik krijg wel eens de vraag: "Wij doen geen backlog meeting/daily standup/ enzovoorts. Doen wij nu scrum?" Het antwoord bestaat uit 2 delen: als je in iteraties werkt, doe je scrum. Het tweede deel van het antwoord is uiteraard: "Wat maakt het uit?" Mensen die op die manier denken hebben last van het Cargo Cult Syndroom.

Het Cargo Cult Syndroom is een fenomeen dat stamt uit de tweede wereldoorlog. In de tijd dat de Verenigde Staten bezig waren met hun opmars naar Japan, hadden ze bases nodig op de diverse eilandjes in de Stille Oceaan. Op een aantal van die eilandjes was nog nooit een westerling geweest. Toen de Amerikanen daar kwamen met hun boten en vliegtuigen en de plaatselijke bevolking chocola en sigaretten aanboden kwamen ze over als halfgoden. Na een aantal jaar kwamen er antropologen naar die eilanden en vonden daar dummy vliegtuigen, landingsbanen en verkeerstoren van palmbomen en bamboe. De plaatselijke bevolking had het idee dat als ze zich gedroegen zoals ze de goden hadden zien doen, de chocola vanzelf weer kwam. Een kwestie van oorzaak en gevolg door elkaar houden. Ik kan me voorstellen dat je nu met een glimlach op je gezicht zit en denkt "die inboorlingen ook". Maar op de een of andere manier doen we dit allemaal. We houden daily standup meetings en retrospectives in de hoop dat al onze software development problemen vanzelf oplossen. Dat doen ze dus niet.
Ik wil niet zeggen dat al die procedures geen nut hebben. Dat hebben ze wel degelijk. Maar ik wil wel dat je ze ziet voor wat zijn: ondersteunde optionele processen die het doel helpen bereiken. Met de nadruk op ondersteunend.

Het Team
De kern van scrum is het team. Daar gaat het om. Het team is wat uiteindelijk de software maakt. Het team is verantwoordelijk voor het resultaat. Het team krijgt de schuld als het goed gaat maar mag ook de eer opeisen als het iets moois oplevert.
Het team is op te splitsen in 3 onderdelen: ik heb ze weergegeven in figuur 1.



Figuur 1

De PM is de Program Manager, beter bekend onder de naam Scrum Master. De PM is verantwoordelijk voor het proces. Deze zorgt er voor dat aan alle randvoorwaarden is voldaan zodat het team optimaal kan

werken. Het team hoeft zich niet bezig te houden met het inplannen van meetings, het opschonen van de backlog en andere taken die afleiden van waar het omgaat: het opleveren van software. Even voor alle duidelijkheid: de Scrum Master is dus niet een ontwikkelaar die "het er even bij doet". Het bijhouden van het proces en het regelen van alle zaken rondom het team is een fulltime bezigheid die alle aandacht verdient.

De PO is de Product Owner. Denk aan: de klant. Dit is de enige persoon die beslissingen kan en mag nemen over de functionaliteit. Deze bepaalt de prioriteit van de taken en bepaalt hoe alle zaken moeten werken of er zien. Degene die betaalt is immers degene die bepaalt.

De Devs zijn de mensen die de software bouwen. De ontwikkelaars, test professionals, designers en anderen. In het meest optimale geval zijn dat er zeven, plus of min twee (scrum taal om te zeggen dat het er tussen de 5 en 9 moeten zijn). Minder is niet wenselijk want dan kun je niet veel opleveren, meer is te veel om dat het dan lastig is om in de gaten te houden wat er allemaal gebeurd.

Nu is het zo dat in theorie een team alles aankan wat het voorgeschoteld krijgt. Als er maar één team is die aan het product werkt is dat ook wel de bedoeling, echter als er meerdere teams zijn wordt het wat minder realistisch om dat te verlangen. Sterker nog: het is beter om dat NIET te verlangen. Niet iedereen is gelijk. Niet iedereen heeft dezelfde skillset of dezelfde interesses. Het is in zo'n geval beter om teams te vormen op basis van gelijkwaardige wensen en mogelijkheden. Je kunt denken aan een team dat meer UI taken oppakt en een team dat zich meer richt op de backend. Dat is iets waar je tijdens de backlog meetings rekening mee zou moeten houden.

Daily StandUp

Ik zie steeds meer organisaties die de daily standup meetings misbruiken als een soort status meeting. Een bijeenkomst waarin de status van het project besproken wordt. Nu is er niets mis mee om dat soort dingen te bespreken maar wel tijdens de standup. De standup is bedoelt om te kijken hoe ieder teamlid er voor staat. Een soort sanity check van de leden, als je wilt. Het is niet een technische meeting, het is ook geen voortgangsmeeting. Het is de vinger aan de pols van de gemoedstoestand van het team.
Bij de standup wordt aan ieder teamlid een set van drie vragen gesteld. Ieder teamlid komt aan het woord. En alleen het teamlid dat het woord heeft mag iets zeggen. Natuurlijk zal het gebeuren dat af en toe er een vraag ter verduidelijking voorkomt, maar dat moet je zoveel mogelijk zien te vermijden. De drie vragen die gesteld worden zijn

1. Wat heb je gisteren gedaan?
2. Wat ga je vandaag doen?
3. Is er iets wat het halen van punt 2 mogelijk in de weg staat?

En meer niet. Deze vragen kun je allemaal af doen met een antwoord van één of twee zinnen. Met een team van 9 personen kun je de standup dus in minder dan een kwartier afronden.

Ik hang altijd een bordje op aan de muur met een P erop (het bekende parkeerbord uit de wegenverkeerswet). De scrum master heeft een stapel post-its in zijn handen en als er tijdens de standup vragen komen worden die opgeschreven en de post-it komt op de Parkeerplaats te hangen. Als iedereen zijn zegje gedaan heeft kunnen de vragen die op P hangen beantwoord worden. De standup is afgelopen dus teamleden die vinden dat wat bij de P hangt niet voor hun is kunnen weer aan het werk gaan.

Je ziet dat op deze manier de focus weg is van de taken die er liggen maar dat er aan iedereen wordt gevraagd: "Hoe gaat het nou met je?" Dat is een hele persoonlijke vraag maar wel een vraag die er voor zorgt dat men kwijt kan wat er eventueel niet lekker loopt. Iedereen krijgt de ruimte om iets te zeggen (iedereen MOET iets zeggen zelfs) en weet dat hij of zij niet onderbroken zal worden. Het gaat om de beleving van de leden, niet om de taken. Hou de focus op de mensen!

Team rollen

Sommige teams werken goed samen, anderen hebben voortdurend conflicten. Sommige teams leveren goed werk op en doen dat op tijd ook, andere teams leveren nooit alles op tijd af en wat ze opleveren is nog slecht ook. Tja, we werken met mensen en mensen zijn niet perfect. Iedereen is anders en heeft zo zijn eigen manier van werken en omgaan met de dagelijkse werkzaamheden. Dat kun je willen veranderen en iedereen in een soort super ontwikkelaar willen veranderen maar dat gaat niet werken.

Er is enorm veel psychologisch onderzoek gedaan naar het functioneren van mensen in teams. Sociologen en psychologen schrijven boeken vol met de manier waarop mensen samen (zouden moeten) werken. Nu zeg ik niet dat je psychologie gestudeerd moet hebben om een goed scrum team samen te kunnen stellen, maar ik zeg wel dat een beetje kennis enorm kan helpen. Zo kun je relatief eenvoudig achterhalen waarom team A nu zo goed draait en team B altijd maar faalt. Vaak ligt dat aan de team rollen.

Iedereen die met anderen samenwerkt, heeft een bepaalde manier van werken. Hoe gedragen we ons in een team? Zijn we geneigd om ons terug te trekken of zijn we juist heel open? Ben je bereid om risico's te nemen door nieuwe ideeën voor te stellen of ben je liever iemand die een taak van het begin tot het einde af maakt volgens de beschrijvingen?

Een ideaal team heeft een beetje van alles wat door elkaar heen. Een team van alleen super-professionals (senior ontwikkelaars, zeer ervaren designers, enzovoorts) zal nooit zo goed draaien als je zou verwachten van een dergelijke groep mensen. Het is de mix die een team zo sterk maakt.

Om te kijken hoe iedereen zich gedraagt in een team zijn er vele testen mogelijk. Zelf gebruik ik altijd de Belbin test. Dit is een test waarin je zeven stellingen krijgt te zien waar je iets van moet vinden. Je kunt punten geven aan antwoorden op de stellingen die je het meest bij jou vindt passen. Als je dat doet, dat kost je niet meer dan een kwartier, dan krijg je een grafiek als resultaat met daarin een aantal rollen die bij jou passen. Ik zeg een aantal, want niemand valt precies in één rol. We hebben allemaal meerdere kanten in ons, maar sommige kanten zijn sterker dan anderen.

Bij Belbin worden de volgende rollen onderscheiden:

- Shaper: Task focused, driven by need to achieve
- Resource investigator:
  Vigorously pursuing contacts and opportunities
- Plant: Creative, unorthodox and generator of ideas
- Monitor Evaluator: Fair and logical observers and judges
- Completer Finisher: Perfectionist, willing to go the extra mile
- Specialist: Passionate about learning in their own field
- Coordinator: Able to see the big picture
- Team Worker: The 'oil between the cogs', listeners and diplomats
- Implementer: Takes ideas and turn them into positive actions

Een ideaal team bevat van alle rollen minstens een vertegenwoordiger. Je hebt iemand nodig die nieuwe ideeën genereert, maar ook iemand die die ideeën omzet in een product feature. Je hebt iemand nodig die de richting aanwijst waar we heen moeten maar ook iemand die de rest kan ondersteunen. De rollen zijn complementair. Een team met alleen shapers (sterke persoonlijkheden, nogal dwingend en zeer

resultaat gericht) zal meer intern ruzie maken dan software. Een team bestaande uit implementers gaat achter zitten wachten tot iemand ze verteld wat ze moeten gaan doen. Dat is uiteraard lichtelijk overdreven maar in de praktijk zie je dit wel gebeuren. Een ideaal team is een team dat bestaat uit een mix van de rollen.Er is nog een voordeel in het identificeren van de rollen in het team. Als je er een team oefening van maakt, en de resultaten van iedereen bespreekt in het team zelf, zul je zien dat mensen in het team meer begrip voor elkaar gaan krijgen. Ze snappen ineens waarom ze een hekel hebben aan de ene persoon en goed met de ander op kunnen schieten. Twee shapers liggen vaak overhoop: ze willen immers hun doel bereiken, ongeacht wat. Maar als ze dat van elkaar weten zijn ze ineens veel meer bereid om naar elkaar te luisteren. Ik heb meegemaakt dat teams waar voortdurend ruzie gemaakt werd ineens goed ging samenwerken doordat men van elkaar wist hoe men in het team zat en wat de drijfveren waren. Ook tijdens de retrospective en de standup is dit kennis die goed van pas komt.

Het gaat om mensen

Je ziet: scrum is niet zo ingewikkeld. Ik kan nog veel meer vertellen over de processen en technieken, maar het gaat om de mensen. De teamleden moeten het product gaan maken. Zorg er dus voor dat de focus op de teamleden ligt en niet op de processen. Producten komen en gaan, mensen blijven in de regel wat langer hangen. Op deze manier kun je het beste uit iedereen halen en dat komt de producten, de sfeer maar vooral ook ieders gemoedsrust ten goede. Het gaat immers om mensen, niet om technieken. •

Dennis Vroegop

Dennis Vroegop (1970) heeft zo'n 20 jaar ervaring als professioneel ontwikkelaar, afgezien van de 10 jaar daar voorafgaand als hobbyist. Het meest van die tijd heeft hij besteed aan het maken van applicaties op het Microsoft platform. Naast het pure programmeren heeft hij een passie voor het uitdenken van goed bruikbare systemen. Die liefde is het meest duidelijk terug te zien in zijn werk op diverse Multi-touch platformen en dan met name de toepassingen voor mensen die een beperking hebben zoals kinderen met autisme.

Naast zijn dagelijkse werk is Dennis voorzitter van dotNed en regelmatig spreker op diverse conferenties. Voor zijn inzet is Dennis regelmatig door Microsoft onderscheiden. Dennis mag zichzelf sinds kort de eerste Hardware Interaction Design and Development MVP noemen.

Delphi Nieuws

Delphi XE4 Nieuws

Tegen de tijd dat dit magazine uit is, is de volgende versie van Delphi en RAD Studio ook al een tijdje beschikbaar. Daarmee doet Embarcadero aan een trendbreuk: vanaf 2007 was het gebruikelijk om eenmaal per jaar, rond eind augustus / begin september de nieuwe versie van Delphi uit te brengen. Maar waar Delphi XE3 inderdaad die periode vorig jaar uitkwam, is het nu april 2013 dat we een XE4 versie zien. En niet eens XE3.5 (zoals de geruchten eerder deden geloven).

Er zal waarschijnlijk een speciale upgrade prijs zijn voor mensen die van XE3 komen, ter vergelijking van mensen die "nog" op XE2 of XE zitten (en voor mensen met een nog oudere versie van Delphi is het wellicht niet meer mogelijk om een upgrade aan te schaffen, alhoewel Embarcadero die regels nog wel eens (tijdelijk) wil aanpassen).
Zie http://www.delphixe.eu voor een overzicht van de XE4 prijzen (die op het moment van schrijven nog niet bekend zijn, maar straks bij het uitkomen van SDN Magazine wel bekend zullen zijn).

FireDAC

AnyDAC was een alternatieve data access library en verzameling componenten die al jaren door Delphi gebruikers wordt toegepast.

De nieuwe product manager van Delphi - Marco Cantu - is er ook een groot fan van. En mede dankzij de inspanningen van Marco, is AnyDAC (het product, het bedrijf en de ontwikkelaar) inmiddels "opgekocht" door Embarcadero. Niet alleen Delphi XE4, maar ook Delphi XE3 ontwikkelaars kunnen profiteren van het feit dat Embarcadero AnyDAC heeft gekocht, en inmiddels onder de naam FireDAC heeft toegevoegd aan Delphi.

FireDAC is gratis voor alle Delphi XE3 Enterprise gebruikers (ga naar **http://cc.embarcadero.com/item/29318** om het te downloaden en installeren). Ook zal FireDAC onderdeel uitmaken van Delphi XE4, en wederom gratis zijn voor de Enterprise (of hogere) edities.
Wie de Professional editie van Delphi heeft (of koopt) kan FireDAC aanschaffen als FireDAC Client/Server Add-on Pack voor Delphi XE3 (maar eerlijk gezegd verwacht ik dat men dan liever voor Delphi Enterprise zou gaan, daar dit dan ook DataSnap in bijvoorbeeld, voor het maken van multi-tier toepassingen).

Delphi Developer Days 2013

Cary Jensen en Bob Swart zijn inmiddels begonnen met de Delphi Developer Days 2013; een aantal 2-daags seminars in Europa en de USA. Steden als Chicago (6 en 7 mei), Frankfurt (30 en 31 mei) en Amsterdam (3 en 4 juni) worden bezocht. De agenda met alle sessies - zowel XE4 als oudere versies van Delphi - en details is te zien op http://www.delphideveloperdays.com/. De prijs is €799 (ex.BTW) voor twee dagen, met 10% korting voor wie al eerder deelnemer was aan DDD.

Wat doe jij volgende maand?

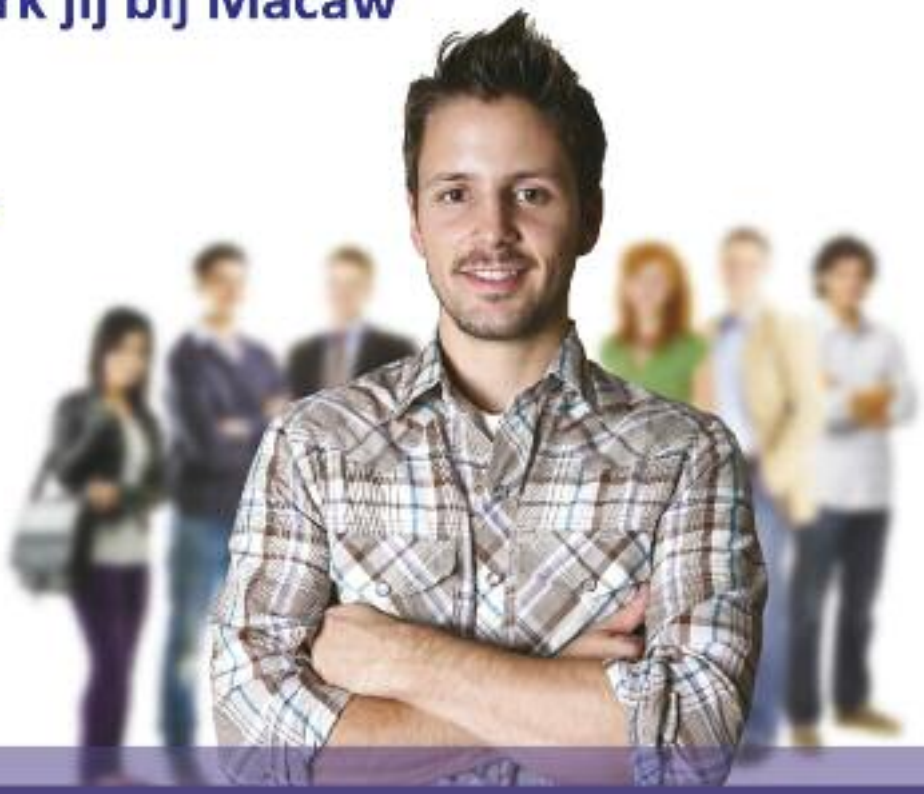
Jij bouwt de nieuwe webapplicatie voor één van de top drie verzekeraars in Nederland, waarmee mensen snel en eenvoudig online hun verzekeringen kunnen afsluiten, inzien en beheren. Je zet jouw creatieve ideeën om in slimme code en gebruikt de nieuwste .NET-technologie om front-ends te maken met de beste koppelingen naar backoffice en betaalsystemen van de klant.

Jij bent namelijk een specialist die wel raad weet met ASP.NET, C#.NET, SharePoint en SQL. Iemand die altijd op zoek is naar nieuwe kennis en als eerste met de technologie van morgen wil werken, maar ook iemand die geniet van het succes dat de klant vandaag boekt met jouw innovatieve oplossingen.

Jij deelt graag je visie en kennis met je omgeving, zowel fysiek als online, en werkt samen met andere Developers, Designers, Architecten en Analisten aan de beste oplossingen voor de klant. Je weet hoe je werk en privé vlekkeloos kunt combineren en onafhankelijk van tijd en plaats de optimale bijdrage levert aan je team.

Volgende maand werk jij bij Macaw

Architect
Senior .NET Developer
Senior SharePoint Developer
Application Developer



Macaw
tomorrow's IT, today

Met vestigingen in Schiphol-Rijk, Barendrecht en Apeldoorn is Macaw Nederlands grootste, uitsluitend op het Microsoft platform gespecialiseerde IT-dienstverlener. Onze kennis is gebundeld in drie Solution Centers en twee Service Centers die zich richten op het realiseren van collaboratieve portalen, ECM-systemen, internet en commerce sites, business solutions, CRM-systemen, applicatiebeheer en hosting diensten.

Bezoek voor meer informatie over deze en andere vacatures onze website www.echtleukwerk.nl, volg ons op www.facebook.com/MacawNL, twitter.com/MacawNL of neem contact op met onze afdeling Recruitment via 020-8510510.