Bing Maps control met MVVMLight op WP7

Een krachtig control met een solide architectuur

Joost van Schaik

Nog niet iedereen kent het, maar MVVM is hard op weg de de facto standaard architectuur te worden voor alle omgevingen die XAML gebruiken – WPF, Silverlight, Surface en Windows Phone 7. Dit artikel laat zien hoe dit pattern kan worden toegepast en hoe dit kan helpen een complexe zaak als tombstoning uiterst eenvoudig te maken. Dit gebeurt aan de hand van een wat ongebruikelijke benadering – het aansturen van het Bing Maps Control.

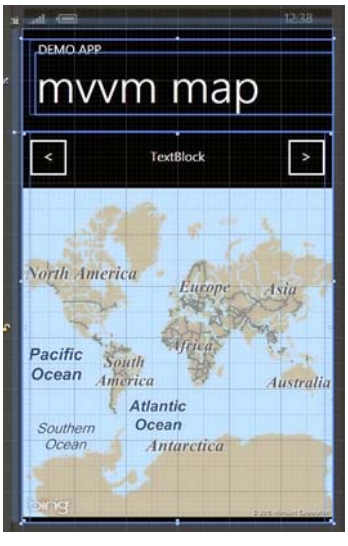
MVVM staat voor Model-View-ViewModel. Model (business classes) en View (XAML) worden gekoppeld middels een ViewModel – dat door Josh Smit 'basically a value converter on steroids' is genoemd . Het ViewModel is een class die properties uit het model op dusdanige wijze beschikbaar stelt dat deze in de View middels data binding kunnen worden getoond.
Omdat data binding twee kanten op kan werken, wordt een wijziging in het ViewModel automatisch in de View getoond, en een wijziging in de View (als de gebruiker iets invoert of wijzigt) automatisch doorgevoerd in het model. Daarnaast kan de View binden aan commands, dit zijn methods in de ViewModel die door een actie in de View (en druk op een knop bijvoorbeeld) worden aangeroepen en dan actie in het model tot gevolg hebben. In theorie is dan de 'code behind' van de XAML files geheel leeg. Deze benadering heeft twee voordelen. Als eerste kan het ViewModel door gebruik te maken van unit tests automatisch getest worden, iets wat met de traditionele code-behind benadering niet of nauwelijks mogelijk is. Ten tweede is de View – de XAML dus – geheel vrij van code. Dit houdt in dat een designer, die totaal geen boodschap aan de code heeft, de View onder handen kan nemen middels Expression Blend.

MVVM – de praktijk op Windows Phone 7

In Windows Phone 7 – gebaseerd op Silverlight 3 – zijn er wat uitdagingen. Het binden van properties gaat prima, maar het binden aan commands gaat niet – om de eenvoudige reden dat de daarvoor benodigde classes niet bestaan. Daarnaast speelt in Windows Phone 7 mee dat de applicatie op elk moment kan worden onderbroken – doordat de gebruiker op 'Start' drukt bijvoorbeeld. Het is dan wel de bedoeling dat de toestand van de applicatie bewaard blijft – het reeds genoemde tombstoning.
Er zijn diverse frameworks op o.a. Codeplex te vinden waarmee volledig MVVM support in Silverlight en Windows Phone 7 kan worden gerealiseerd. De meest bekende zijn (in willekeurige volgorde) MVVMLight van Laurent Bugnion, Silverlight MVP van het jaar 2010 en Caliburn.Micro van Rob Eisenberg. Beiden kennen hun voor- en nadelen, beide ook hun voor- en tegenstanders. In dit artikel wordt ingegaan op de toepassing van MVVMLight. In het volgende voorbeeld zitten alle classes in één project – in de praktijk zullen bepaalde classes in aparte assemblies terechtkomen. Het idee van dit artikel is een principe te laten zien.

Basis applicatie

De samenwerking tussen Bing Maps en MVVMLight wordt gedemonstreerd aan de hand van een simpele applicatie "MapBindingDemo". Deze bestaat uit een Bing Maps control, twee buttons om te wisselen tussen de kaarttypen en een TextBlock om de naam van de geselecteerde kaart te tonen. Daarnaast komen natuurlijk de pinch zoom, pan en double taps voor kaartmanipulatie van het Bing Maps control mee. Het makkelijkste is om te beginnen met een lege "Windows Phone Application" in Visual Studio 2010, deze vervolgens in Blend te openen, in de Assets box te zoeken op "Map", en daarna de kaart op de design pane van MainPage.xaml te slepen. Voeg daar de overige controls aan toe tot het resultaat er als volgt uitziet:



Of, in XAML: [**listing: designedapp.xaml**]

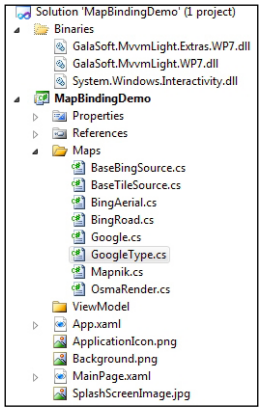
(**Alle sources in deze tekst zijn te downloaden via http://www.schaikweb.net/dotnetmag/MapBindingDemo.zip**)

Zij die het graag zonder Blend doen: het project moet een reference krijgen naar Windows.Controls.Phone.Maps en in de namespace declaraties van de PhoneApplicationPage moeten de volgende namespace declaraties staan:
xmlns:Microsoft_Phone_Controls_Maps="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps".
Wat verder belangrijk is bij het gebruik van het Bing Maps Control: de credentials die nodig zijn. Maak deze aan middels https://www.bingmapsportal.com/. De key die daar wordt verkregen komt in het CredentialsProvider attribuut van het map control. De uri die wordt gevraagd bij het aanmaken van de credentials doet niet ter zake – dat is alleen van toepassing als het Bing Maps control in een web context wordt gebruikt. Bij een Windows Phone 7 applicatie is dat duidelijk niet het geval.

Bing Maps control en TileSource

Het Bing Maps control is een bijzonder flexibel en krachtig control, dat zowel raster als vector kaarten kan laten zien; dit voorbeeld concentreert zich op het eerste. Hierbij is het belangrijk om te weten dat het control drie instellingen qua kaartsoort heeft: "Road", "Aerial" en de minst bekende - "Mercator". Als Mercator is ingesteld, haalt het control geen data op maar moet de programmeur zelf een class aanleveren die Microsoft.Phone.Controls.Maps.TileSource implementeert. Daarin hoeft alleen de method "GetUri" te worden overridden. Deze method krijgt van het Bing Maps control drie parameters voor zijn kiezen: een x, een y en een zoomlevel. Op level 1 is de wereld in van 2x2=4 tiles van 256 pixels verdeeld, op 2 in 16, op 3 in 64, enzovoort. Het is aan de GetUri method om deze drie getallen om te zetten in een uri die verwijst naar een plaatje. Voor geo fanaten is dit gesneden koek, voor de rest van de wereld vermoedelijk ietwat vaag. Dit is voor het begrip van MVVM geen probleem en bovendien zitten de implementaties die nodig zijn gewoon bij dit artikel. In de applicatie komt een folder "Maps" met daarin een aantal classes die TileSource implementeren:

De code voor deze classes is voor dit artikel in één bestand gevoegd om één en ander wat overzichtelijker te maken. Normaal gesproken plaatsen alleen code generatoren meerdere classes in één bestand.



[**listing: titlesources.cs**]

De start is een base class "BaseTileSource" die alleen een Name property toevoegt en een Equals op basis van de naam – een wat naïeve methode, maar voldoende voor een voorbeeld. Een subclass "BaseBingSource" bevat de meeste 'heavy lifting' voor de berekeningen voor Bing Maps, waarna er nog twee subclasses "BingAerial" en "BingRoad" nodig zijn om Bing Aerial en Bing Road te implementeren. Tot zover is er nu alleen op een wat ingewikkelde manier de standaard functionaliteit van het Bing Maps Control nagemaakt, maar het wordt leuker als er twee bekende open source mapservers van OpenStreetMaps kunnen worden gekoppeld, te weten OsmaRender en MapNik, en natuurlijk mijn persoonlijke favoriet in de categorie 'plaagstootje richting concurrentie': Google Maps.

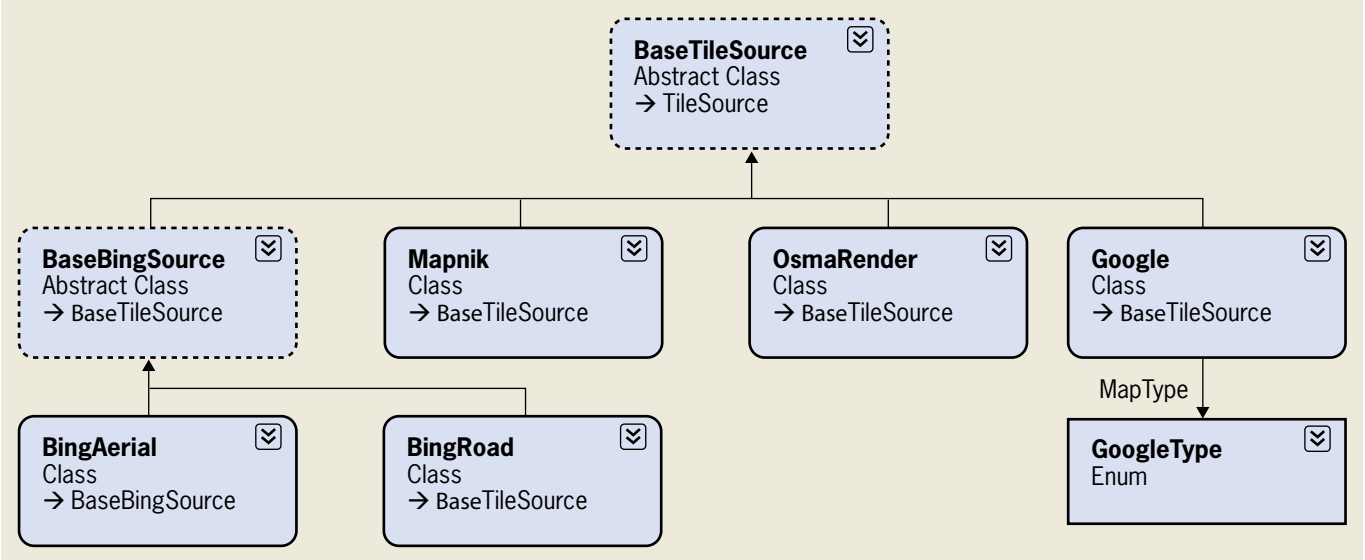
Zoals eerder vermeld moet de Bing Maps control worden ingesteld op Mercator mode . Dat kan als volgt:

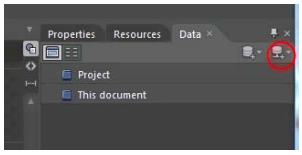
[**listing: mercatormode.xaml**]

De namespace "MSPCMCore" is gedeclareerd als "clr-namespace:Microsoft.Phone.Controls.Maps.Core;assembly=Microsoft.Phone.Controls.Maps" in de namespace declaraties in de PhoneApplicationPage tag.

Setup MVVMLight

Het framework zelf bestaat uit twee assemblies: GalaSoft.MvvmLight.WP7.dll en GalaSoft.MvvmLight.Extras.WP7.dll. Deze zijn te downloaden van http://mvvmlight.codeplex.com/. Daarnaast is de assembly System.Windows.Interactivity.dll nodig – deze komt uit de Expression Blend SDK en kan gevonden worden in de <drive>:\Program Files\Microsoft SDKs\Expression\Blend\Windows Phone\v7.0\Libraries folder - of <drive>:\Program Files (x86)\Microsoft SDKs\Expression\Blend\Windows Phone\v7.0\Libraries in het geval van een 64bit OS (wat steeds





meer gemeengoed wordt). Het is aan te raden om deze assemblies in een solution folder te plaatsen zodat ze ook altijd netjes bij de solution beschikbaar zijn, wat handig is als er meerdere mensen aan een Windows Phone 7 project werken met behulp van een source control mechanisme. In dit voorbeeld is gekozen voor een folder "Binaries". Vanuit het project dient een referentie naar deze drie assemblies te zijn. Maak tenslotte een lege folder "ViewModel", waarna de solution er als volgt uitziet:

ViewModel voor de kaart applicatie

Een ViewModel in MVVMLight is afgeleid van een class die uiterst orgineel "ViewModelBase" heet. In het model is een aantal properties gedefinieerd: ZoomLevel, MapCenter, AvailableMaps, en CurrentMap. Daarnaast zijn twee commands gedefinieerd: één om een kaartdefinitie vooruit te springen, en één om een kaartdefinitie terug te springen. Het ViewModel ziet er als volgt uit:

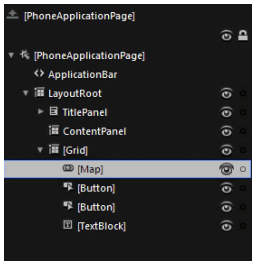
[listing: MvvmMap.cs]

Koppelen van het model met de user interface met behulp van Blend

ZoomLevel en MapCenter kunnen zonder meer middels data binding worden gekoppeld aan een Bing Maps control, terwijl in het TextBlock met als inhoud "Textblock" de naam van de getoonde kaart moet worden getoond. Het makkelijkst gaat dat door het project te openen in Blend. Een versie hiervan voor Windows Phone zit bij de gratis tools voor Windows Phone in. Wanneer Blend is opgestart en het project ingeladen is, staan rechts boven drie tabs: "Properties", "Resources" en "Data".

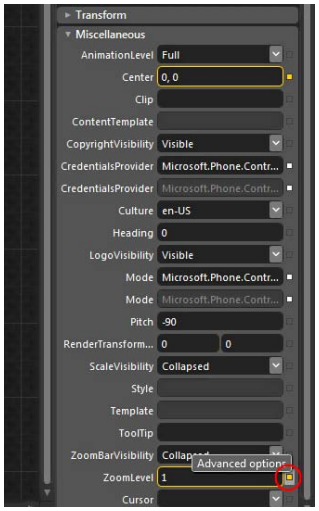
Klik op het omcirkelde symbool en er verschijnt een menu met onder andere de tekst "Create Object Data Source". Bij selectie daarvan verschijnt een popup met daarin alle objecten van de solution. Klik achtereenvolgens MapBindingDemo en MapBindingDemo.ViewModel open, waaronder zich "MvvmMap" bevindt. Klik deze aan - aan de rechterzijde worden nu alle properties van MvvmMap getoond. Sleep vervolgens de "Instance" property van MvvmMap naar het Grid "LayoutRoot" in "Objects and TimeLine". Blend toont de tekst "Data bind LayoutRoot.DataContext to Instance". Laat de muisknop los en klaar – data binding.

Helemaal rechts onder verschijnt nu een panel "Data context". Sleep de property
"Name" van "CurrentMap" op het TextBlock. Het resultaat is dat de inhoud van de text box direct wijzigt in "Bing Aerial". En dat klopt ook, want uit het model blijkt dat indien er geen CurrentMap is, de eerste uit de "AvailableMaps" moet worden geselecteerd en dat is inderdaad "Bing Aerial".



Selecteer vervolgens in het panel "Objects and Timeline" het object "Map". Klap hiervoor eerst achtereenvolgens "Layoutroot" en "Grid" open, waarna "Map" zichtbaar wordt:

Koppel de property MapCenter van het model aan de property "Center" van de Map door het property op het Bing Maps control te slepen en met behulp van de popup die daarna verschijnt het property "Center" te selecteren.



Dat lukt helaas niet met de property "ZoomLevel" – om wat voor duistere reden dan ook wil Blend die alleen aan "Content" binden. Om dit toch voor elkaar te krijgen, dient de volgende werkwijze te worden gevolgd: selecteer de tab "Properties" - onder het uitklapmenu "Miscellaneous" staat bijna onderaan ZoomLevel. Een klik op het kleine vierkantje rechts daarvan geeft een menu met daarop onder andere de tekst "Data Binding". Selecteer dit, en selecteer in de daarop volgende popup de property "ZoomLevel" van het model. In de XAML view staat nu in het Bing Maps Control Center="{Binding MapCenter}" ZoomLevel="{Binding ZoomLevel}"; om het model straks goed te laten functioneren is het handig daar Center="{Binding MapCenter, Mode=TwoWay}" ZoomLevel="{Binding ZoomLevel, Mode=TwoWay} van te maken. Dat kan overigens ook vanuit de grafische designer van Blend, maar dit gaat wat sneller.

Het koppelen van de commands gaat middels de EventToCommand behavior uit MVVMLight. Typ in de "Search" box de tekst "Event", waarna rechts "EventToCommand" verschijnt. Sleep deze op de buttons "<" en ">", en sleep vervolgens uit het model "PreviousCommand" op de EventToCommand onder de "<" button (in het objects and timeline panel) koppel deze aan de "Command" property van de EventToCommand. "NextCommand" wordt op eenzelfde wijze gekoppeld aan de ">" button.
Als nu vanuit Blend het project wordt opgestart (F5) in de emulator verschijnt direct het scherm; als op de knoppen ">" en "<" wordt geklikt verschijnt in het TextBlock de naam van de geselecteerde TileSource implementatie. Maar de kaart doet nog helemaal niets – sterker nog, er is helemaal geen kaart meer te zien. Dat komt omdat er nog geen TileSource is gekoppeld.

Attached dependency properties - het betere loodgieterswerk van data binding

Het binding van een TileSource aan een Bing Maps Control gaat niet, omdat er geen enkele property is waarop dit past. Gelukkig kent XAML het principe van attached dependency properties, die in zekere zin het property-equivalent van een extension method zijn. Terug naar Visual Studio dus, waar in de directory "ViewModel" een attached dependency property komt:

[listing – bindingHelpers.cs]

Hier is goed te zien waarom een attached dependency property nodig is: een Bing Map heeft Children waarin onder andere objecten van het type MapTileLayer kunnen zitten en pas daarin zit een TileSource. Dus moeten eerst alle MapTileLayer objecten worden verwijderd, waarna een nieuwe MapTileLayer kan worden gemaakt, met daarin de tilesource, en die wordt vervolgens aan de Children van de map toegevoegd.

Nu kan het gehele model gekoppeld worden aan het Bing Maps control. Dit lukt niet met Blend, maar moet in XAML. Voeg de namespace waarin BindingHelper zich bevindt toe in de namespace declaraties bovenin de PhoneApplicationPage:
xmlns:MapBindingDemo_ViewModel="clr-namespace:MapBindingDemo.ViewModel
Daarna moet de daadwerkelijke binding plaatsvinden in de map. Op dezelfde plek waar net de binding aan ZoomLevel en MapCenter heeft plaatsgevonden komt nu de binding aan het attached dependency property:

[**listing: attached_binding.cs**]

Hierna is de applicatie volledig functioneel, en zelfs in design mode wordt de default kaart (Bing Aerial) getoond. De knoppen links en rechts van de kaarttitel tonen nu steeds een andere kaart. En er staat geen letter extra code in de code behind. De user experience en de look en feel verdient nog wat aandacht, maar voor de rest - mission accomplished. Nou ja, bijna dan – want zoom eens een stuk in, selecteer een andere kaart dan Bing Aerial kaart en druk dan bijvoorbeeld op de "search" button van de telefoon en daarna op "back". Helaas - de eerste kaart wordt weer getoond, op de volledige wereld. De status van de applicatie wordt niet bewaard, of anders gezegd - de applicatie ondersteunt geen tombstoning.

Tombstoning een MVVMLight applicatie

Mike Talbot heeft een briljant artikel geschreven waarin hij een helper object beschrijft dat hij "SilverlightSerializer" heeft gedoopt. Dit helper object kan in principe elk object binnen Silverlight en Windows Phone 7 binair serialiseren. Het leuke hiervan is dat ook het in principe niet-serialiseerbare ViewModelBase van MVVMLight toch kan worden geserialiseerd. Het artikel, getiteld "Silverlight Binary Serialization" is samen met de bijbehorende source code te vinden op http://whydoidoit.com/2010/04/08/silverlight-serialization/. De SilverlightSerializer source komt in een folder "Serialization" in de solution. In diezelfde folder (en namespace) komt een file "ApplicationExtensions.cs" die twee extension methods op de Application class bevat bevat: één voor het wegschrijven van het complete ViewModel naar isolated storage en één voor het teruglezen.

[**listing: applicationExtensions.cs**]

Om deze methodes te kunnen gebruiken moet er in de code behind van App.Xaml worden geprogrammeerd. Dit lijkt een afwijking van MVVM, maar dat valt mee. Sowieso moeten een aantal methods die daarin default staan blijven staan (de Application_* methods), want die worden aangeroepen bij starten en afsluiten van een applicatie. Er komt alleen wat code in de bestaande methodes.
Voeg eerst twee "usings" toe:
using MapBindingDemo.Serialization;
using MapBindingDemo.ViewModel;

Pas daarna de methods die worden aangeroepen in App.xaml.cs bij starten, activeren, deactiveren en afsluiten van de applicatie als volgt aan:

[**listing: app.xaml.fragment.cs**]

Merk op dat zowel bij deactivation als bij closing hetzelfde gebeurt: er wordt altijd data naar isolated storage geschreven en dus wordt bij launching en activation ook altijd de method RetrieveFromIsolatedStorage aangeroepen. Het is ook mogelijk het model in zijn geheel op te slaan in de PhoneApplicationService maar dan is het opgeslagen model niet beschikbaar indien de applicatie opnieuw wordt opgestart vanuit het start menu. Dat is weliswaar een 'works as designed' voor wat betreft Windows Phone 7, maar om wat consistenter gedrag te bereiken wordt de state in beide gevallen opgeslagen in isolated storage. Dus of de applicatie nu via de 'back' toets wordt bereikt of door opnieuw op te starten, de laatste state wordt altijd getoond.

Belangrijkste nadeel van deze methode is overigens dat bij een belangrijke wijziging van het viewmodel het laden uit isolated storage mislukt. Daarom staat er ook altijd een try/catch omheen staan. Oplettende lezers merken tevens op dat de property "AvailableMapSources" van het model voor niets wordt geserialiseerd, want die wordt toch altijd vanuit de constructor van het model gevuld. Default serialiseert de SilverlightSerializer alles, maar datgene wat niet geserialiseerd hoeft te worden kan worden gemarkeerd met het [DoNotSerialize] attribute dat SilverlightSerializer zelf definieert.

Conclusie

Voor alle Windows Phone 7 applicaties behalve de meest triviale is MVVM eenvoudigweg "the way to go", al is het alleen maar omdat met behulp van de SilverlightSerializer iets complex als tombstoning zo ongeveer gratis is. Voor complexe applicaties met meerder modellen wordt het uiteraard ingewikkelder, maar MVVMLight en SilverlightSerializer vormen samen een 'winning team'.

Joost van Schaik, is Architect/Developer bij Sevensteps in Amersfoort. Hij is te volgen via twitter als @localjoost en te bereiken via mail op Joost.van.schaik@gmail.com. De sources behorende bij dit artikel zijn te downloaden via http://www.schaikweb.net/dotnetmag/MapBindingDemo.zip

Met dank aan Jarno Peschier en Dennis Vroegop voor hun suggesties.