# ArcGIS Runtime SDK for WPF und MVVM

## MVVM

Das Model-View-ViewModel-Pattern (MVVM) ist ein Vertreter einer ganzen Familie von Pattern, deren vorrangige Aufgabe es ist, die Daten von deren Darstellung zu trennen. Die Ursprünge dieses Pattern liegen schon 30 Jahre zurück. In Smalltalk als Model-View-Controller (MVC) implementiert fand dieses Pattern über die Weiterentwicklung Model-View-Presenter (MVP) zur aktuellen Ausprägung. Dabei bezeichnet MVVM eine spezielle Form des Presentation-Pattern durch Microsoft für deren neuere Technologien WPF und Silverlight. Die Idee dahinter ist die Kapselung der Geschäftslogik in einem eigenen Konzept (Model), deren Daten mittels eines Vermittlers (ViewModel) im Frontend (View) dargestellt werden. Diese Art der Aufteilung adressiert gleich mehrere Aufgaben:

* **Fokussierung der technischen Verantwortlichkeiten** - Das Model kümmert sich ausschließlich um das Management (Lesen, Bearbeiten, Speichern) von Daten. Das ViewModel wiederum dient ausschließlich zur visuellen Aufbereitung von Daten und steuert die Logik des User Interfaces (UI). Der View ist lediglich für Anzeige und Manipulation von Daten verantwortlich. Damit wird auch das Architekturpattern SOC (separation of concerns)[1] umgesetzt.
* **Fokussierung der organisatorischen Verantwortlichkeiten** - Während der Entwickler die eigentliche Programmlogik programmiert, kann das Frontend völlig losgelöst von einem UX-Experten gestaltet werden.
* **Flexibilität** - Eine derartige Architektur erlaubt das schnelle Auswechseln und Anpassen von Komponenten. Was heute noch auf dem Desktop ist, kann morgen einfach ins Web portiert werden. Dazu muss entsprechend nur der View mit seinem zugehörigen ViewModel implementiert werden. Die meist aufwändige Businesslogik bleibt unangetastet.
* **Testbarkeit** - Durch den fehlenden UI-Bezug ist die Geschäftslogik relativ einfach Unit-testbar.

Das Besondere an MVVM ist das sogenannte DataBinding. Dabei werden die UI-Elemente vom Designer einfach auf entsprechende Anknüpfpunkte des ViewModels geklemmt. Die Technologie sorgt dann völlig eigenständig dafür, dass die Daten korrekt im Frontend angezeigt oder in der anderen Richtung im ViewModel aktualisiert werden. So kann beispielsweise ein Integerwert mit einem Slider, Spinedit oder Eingabefeld verbunden werden. Der Entwickler muss sich keine Gedanken mehr machen, den Slider oder das Spinedit korrekt zu setzen - das passiert automatisch im Hintergrund durch das Binding.

## ArcGIS Runtime SDK for WPF

ArcGIS Runtime for Desktops ist ein neues Produkt von Esri für die Entwicklung von leichtgewichtigen und funktional reichhaltigen GIS-Anwendungen für verschiedene Desktop-Plattformen (siehe Artikel in arcAKTUELL 4/2011, 2/2012 und diese Ausgabe). Dabei kann der Entwickler zwischen verschiedenen nativen SDKs wählen. Diese SDKs sind im Rahmen einer Subskription des Esri Developer Network (EDN) erhältlich. Eines davon ist das ArcGIS Runtime SDK for WPF [2]. Bei der Erstellung von Applikationen auf dieser Basis sollte natürlich auch das bewährte MVVM-Pattern angewendet werden. Ein möglicher Ansatz dazu wird im Folgenden beschrieben.

## Werkzeuge

### Entwicklungsumgebungen

Neben Visual Studio bietet Microsoft mit Expression Blend ein ausgezeichnetes Programm zum Designen von Oberflächen an. Die anvisierte Zielgruppe wird schon auf dem ersten Blick klar: Designer. Visuell orientiert sich Expression Blend an den Adobe-Klassikern - ein dunkles UI um die Augen nicht abzulenken und zahlreiche kleine Panels zur Parametrierung. Nicht nur dem Designer wird die Arbeit erleichtert, auch dem Entwickler kann Expression Blend eine gute Hilfe sein. Unter anderem lässt sich das DataBinding über diverse Menüs sehr komfortabel umsetzen.

### MVVM Light Toolkit

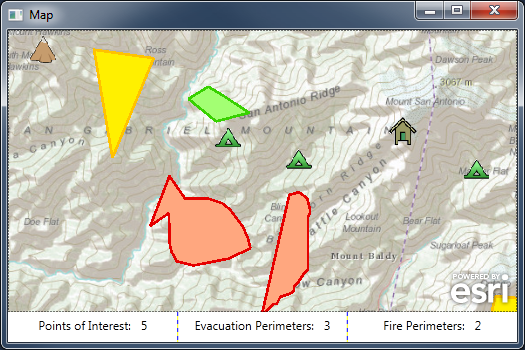
Darüber hinaus gibt es für die Entwicklung mit MVVM unzählige kleine und große Helferlein. Ein elegantes und gerne von uns verwendetes Framework ist das kostenfreie MVVM Light Toolkit von Galasoft[3]. Die Vorteile sind:

* Verwendbarkeit für WPF- und Silverlight-Projekte
* Projekt- und Item-Templates
* Viele nützliche Hilfen für DataBinding (Blendability)
* Messaging-Unterstützung
* Umfangreiche Dokumentation

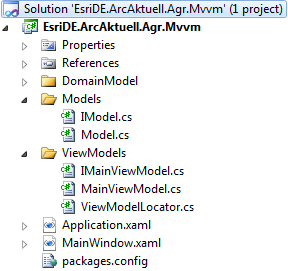
Mittels NuGet[4] lässt sich das Toolkit einfach in eigene Projekte einbinden. Dabei wird neben dem Einbetten der eigentlichen Bibliotheken auch der DevTree für die Verwendung angepasst.

## Praxis-Check

Anhand eines praktischen Beispielprojektes wird exemplarisch eine mögliche Vorgehensweise beschrieben. Wir erstellen eine kleine GIS-Anwendung auf Basis von ArcGIS Runtime SDK for WPF und nutzen das MVVM Light Toolkit als Hilfsmittel für unsere MVVM-Architektur. Die Funktion der Anwendung ist ganz simpel – es soll für jeden Feature-Layer die Anzahl der Features im momentan sichtbaren Kartenausschnitt ermittelt und angezeigt werden.



In Visual Studio erstellen wir dazu ein Projekt mithilfe eines Templates, welches nach der Installation des ArcGIS Runtime SDK for WPF in Visual Studio integriert ist. Die automatisch generierte MainWindow.xaml ist unser View. In diesem konfigurieren wir unsere Map und fügen noch einige Controls für die Anzeigen der Anzahl der Features hinzu. Anschließend erweitern wir mit NuGet unser Projekt um die MVVM Light Toolkit-Komponenten. Dabei wird unter anderem das Projekt um unser ViewModel ergänzt. Anschließend werden noch einige Komponenten für unser Model benötigt. Damit hat man strukturell schon mal alles zusammen. Dies könnte in etwa so aussehen:

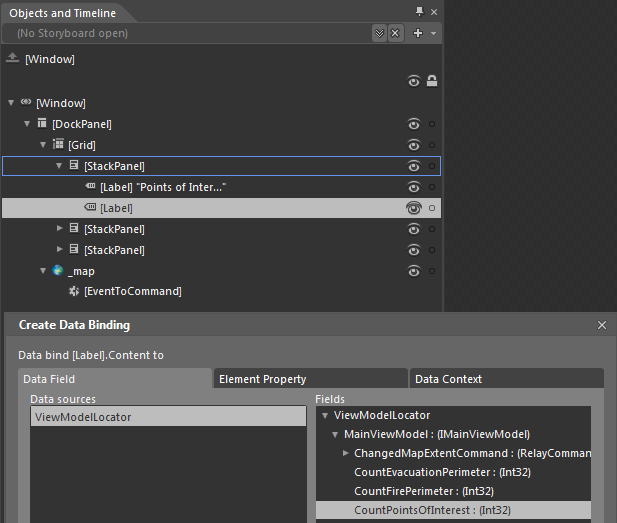


Der Sinn von MVVM ist das Trennen von UI und Logik. Für unsere Funktion müssen wir vereinfacht betrachtet auf das ExtentChanged-Event der Map lauschen, den neuen Extent den Query-Tasks mitteilen, anschließend die Queries starten und die ermittelten Ergebnisse in den jeweiligen Controls darstellen. Das Auslösen des ExtentChanged-Events und das Darstellen der Ergebnisse geschieht in der UI, also in unserem View. Das Ermitteln der Anzahl der Features durch Queries ist eine Logik, welche dementsprechend in unserem Model implementiert wird. Das ViewModel hingegen fungiert als Bindeglied zwischen beiden, es reicht nur Informationen durch.

Wie setzen wir das nun in die Praxis um? Im Model implementieren wir die gesamte Logik für die Queries, eine öffentliche Methode zum Starten und ein QueryCompleted-Event zum Mitteilen einer erfolgreichen Ausführung. Im ViewModel werden öffentliche Integer-Properties für die Anzahl der Features und ein ICommand für das Behandeln des ExtentChanged-Event der Map definiert. Zusätzlich halten wir darin eine Instanz unseres Model vor. Das ICommand stößt später im Model die Queries an und im QueryCompleted-Eventhandler werden die Ergebnisse in die Integer-Properties des ViewModels geschrieben. Nun kommt der Clou – wir verknüpfen mittels DataBinding unser ViewModel mit dem View, also der MainWindow.xaml. Damit haben alle UI-Controls Zugriff auf alle öffentlichen Komponenten in unserem ViewModel. Das DataBinding kann sehr komfortabel in Expression Blend ohne Programmierung erledigt werden. Dazu wird einfach das gleiche Visual Studio-Projekt geöffnet. In der MainWindow.xaml sieht es dann so aus:



Als nächstes binden wir mit Expression Blend die Integer-Properties des ViewModels an die jeweiligen Text-Eigenschaften der Controls für die Anzeige der Feature-Anzahl, in unserem Fall an die Content-Eigenschaft von Labels.



In XAML liest sich der Code dann folgendermaßen:



Als letztes fehlt noch die Anbindung des ExtentChanged-Events der Map an unser ICommand aus dem ViewModel. Dazu bietet das MVVM Light Toolkit ein nettes Asset namens EventToCommand, mit dessen Hilfe jegliche Events inklusive Argumente an ICommands gekoppelt werden können. Genau das nutzen wir auch und implementieren ein EventToCommand und übergeben als Argument das ExtentEventArgs mit der neuen Extent-Geometrie, welche wir für die Queries brauchen.

Das Beispielprojekt haben wir zum Download auf GitHub[5] zur Verfügung gestellt.

## Links

[1] Separation of concerns  
<http://bit.ly/OYRadt>

[2] ArcGIS Runtime SDK for WPF  
<http://bit.ly/OYRIzY>

[3] MVVM Light Toolkit  
<http://bit.ly/OYSb5c>

[4] NuGet  
<http://bit.ly/OYSrRJ>

[5] Sourcecode auf GitHub  
<http://bit.ly/OYSU6E>

Alexander Erbe und Marko Apfel  
[a.erbe@esri.de](mailto:a.erbe@esri.de), [m.apfel@esri.de](mailto:m.apfel@esri.de)   
http://geekswithblogs.net/mapfel/   
Esri Deutschland GmbH, Kranzberg