Model View ViewModel

aus Wikipedia, der freien Enzyklopädie

Model View ViewModel (MVVM) ist ein Architekturmuster der Softwaretechnik, das von Microsoft stammt. Es ist eine Spezialisierung des Entwurfsmusters *Presentation Model* von Martin Fowler.[1] MVVM basiert zum größten Teil auf Model View Controller (MVC), und ist auf moderne UI-Entwicklungsplattformen (HTML5 [2][3], WPF, und Silverlight) ausgerichtet, bei denen es einen User Experience (i.e., User Interface) (UXi) Entwickler mit anderen Anforderungen als jenen eines eher traditionellen Entwicklers gibt (der sich z.B. in Richtung Geschäftslogik- und Back-End-Entwicklung orientiert). Das View-Model von MVVM ist ein Value Converter[4] was bedeutet, dass das View-Model dafür verantwortlich ist, die Datenobjekte des Modells derart bloßzulegen, dass sie einfach verwaltet und benutzt werden können. Insofern ist das View-Model eher Model als View, es wickelt die meiste wenn nicht alle Anzeigelogik des Views ab (wobei die Abgrenzung, welche Funktionen von welcher Schicht implementiert werden, ein Gegenstand laufender Diskussionen [5] und Untersuchungen ist).

MVVM wurde entworfen, um mittels der spezifischen Funktionen in WPF durch Entfernung praktisch alles Code-Behind von der View-Schicht[6] eine bessere Trennung der View-Schicht vom Rest zu ermöglichen. Statt View-Code zu schreiben, kann ein Designer die native WPF-Auszeichnungssprache XAML verwenden und Bindungen zum ViewModel erzeugen, die durch Anwendungsentwickler geschrieben und gewartet werden. Diese Rollentrennung erlaubt es Designern, eher einen Fokus auf die UX-Anforderungen zu legen, als auf Programmierung oder Geschäftslogik, wodurch die Schichten einer Anwendung in mehreren Arbeitsgruppen entwickelt werden können.

Inhaltsverzeichnis

Geschichte

Microsoft MVP Josh Smith sagte[6]

> "In 2005, John Gossman, currently one of the WPF and Silverlight Architects at Microsoft, unveiled the Model-View-ViewModel (MVVM) pattern on his blog. MVVM is identical to Fowler's Presentation Model, in that both patterns feature an abstraction of a View, which contains a View's state and behavior. Fowler introduced Presentation Model as a means of

> creating a UI platform-independent abstraction of a View, whereas Gossman introduced MVVM as a standardized way to leverage core features of WPF to simplify the creation of user interfaces. In that sense, I consider MVVM to be a specialization of the more general PM pattern, tailor-made for the WPF and Silverlight platforms."

In deutscher Übersetzung:

> "2005 hat John Gossman, zurzeit einer der WPF- und Silverlight-Architekten bei Microsoft, das Model-View-ViewModel (MVVM) Muster auf seinem Blog enthüllt. MVVM ist mit Fowler's Presentation Model insofern identisch, als dass beide Muster eine Abstraktion der View aufweisen, die den Zustand und das Verhalten einer View beinhaltet. Fowler hat das Presentation Model vorgestellt als ein Mittel um eine UI plattformunabhängige Abstraktion einer View zu erzeugen, wohingegen Gossman das MVVM als einen standardisierten Weg vorgestellt hat, die Kernfunktionen der WPF zu nutzen, um die Erzeugung von Benutzerschnittstellen zu vereinfachen. In diesem Sinne würde ich MVVM als eine Spezialisierung der allgemeineren PM-Muster auffassen, maßgeschneidert für die WPF- und Silverlight-Plattformen."

Das MVVM-Muster wurde erdacht, um WPF und Silverlight zu unterstützen, beides Komponenten die 2006 mit dem .NET Framework 3.0 eingeführt wurden. Das Muster findet nun in allgemeinerer Form auch in anderen technologischen Bereichen Anwendung, ganz ähnlich wie bei den früheren Mustern MVC und Model View Presenter (MVP).

Mehrere Microsoft Architekten, die an WPF und Avalon arbeiten, haben in Online-Medien ausführlich über das MVVM geschrieben, einschließlich dessen Schöpfer John Gossman, Microsoft MVP Josh Smith, und Microsoft Program Manager Karl Shifflett.

So wie sich das Verständnis des Musters durch die Softwareindustrie verbreitet, dauert auch die Diskussion an, welche Werkzeuge entwickelt werden können um das Muster zu unterstützen, wo verschiedene Arten von unterstützendem Code platziert werden sollen, die beste Methode für die Datenbindung, wie Daten innerhalb des ViewModel bloßgelegt werden sollen, wie angebracht das Muster für Javascript ist, und so weiter.

Beschreibung

Allgemein gesagt[6][7][8] versucht das Model View ViewModel Muster, sowohl die Vorteile einer Trennung funktionaler Entwicklung (bereitgestellt durch MVC), als auch die Vorteile von XAML und der Windows Presentation Foundation zu nutzen, indem es Daten so weit zurück (so nahe am Modell) wie möglich bindet, während es das XAML, ViewModel, und die Datenüberprüfungsfunktionen jeder Geschäftsebene benutzt, um jegliche eingehenden Daten zu validieren. Das Ergebnis ist, dass das Model und die Foundation soviel von den Operationen wie möglich antreiben, was den Bedarf nach Code-Behind minimiert, vor allem in der View.

Elemente des MVVM-Musters beinhalten:

Model: wie im klassischen MVC-Muster bezieht sich das Modell entweder auf (a) ein Objektmodell das den realen Zustandsinhalt repräsentiert (objektorientierter Ansatz) oder (b) die Datenzugriffsschicht welche diesen Inhalt repräsentiert (datenzentrischer Ansatz).

View: wie im klassischen MVC-Muster bezieht sich die View auf alle Elemente die durch die GUI angezeigt werden, etwa Schaltflächen, Fenster, Grafiken, und andere Steuerelemente.

ViewModel: das ViewModel ist ein “Modell der View” was bedeutet, dass es eine Abstraktion der View ist, die auch der Datenbindung zwischen View und Model dient. Es könnte betrachtet werden als ein spezieller

Aspekt eines Controllers (im MVC-Muster) für die Datenbindung und -konvertierung, der Information vom Model zu solcher für die View konvertiert und Befehle von der View zum Model reicht. Das ViewModel legt öffentliche Properties, Befehle und Abstraktionen bloß. Das ViewModel kann mit einem konzeptionellen Zustand der Daten verglichen werden, im Gegensatz zum realen Zustand der Daten im Model.[9]

Controller: manche Referenzen für MVVM enthalten auch eine Controller Schicht oder illustrieren dass das ViewModel eine spezielle Funktionsmenge parallel zu einem Controller ist, während andere dies nicht tun (http://karlshifflett.files.wordpress.com/2008/11/wpflobmvvm1.png) . Der Unterschied ist ein Gebiet laufender Diskussionen hinsichtlich der Standardisierung des MVVM-Musters.

Eine Implementierung des ViewModel

Snippet: hier eine simple Implementierung des Musters, die mittels TDD auf WPF How to implement MVVM (Model-View-ViewModel) in TDD (http://code.msdn.microsoft.com/How-to-implement-MVVM-71a65441) , auf Code MSDN (http://code.msdn.microsoft.com) realisiert wurde.

Eine Implementierung des ViewModel in C#.

```
public class CustomerViewModel
        : ViewModelBase<CustomerViewModel>
{
        private readonly IDialogService dialogService;
        private Customer currentCustomer;
        private int i;

        public CustomerViewModel()
        {
                CustomerList = new ObservableCollection<Customer>();
                AddNewCustomer = new RelayCommand(PerformAddNewCustomer);
        }

        public CustomerViewModel(IDialogService dialogService)
                : this()
        {
                this.dialogService = dialogService;
        }

        public Customer CurrentCustomer
        {
                get { return currentCustomer; }

                set { SetProperty(ref currentCustomer, value, x => x.Curr
        }

        public ObservableCollection<Customer> CustomerList
        {
                get;
                private set;
        }

        public ICommand AddNewCustomer { get; private set; }
```

```
        private void PerformAddNewCustomer()
        {
                CustomerList.Add(new Customer { Name = "Name" + i });
                i++;

                if (dialogService != null)
                {
                        dialogService.Show("Customer added");
                }
        }
}
```

Timeline

- November 2010 - Das Microsoft Patterns & Practices Team veröffentliche eine Anleitung für die Benutzung von MVVM unter dem Namen Prism v4 (http://microsoft.com/prism) .

Kritik

Eine Kritik am Muster kommt vom MVVM-Schöpfer John Gossman selbst,[10] welcher darauf hinweist, dass man bei der Implementierung simpler UI-Operationen mittels MVVM mit Kanonen auf Spatzen schießt. Er legt auch dar, dass für große Anwendungen die Verallgemeinerung der View komplizierter wird. Außerdem illustriert er, dass schlecht verwaltete Datenbindung zu einem erheblichen Speicherbedarf der Anwendung führen kann.

Quelloffene MVVM-Frameworks

- IdeaBlade: *Cocktail (http://cocktail.ideablade.com/)* .
- Josh Smith: *MVVM Foundation (http://mvvmfoundation.codeplex.com)* .
- Sacha Barber: *Cinch. (http://cinch.codeplex.com)* .
- Daniel Vaughan: *Calcium SDK (http://www.calciumsdk.net)* .
- Karl Shifflett: *Ocean (http://karlshifflett.wordpress.com)* .
- Tony Sneed: *Simple MVVM Toolkit (http://simplemvvmtoolkit.codeplex.com)* .
- Laurent Bugnion: *MVVM Light Toolkit (http://www.galasoft.ch/mvvm/getstarted/)* .
- Eye.Soft: *Hyperion SDK (http://hyperionsdk.codeplex.com)* .
- Lester Lobo: *CoreMVVM (http://coremvvm.codeplex.com/)* .
- Paul Betts: *ReactiveUI (http://www.reactiveui.net)* .
- Rob Eisenberg: *Caliburn (http://www.caliburnproject.org/)* .
- Rob Eisenberg: *Caliburn Micro (http://caliburnmicro.codeplex.com/)* .
- William e Kempf: *Onyx (http://wpfonyx.codeplex.com/)* .
- Peter O'Hanlon: *GoldLight (http://goldlight.codeplex.com/)* .
- jbe: *WPF Application Framework (WAF) (http://waf.codeplex.com)* .
- WPF Team: *WPF Model-View-ViewModel Toolkit (http://wpf.codeplex.com /wikipage?title=WPF%20Model-View-ViewModel%20Toolkit&referringTitle=Home)* .
- Michael L Perry: *Update Controls (http://updatecontrols.codeplex.com/)* .
- Steve Sanderson: *KnockoutJS (http://knockoutjs.com/documentation/observables.html)* .
- Geert van Horrik: *Catel (http://catel.codeplex.com)* .
- Jeremy Likness: *Jounce (http://jounce.codeplex.com)* .
- Xomega.Net: *Xomega Framework (http://xomfwk.codeplex.com)* .

- Marcus Egger, EPS Software: *Code Framework (http://codeframework.codeplex.com/)* .

Kostenlose MVVM-Frameworks

- Rhea NV (Visual Studio Partner): *Vidyano (http://www.vidyano.com)* .

Kommerzielle MVVM-Frameworks

- Intersoft Solutions (Visual Studio Partner): *ClientUI (http://www.clientui.com)* .

Siehe auch

- Model View Controller
- Model View Presenter

Einzelnachweise

1. Presentation Model (http://martinfowler.com/eaaDev/PresentationModel.html)
2. Steve Sanderson: *KnockoutJS (http://knockoutjs.com/documentation/observables.html)* .
3. Rafael Weinstein: *Google mdv (http://code.google.com/p/mdv/)* .
4. Google groups: *Thought: MVVM eliminates 99% of the need for ValueConverters (http://groups.google.com/group/wpf-disciples/browse_thread/thread/3fe270cd107f184f?pli=1)* .
5. Google groups: *Thought: MVVM eliminates 99% of the need for ValueConverters (http://groups.google.com/group/wpf-disciples/browse_thread/thread/3fe270cd107f184f /3ede55778f5a45dd)* .
6. Josh Smith: *WPF Apps With The Model-View-ViewModel Design (http://msdn.microsoft.com/en-us /magazine/dd419663.aspx)* .
7. John Gossman: *Tales from the Smart Client: Introduction to Model/View/ViewModel pattern for building WPF apps*. (http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx) .
8. Karl Shifflett: *Learning WPF M-V-VM. (http://karlshifflett.wordpress.com/2008/11/08/learning-wpf-m-v-vm/)* .
9. Pete Weissbrod: *Model-View-ViewModel Pattern for WPF: Yet another approach. (http://www.acceptedeclectic.com/2008/01/model-view-viewmodel-pattern-for-wpf.html)* .
10. John Gossman: *Tales from the Smart Client: Advantages and disadvantages of M-V-VM. (http://blogs.msdn.com/johngossman/archive/2006/03/04/543695.aspx)* .

Weblinks

- Martin Fowler: *Presentation Model (http://martinfowler.com/eaaDev/PresentationModel.html)* .
- Josh Smith: *WPF Apps With The Model-View-ViewModel Design Pattern (http://msdn.microsoft.com /en-us/magazine/dd419663.aspx)* .
- Josh Smith: *Advanced MVVM (http://advancedmvvm.com)* .
- Jason Dolinger: *A tutorial refactoring a traditional event-handling application to use MVVM and Commands. (http://blog.lab49.com/archives/2650)* .

- Kishor Aher: *Attaching Routed Event to Command (Zero code behind M-V-VM) (http://kishordaher.wordpress.com/2009/07/18/routedevent-to-command-action-behavior-blend-3/)* .
- John Gossman: *Tales from the Smart Client: Introduction to Model/View/ViewModel pattern for building WPF apps (http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx)* .

- Karl Shifflett: *Learning WPF M-V-VM. (http://karlshifflett.wordpress.com/2008/11/08/learning-wpf-m-v-vm/)* .
- Pete Weissbrod: *Model-View-ViewModel Pattern for WPF: Yet another approach. (http://www.acceptedeclectic.com/2008/01/model-view-viewmodel-pattern-for-wpf.html)* .
- John Gossman: *Tales from the Smart Client: Advantages and disadvantages of M-V-VM. (http://blogs.msdn.com/johngossman/archive/2006/03/04/543695.aspx)* .
- Jason Rainwater: *MVVM, providing the Association of View to ViewModel (http://torinth.spaces.live.com/blog/cns!DD5A60A80E18EE33!185.entry)* .
- Julian Dominguez: *Presentation Model pattern and the Composite Application Guidance (aka Prism) (http://blogs.southworks.net/jdominguez/category/presentation-model/)* .
- Nikhil Kotari: *ViewModel Pattern in Silverlight using Behaviors (http://www.nikhilk.net/Silverlight-ViewModel-Pattern.aspx)* .
- Glenn Block: *The spirit of MVVM (ViewModel), it's not a code counting exercise. (http://blogs.msdn.com/gblock/archive/2009/08/03/the-spirit-of-mvvm-viewmodel-it-s-not-a-code-counting-exercise.aspx)* .
- Rob Eisenberg: *Study in MVVM (http://www.caliburnproject.org/)* .
- Xomega Team: *Take MVVM to the Next Level with Xomega Framework (http://www.codeproject.com /KB/WPF/xomfwk.aspx)* .

- Geert van Horrik: *Catel - Part 0 of n: Why choose Catel? (http://www.codeproject.com/KB/WPF /CatelPart0WhyChoose.aspx)* .
- Geert van Horrik: *Catel - Part 1 of n: Data Handling the Way it Should (http://www.codeproject.com /KB/WPF/Catel.aspx)* .
- Geert van Horrik: *Catel - Part 2 of n: Using WPF Controls and Themes (http://www.codeproject.com /KB/WPF/Catel_Part2.aspx)* .
- Geert van Horrik: *Catel - Part 3 of n: The MVVM Framework (http://www.codeproject.com/KB/WPF /CatelPart3.aspx)* .
- Geert van Horrik: *Catel - Part 4 of n: Unit Testing with Catel (http://www.codeproject.com/KB/WPF /Catel_Part4.aspx)* .
- Geert van Horrik: *Catel - Part 5 of n: Building a WPF example application with Catel in 1 hour (http://www.codeproject.com/KB/WPF/Catel_Part5.aspx)* .
- Raffaele Garofalo: *UI Patterns tutorials including MVVM (http://blog.raffaeu.com/archive/2010/01 /31/ui-patterns-tutorials.-mvp-mvvm-and-composite-app-with-wpf.aspx)* .
- David Buksbaum: *Caliburn.Micro – Hello World (http://buksbaum.us/2010/08/01/caliburn-micro-hello-world/)* .
- David Buksbaum: *Caliburn.Micro the MEFtacluar (http://buksbaum.us/2010/08/04/caliburn-micro-the-meftacluar/)* .
- David Buksbaum: *How To Do Logging with Caliburn.Micro (http://buksbaum.us/2010/08/08/how-to-do-logging-with-caliburn-micro/)* .
- David Buksbaum: *Bootstrapping Caliburn.Micro with Autofac (http://buksbaum.us/2010/08 /20/bootstrapping-caliburn-micro-with-autofac/)* .
- David Buksbaum: *Caliburn.Micro ViewModel File Template for ReSharper (http://buksbaum.us /2010/08/27/caliburn-micro-viewmodel-file-template-for-resharper/)* .

Kategorien: Entwurfsmuster | Softwarearchitektur
