FHWS

Hochschule für angewandte Wissenschaften
Fachhochschule Würzburg-Schweinfurt

Bachelor-Arbeit

Entwicklung einer Multi-Plattform-Anwendung: Serverseitige Konzeption und prototypische Umsetzung

vorgelegt an der
Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt
in der Fakultät Informatik und Wirtschaftsinformatik zum Abschluss eines Studiums
im Studiengang Informatik

Studienschwerpunkt: Medieninformatik

Angefertigt in Firma:	Solus-Software GmbH Würzburg, betreut von Markus Harzdorf, Dipl. Informatiker (FH)
Erstprüferin:	Prof. Dr. Isabelle John
Zweitprüfer:	Prof. Dr. Karsten Huffstadt
Abgabetermin:	12. Juli 2013
Eingereicht von:	Johannes Krenig aus Randersacker

Inhaltsverzeichnis

1 MVVM

1.1 WPF und MVVM-Entwurfsmuster

1.1.1 WPF

Mit .NET Framework 3.0 wurde WPF als Programmierschnittstelle für Windows Anwendungen eingeführt. Das rendern und zeichnen von von Inhalten basiert auf DirectX. Entwickler und GUI-Designern können damit ansehnliche Benutzeroberflächen erstellen. In Gegensatz zur altbekannten WinForm-API trennt die WPF die Präsentations- und Geschäftslogik nach strengen Richtlinien, die auf der Auszeichnungssprache XAML basiert. XAML ist wie der Name schon impliziert an XML angelehnt und ist in der Syntax sehr ähnlich.

Die wichtigsten Eigenschaften einer WPF-Anwendung im Überblick:

- Design mit der Auszeichnungssprache XAML, alternativ auch Ausprogrammiert möglich.
- Unterstützung von 2D und 3D Grafiken
- Die Ausgabe ist vektorbasiert anstatt pixelbasiert. Daraus erfolgt eine bessere Skalierbarkeit der Bildschirmausgabe.
- Vielfältige Datenbindungsmöglichkeiten
- Grafikberechnungen auf der GPU anstatt CPU

Die strikte Trennung von Oberflächengestaltung und Code ermöglicht es, dass Programmentwickler und Oberflächendesigner vollkommen unabhängig voneinander Code und Benutzeroberfläche entwickeln können.

1.1.2 MVVM

Es wurde bei Microsoft von John Gossman als Variante des MVP Entwurfsmuster auf seinem Blog vorgestellt. Hintergrund war die Entwicklung moderner UIs, wie der WPF und Silverlight jeweils von Microsoft, mit seinen neuen Möglichkeiten zur Erstellung von Benutzeroberflächen. Es gibt viele Vorteile durch das Pattern, allerdings erhöht es auch die Komplexität einer Anwendung. In besonders großen Anwendungen ist das MVVM sinnvoll, in kleiner Anwendungen, ist es aufgrund des Overheads nicht wirklich zu empfehlen. Die Abkürzung MVVM steht wie bei MVC und MVP für die Komponenten des Entwurfsmusters (Siehe Abbildung 1.1). Die Langform Model View ViewModel ist auf den ersten Blick ungewöhnlich und bedarf der Erklärung.

- Model: Datenzugriffsschicht für die Inhalte, dem Benutzer angezeigt und von ihm manipuliert werden. Dazu benachrichtigt es über Datenänderungen und führt eine Validierung der vom Benutzer übergebenen Daten durch. Hierdurch wird vor allem in der View der Code-Behind minimiert.
- View: Alle durch die GUI angezeigten Elemente. Es bindet sich an Eigenschaften des ViewModel, um Inhalte darzustellen und zu manipulieren sowie Benutzereingaben weiterzuleiten. Durch die Datenbindung ist die View einfach austauschbar und ihr Code-Behind gering.
- ViewModel: beinhaltet die UI-Logik (Model der View) und dient als Bindeglied zwischen View und obigem Model. Einerseits tauscht es Information mit dem Model aus, ruft also Methoden oder Dienste auf. Andererseits stellt es der View öffentliche Eigenschaften und Befehle zur Verfügung. Diese werden von der View an Steuerelemente gebunden, um Inhalte auszugeben bzw. UI-Ereignisse weiterzuleiten. Insgesamt wird CRUD ermöglicht. Das ViewModel darf dabei keinerlei Kenntnis der View besitzen.

Die vom MVVM genutzte funktionale Trennung und Datenbindung des MVC wird dazu genutzt, die bereits beschriebene lose Kopplung zu erreichen. Diese ist jedoch im MVVM um einiges stärker verwirklicht als in MVC und MVP. Ein Vorteil des MVVM in der Entwicklung von Anwendungen ist zudem die bessere Einbindung der UI-Logik in Unit Tests. Aufgrund der Tatsache, dass Sie beim MVVM die Logik und das Command in der ViewModel-Klasse haben möchten, haben sich bei diesem Entwurfsmuster Commands wie das ActionCommand gegenüber den RoutedCommands durchgesetzt. Ein ActionCommand erlaubt beim Instanziieren direkt die Angabe eines Delegates, der auf eine Methode im ViewModel zeigen kann. MVVM in Verbindung mit WPF verwendet üblicherweise eine Implementierung mit Delegates basierend auf ICommand-Interfaces.

1.2 Problem

1.3 Lösung

Eine bessere Trennung der Schichten ermöglicht größere Flexibilität

1.4 Konsequenz

1.5 Beispiel

- Problem - Lösung - Konsequenzen - Ãœbung / Beispiel

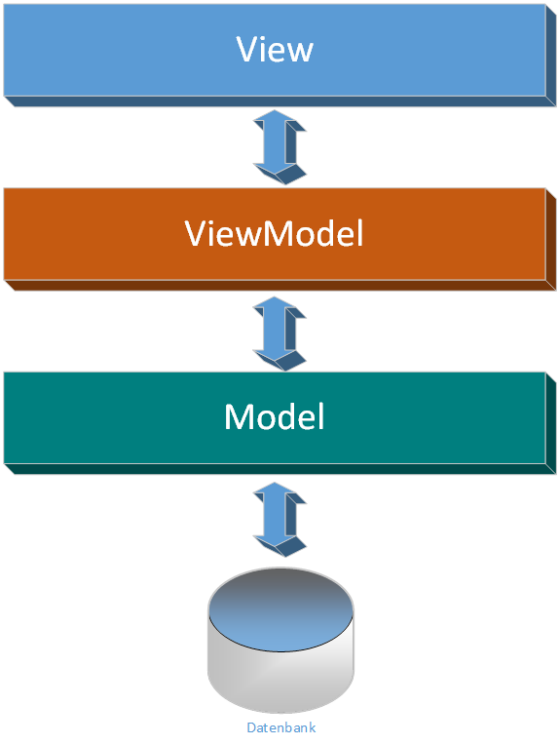

Abbildung 1.1: MVVM-Entwurfsmuster

Literaturverzeichnis

Listings

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis