

The background features a complex network of thin, light gray lines connecting various points, creating a web-like structure. Scattered throughout are numerous triangles of different sizes and orientations, some with solid outlines and others with dashed or dotted lines. The overall aesthetic is clean, modern, and technical.

Clean Code & Design Pattern

Tag 5

Zeitplan



12:00
Start

Pause

13:15 & 14:45



16:00
Ende

The background features a complex network of thin, light gray lines and dots, primarily concentrated on the left side, creating a web-like or molecular structure. Scattered across the entire background are numerous triangles of various sizes and orientations, some outlined in a slightly darker gray than the background. The overall aesthetic is minimalist and technical.

Review Tag 4

Wrapper

- Decorator
- Adapter
- Facade



Design Pattern / Clean Code Prinzipien anwenden

- Tagging Interface Pattern
- (Lambda) Factory beim Resolve
- Eigenes Interface statt Abhängigkeit
- Singleton / Transient (/ Scoped)



Dependency Injection

- Microsoft Dependency Injection Extensions
 - a. Alternativen sind u.a. StructureMap oder Autofac oder ... oder ... oder ...
- Mapping Interface → Klasse
- Registrieren von Klassen
- Registrieren von Instanzen
- On-Demand Erzeugung



The background features a complex network of thin grey lines connecting various points, forming a web-like structure. Scattered throughout are numerous triangles of different sizes and orientations, some with solid outlines and others with dashed or dotted outlines. The overall aesthetic is technical and modern.

MVC, MVP, MVVM

...oder wie trenne ich die Präsentation von der Logik

“Dark Ages” – User EditForm

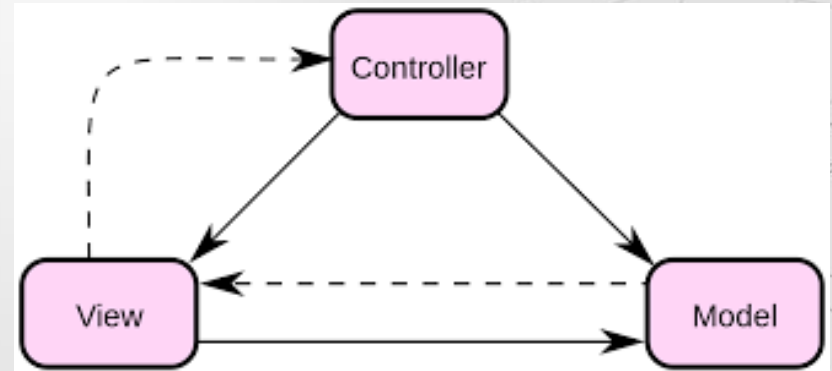
- OnInitialize()
 - a. Datenbankverbindung aufbauen + SQL + Daten abrufen
 - b. Füllen der Controls mit Daten
- OnXyChanged()
 - a. Validierung
- OnButtonPress()
 - a. Datenbankverbindung aufbauen + SQL + Daten speichern

→ Riesige Klassen mit viel Technik- und Fachwissen in einer Klasse



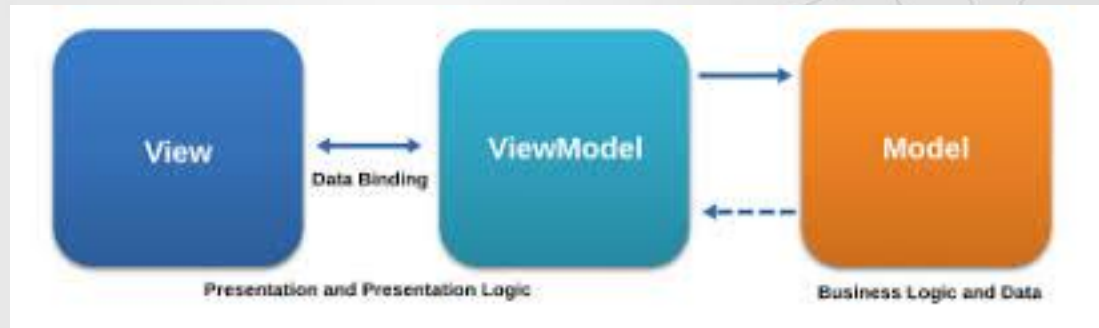
MVC

- Insbesondere in Web Technologien
- Controller als Einstiegspunkt für den Request
- Model, View
- Controller füllt die Daten in die View



MVVM

- Im Zusammenhang mit WPF
- Model, View
- ViewModel für die Kommunikation



Die Idee 💡 in WPF

- WPF hat XAML Frontend (View) und C# Backend (ViewModel)
- Trennung von Präsentation (View) und Daten (ViewModel)
- Datenbindung und Events für die Kommunikation
- Nachteile
 - a. EventHandler mit "EventArgs" also View-bezogene Klassen
 - b. Zugriff von ViewModel auf die View
 - c. Eigene Models für den DataContext+Bindung nicht stringent



Die Lösung für WPF

- Separate ViewModel Klasse
- Datenbindung im Backend der View
- Kein Zugriff vom ViewModel auf die UI
- Framework (z.B. Caliburn Micro) für die Magie (Datenbindung)



Frage

Was ist das Model?

Was steckt da drin?
Was "kann" das Model?



The background features a complex network of thin grey lines and dots, forming a web-like structure. Scattered throughout are various triangles of different sizes and orientations, some with solid outlines and others with dashed or dotted lines. The overall aesthetic is modern and technical.

Unsere kleine Applikation

Aufgabe

DI implementieren

Microsoft DI Framework



Aufgabe

Logging implementieren

Console Logger
Trace Logger



Aufgabe

EditForm

View anlegen



Demo

WinForms Beispiel

Datenbindung



Aufgabe

EditForm

ViewForm anlegen
Datenbindung anlegen



Aufgabe

Icommand implementieren

HelloWorldCommand
"RelayCommand"



Aufgabe

BindingSource

Binding Source für das DataBinding

