Wzorzec projektowy MVVM

Wojciech Szlosek, Klaudia Juśko, Konrad Nowak

23 maja 2021

Spis treści

1 Wstęp

1.1 Czym jest MVVM?

Wzorzec Model-View-ViewModel (MVVM), to wzorzec projektowy opierający się na wydzieleniu odpowiednich warstw w systemie, w celu podziału zadań oraz zmniejszenia zależności pomiędzy klasami.

Wyróżniamy następujące warstwy:

MODEL

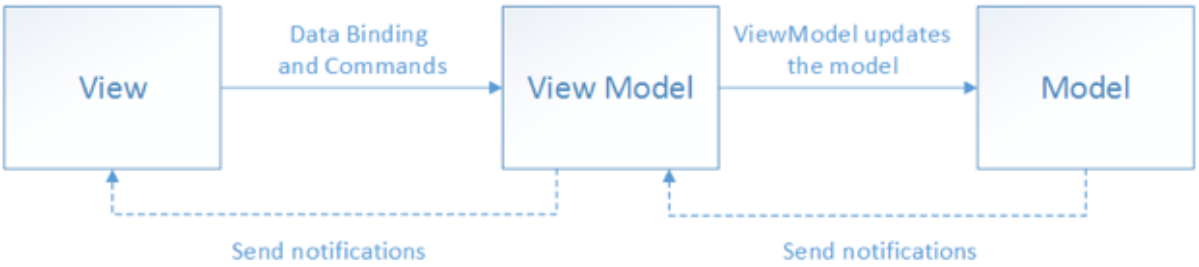
Warstwa widoku komunikuje się z warstwą modelu widoku za pomocą komend oraz wiązania danych. O komendach można myśleć jak o metodach zdarzeniowych podpiętych do poszczególnych kontrolek w definicji widoku. Wiązanie danych polega na odwoływaniu się do właściwości podpiętego modelu widoku jako źródła danych.

VIEW

Warstwa widoku zawiera wyłącznie interfejs użytkownika i jest odpowiedzialny wyłącznie za interakcje z użytkownikiem. Zgodnie z koncepcją MVVM kod ten powinien być ograniczony do minimum.

VIEW MODEL

Warstwa modelu widoku, która łączy model z widokiem. Udostępnia dane dla widoku i odpowiada za wymianę informacji z modelem - pobiera dane oraz dokonuje ich aktualizacji.



Rysunek 1: MVVM pattern scheme.

2 Zalety i wady

2.1 Zalety

- Programiści mogą tworzyć testy jednostkowe dla modelu i modelu widoku, bez korzystania z widoku; ponadto dla modelu widoku mogą one mieć dokładnie te same funkcje, które są używane przez widok.
- Projektanci i deweloperzy mogą działać niezależne (i równolegle) na składnikach wzorca podczas procesu tworzenia. Przykładowo: podczas gdy deweloperzy korzystają z modelu widoku i składników modelu, projektanci tworzą widok.
- Model widoku działa jako karta klas modelu i pozwala uniknąć wprowadzania istotnych zmian w kodzie modelu (czasami jest to pożądane).
- Prosty do zrozumienia i zaimplementowania wzorzec projektowy, tworzący przejrzysty obraz całego programu.
- Aktualizacje wprowadzane w interfejsie użytkownika nie wymagają przeprojektowania całej aplikacji, ponieważ warstwa widoku jest odseparowana od logiki biznesowej.

2.2 Wady

- Implementacja MVVM dla prostych operacji UI jest nieopłacalna, natomiast dla większych aplikacji stworzenie ogólnej warstwy ViewModel staje się czasem bardzo trudne.
- Wiązanie danych jest bardzo zasobożerne, szczególnie w bardzo rozbudowanych aplikacjach.
- Warstwa View i ViewModel zawiera powtarzalne fragmenty kodu, co w przypadku wprowadzenia zmian powoduje problemy z utrzymaniem aplikacji.

3 Przypadki użycia

- Najczęstszym zastosowaniem modelu MVVM jest programowanie GUI (Graphical User Interface). Służy do paradygmatu zwanego programowaniem sterowania zdarzeniami - poprzez oddzielenie warstwy widoku od warstwy logiki biznesowej.
- MVVM używany jest w Windows Presentation Foundation (WPF) przy użyciu języka XAML. Pliki XAML są powiązane z warstwą ViewModel. W ten sposób warstwa View jest odpowiedzialna wyłącznie za prezentację, a warstwa ViewModel odpowiada za aktualizację i zarządzanie danymi modelu.
- Używany jest również w bibliotece KnockoutJS w języku JavaScript.

4 Przykład

4.1 Jak wyglądają warstwy?

MODEL

W naszym przykładowym programie, warstwa ta jest napisana w języku C#. Pojedyncza klasa przechowuje dane o osobie, każda ze zmiennych ma swoją funkcję ustawiającą i zwracającą wartość.

VIEW

W naszym przykładowym programie, warstwa ta jest napisana w kodzie XAML, zawiera ona tylko przyciski, textboxy oraz listę. Użytkownik może wprowadzać dane i je modyfikować, lecz sam kod XAML tego nie robi, tylko przekazuje odpowiednią instrukcję warstwę dalej.

VIEW MODEL

W naszym przykładowym programie, warstwa ta jest za stworzenie obiektów klasy osób oraz decyduje za akcje, jakie przekazał jej użytkownik. Przykładowo, jeśli użytkownik kliknie przycisk aktualizuj, klasa ta zadecyduje, że trzeba w odpowiednich obiektach zmienić wartości zmiennych i wyświetlić je ponownie na ekran.