MVVM gombok és parancsok

# Projekt létrehozás

1. Hozza létre a RectangleProject-et
2. A projektbe hozza létre a Models mappát
3. A mappába hozza létre a Rectangle osztályt.

# Model réteg

public class Rectangle

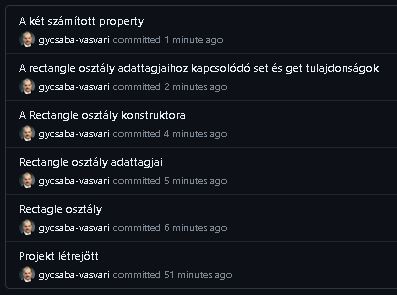
{

private double \_sideA;

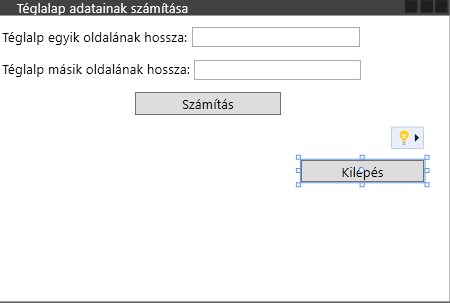
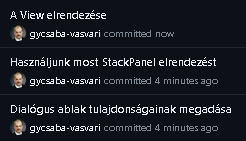
private double \_sideB;

}

A fejezethez tartozó commitok:



View réteg



# ViewModel réteg



Az MVVM-minta használatának előnyei:

* Ha egy meglévő modell implementációja magában foglalja a meglévő üzleti logikát, annak módosítása nehéz vagy kockázatos lehet. Ebben a forgatókönyvben a nézetmodell (ViewModel) megakadályozza, hogy jelentős módosításokat hajtson végre a modellkódon.
* Az alkalmazás felhasználói felülete (view) a nézetmodell és a modellkód érintése nélkül újratervezhető, feltéve, hogy a nézet teljes mértékben XAML-ben vagy C#-ban van implementálva. Ezért a nézet új verziójának a meglévő nézetmodellel (ViewModel) kell működnie.
* A Tervező és a fejlesztők egymástól függetlenül és egyidejűleg dolgozhatnak az összetevőiken a fejlesztés során. Tervező a nézetre összpontosíthatnak, míg a fejlesztők a nézetmodellen és a modell összetevőin dolgozhatnak.

## Nézet (View)

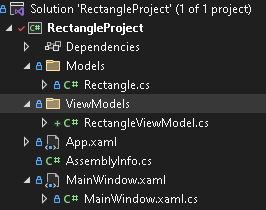
A nézet feladata annak meghatározása, hogy a felhasználó milyen struktúrát, elrendezést és megjelenést lát a képernyőn.

ViewModel

A nézetmodell olyan tulajdonságokat és parancsokat implementál, amelyekhez a nézet adatkötést végezhet, és a változásértesítési eseményeken keresztül értesíti az állapotváltozások nézetét. A nézetmodell által biztosított tulajdonságok és parancsok határozzák meg a felhasználói felület által kínált funkciókat, de a nézet határozza meg a funkció megjelenítésének módját.

### ViewModel réteg fejlesztése

Hozzuk létre a szükséges mappát és osztályt a ViewModel rétegben:



using RectangleProject.Models;

namespace RectangleProject.ViewModels

{

public class RectangleViewModel

{

private Rectangle \_rectangle;

}

}

A nézet osztály egy négyzet adataival foglalkozik. Ügyeljünk a szükséges using-ra, a megfelelő névteret töltsük be!

Fejlesszük a ViewModel réteget:



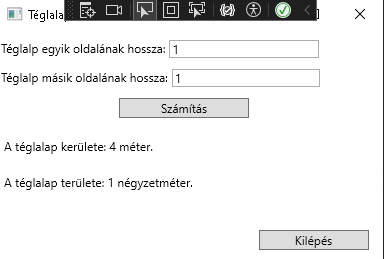
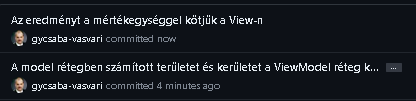
# View réteg

Az elkészített ViewModelben lévő property-ket kössük a View-n!



# Eredmény megjelenítése

Az eredmények megjelenítésére string típust használunk, hogy a mértékegységet is meg tudjuk jeleníteni. A kódok megírása után a program már megjeleníti a kezdő adatok esetén az eredményeket.



# Számítás elvégzése Button segítségével

A számítások elvégzéséhez a ViewModel réteget a megadott ViewModelBase osztálytól örököltetjük:

class RectangleViewModel : ViewModelBase

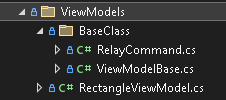
{

private Rectangle rectangle;

amely megvalósítja

OnPropertyChanged(nameof(Perimeter));

metódust:



Használjuk még a megadott RelayCommand osztályt a parancsok létrehozására:

CalculationCommand = new RelayCommand(execute => Calculation());

…

public RelayCommand CalculationCommand { get; private set; }

…

public void Calculation()

{

OnPropertyChanged(nameof(Perimeter));

…

}

A megvalósított commitok:

