RVA

MVVM

Sve zajedno

UI Klasa

XAML

Biznis logika

Skladištenje podataka

Podeljeno

UI Klasa

XAML

Biznis logika

Skladištenje podataka

Razdvajanje uloga

Nakon razdvajanja uloga klase za korisnički interfejs su mnogo jednostavnije, pored XAML, imaju minimalnu neohodnu količinu koda

Biznis logika i rad sa uskladištenim podacima su izdvojeni u posebne klase, što dozvoljava različite predstave podataka interno, kao i u korisničkom interfejsu i omogućava veću fleksibilnost

MVVM obrazac

MVVM (Model-View-ViewModel) obrazac se može koristiti na svim XAML platformama.

MVVM obrazac je aplikacioni obrazac koji razdvaja korisnički interfejs od biznis logike “ispod haube”

Olakšava nezavisno razvijanje aplikacije i njenog korisničkog interfejsa

Varijanta MVC (model-view-controller) obrasca

Prednosti

Razdvajanje različite vrste koda (funkcionalnosti), olakšava izmenu svake pojedinačno

Olakšava pisanje unit testova

Dobro prilagođen XAML platformama zbog data bindinga

Povećava ponovnu iskoristivost koda (isti model se može koristiti uz drugačije poglede na model)

Mane

Potencijalno prekomplikovan za jednostavne UI

U služenim slučajevima, dizajniranje pogleda na model može biti teško

Debagovanje može biti otežano ukoliko ima složenih data bindinga

Delovi MVVM

Model - pruža reprezentaciju entiteta korišćenih u aplikaciji, nezavisnu od korisničkog interfejsa; arhitektura modela treba biti optimizovana za biznis logiku, nezavisno od predstave podataka u korisničkom interfejsu

Pogled (view) - korisnički interfejs; prikazuje informacije korisniku i pokreće događaje (fires events) kao odgovor na korisničke interakcije

Pogled na model (viewmodel) - most između pogleda i modela; svaki pogled ima odgovarajući pogled na model; pogled na model preuzima podatke od modela i transformiše ih u format koji pogled zahteva; obaveštava pogled kada dođe do promena u modelu i ažurira model kao odgovor na događaje koje pogled pokreće

Pogled (View) klasa

Pogled je vizualni element poput prozora, strane ili korisničke kontrole. Zadatak pogleda je da definiše strukturu i izgled onoga što korisnik vidi na ekranu. U idealnom slučaju u kodu modela se nalazi samo konstruktor koji poziva InitializeComponent() metodu. U nekim slučajevima može se dodati kod za UI logiku koji je teško implementovati u XAML. Pogledi su uglavnom klase izvedene iz Control ili UserControl klasa. Pogled se povezuje sa pogledom na model (viewmodel), preko DataContext propertya. Pogled može da definiše ponašanje data bindinga između pogleda i pogleda na model (na primer može da definiše konvertore vrednosti koji formatiraju podatke koji se ispisuju na korisnički interfejs)

Pogled na model (Viewmodel)

Pogled na model je ne vizuelna klasa koja enkapsulira prezentacionu logiku i podatke za pogled. Tipično implementira INotifyPropertyChanged i/ili INotifyCollectionChanged interfejse. Pogled na model implementira propertye i komande na koje se pogled može nakačiti i obaveštava pogled o promenama. Pogled na model je odgovoran za saradnju između pogleda i bilo koje klase modela. Pogled na model može da konvertuje i upravlja podacima iz modela tako da budu prilagođeni za pogled. Za potrebe validacije podataka može da implementuje IDataErrorInfo i INotifyDataErrorInfo interfejse.

Model klasa

Model je ne vizualna klasa koja enkapsulira podatke i biznis logiku aplikacije. Pod biznis logikom se smatra bila koja aplikaciona logika koja se odnosi na upravljanje aplikacionim podacima i postavljanjem pravila i ograničenja nad njima. Definiše strukture podataka i sve prateće operacije. Obično podržava INotifyPropertyChanged i INotifyCollectionChanged interfejse, a kolekcije se obično izvode iz ObservableCollection<T> klase. Model može podržati validaciju podataka i rukovanje greškama preko IDataErrorInfo i INotifyDataErrorInfo interfejsa.

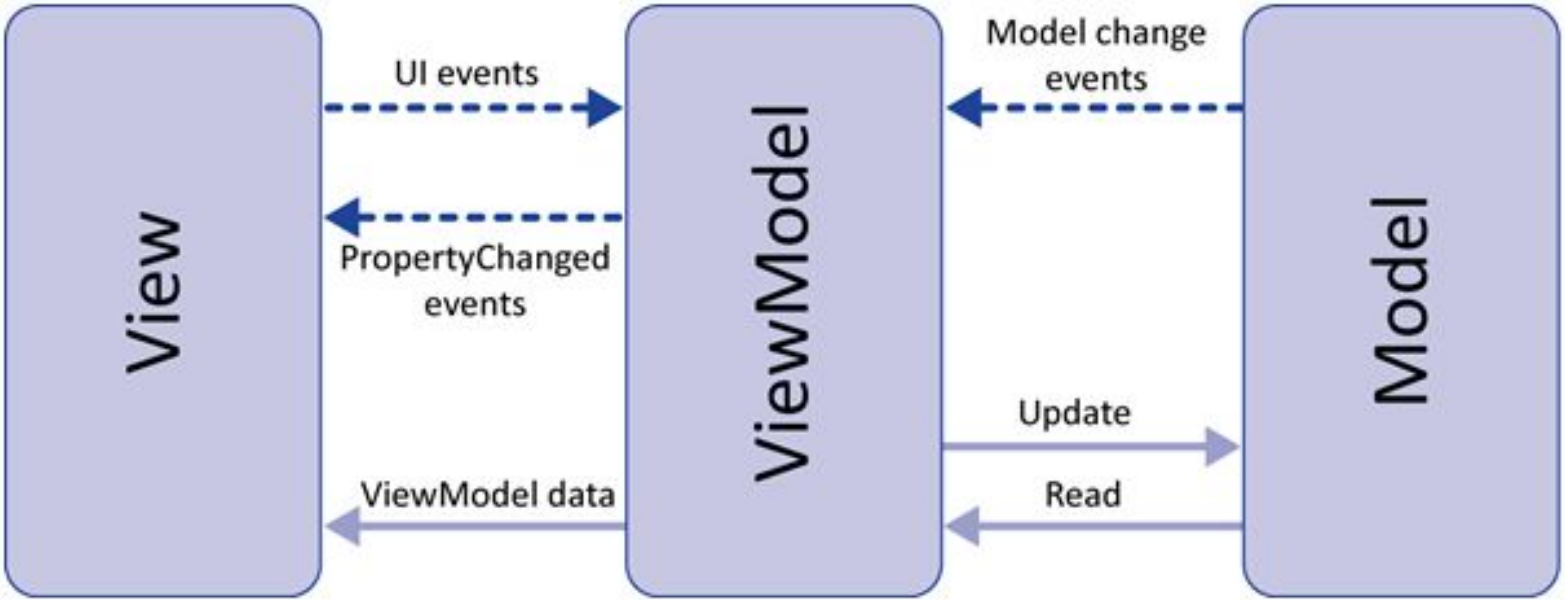
Interakcija

Pogled (view) klasa generiše događaje kao odgovore na korisničke interakcije, kojima upravlja odgovarajuća pogled na model (viewmodel) klasa; pogled ne zna kako se događaji obrađuju

Pogled na model klasa određuje da li je potrebno ažurirati model kao odgovor na interakciju sa korisnikom

Model obaveštava pogled na model ako dođe do promena u uskladištenim podacima

Pogled na model obaveštava pogled kada se podaci promene


View
UI events
PropertyChanged events
ViewModel
Model change events
Update
Read
ViewModel data
Model