Gestione Esercizi | Diario di lavoro - 24.10.2019

Gabriele Alessi

Canobbio, 24.10.2019

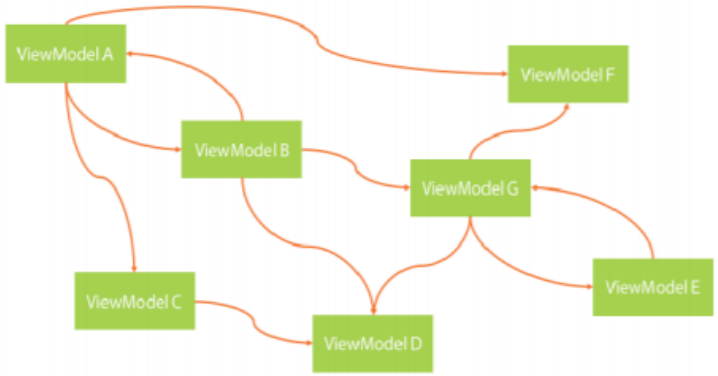
Lavori svolti

Durante questa giornata ho continuato a lavorare sullo sviluppo delle interfacce del programma su Visual Studio. In particolare mi sto concentrando sulla comunicazione tra i ViewModels in modo da consentire il passaggio dei dati tra essi. Per fare ciò ho ricevuto del materiale da parte del docente responsabile di progetto:

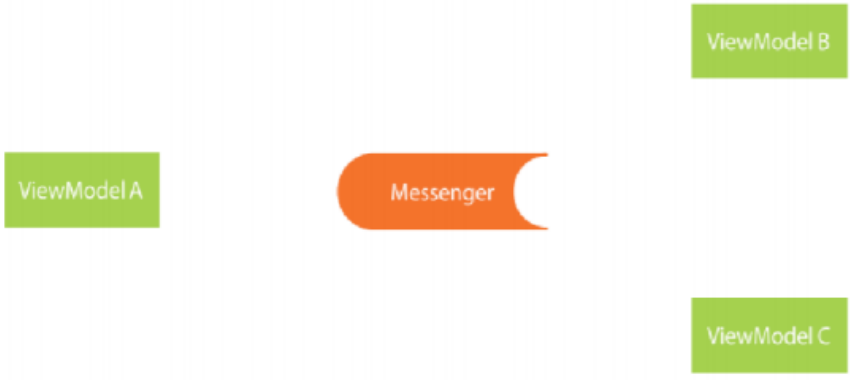
Comunicazione tra ViewModel

Un aspetto importante è la comunicazione tra i vari ViewModel per passare informazioni tra le View senza avere alcun HardLink.

Per risolvere questo problema ci si affida a un Messenger (o Mediator). Si trova in mezzo nella comunicazione tra più ViewModels che tutti conoscono e comunicano attraverso lui.



Per esempio si registra ViewModelA con il Messenger e si definisce che potrebbe inviare un messaggio di un certo tipo. ViewModelB e ViewModelC sono interessati a quello che "dice" ViewModelA. Questi "informano" il Messenger che sono interessati ad un certo tipo di messaggio e non appena riceve un messaggio del tipo coinvolto, questi lo invierà agli interessati.



Quindi ho implementato questo sistema e ho concluso `AboutViewModel` e `BenvenutoViewModel`.

```
public class AboutViewModel : BindableBase
{
    private BenvenutoViewModel benvenutoViewModel;

    public IDelegateCommand BenvenutoCommand { get; set; }

    public AboutViewModel() { RegisterCommands(); }

    private void RegisterCommands()
    {
        benvenutoViewModel = new BenvenutoViewModel();
        BenvenutoCommand = new DelegateCommand(OnBenvenuto, CanBenvenuto);
    }

    private void OnBenvenuto(object obj)
    {
        Messenger.Default.Send<BindableBase>(benvenutoViewModel;
    }
    private bool CanBenvenuto(object arg) { return true; }
}
```

```
private void OnImpostazioniBase(object obj)
{
    Messenger.Default.Send<BindableBase>(impostazionibaseViewModel);
}
private bool CanImpostazioniBase(object arg) { return true; }
private void OnEsercizio(object obj)
{
    Messenger.Default.Send<BindableBase>(esercizioViewModel);
}
private bool CanEsercizio(object arg) { return true; }
private void OnProva(object obj)
{
    Messenger.Default.Send<BindableBase>(provaViewModel);
}
private bool CanProva(object arg) { return true; }
```

Per la prossima giornata dovrei procedere con i ViewModels in quanto ho risolto i problemi riscontrati e ho trovato i metodi corretti per implementare il sistema.

Orario	Lavori svolti
13:15 - 16:30	Implementazione e Documentazione

Problemi riscontrati e soluzioni adottate

Ho riscontrato problemi durante l'implementazione della comunicazione tra ViewModels tramite il `Messenger`. Questo perché non sono sicuro di come debbano essere implementati i metodi `Send` e `Register` (in che classi e che parametri passare e come gestire il tutto nelle Views).

Alla fine ho risolto correggendo i Commands grazie all'aiuto del docente:

```
private void OnBenvenuto(object obj)
{
    Messenger.Default.Send<BindableBase>(benvenutoViewModel);
}
```

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.