Gestione Esercizi | Diario di lavoro - 08.10.2019

Gabriele Alessi

Canobbio, 08.10.2019

Lavori svolti

Oggi ho iniziato con l'implementazione dell'App WPF del progetto su Visual Studio. Dopo aver concluso la libreria delle classi (cioè le classi che gestiscono i dati e il database) mi sono consultato con il docente responsabile per controllare ciò e per vedere come cominciare con il resto del progetto.
Come prima cosa ho capito l'utilità del metodo `Where()`:

```
/// <summary>
/// Metodo utile per filtrare dei dati.
/// </summary>
/// <param name="predicate">Espressione filtrante.</param>
/// <returns>I dati filtrati.</returns>
public virtual IQueryable<T> Where(Expression<Func<T, bool>> predicate)
{
    return context.Set<T>().Where(predicate);
}
```

Quindi ho cominciato con lo sviluppo dell'App WPF partendo dai ViewModels. Prima di tutto ho reperito le basi Helper di MVVM dai vecchi progetti. Ad esempio `BindableBase` è la superclasse di ogni altro ViewModel in quanto contiene i metodi base sulla gestione dei dati e il collegamento con le View.

```
public class ClasseListViewModel : BindableBase
{
    /// <summary>
    /// Insieme dinamico dei dati delle classi.
    /// </summary>
    public ObservableCollection<Classe> Classi { get; set; }

    /// <summary>
    /// Metodo costruttore del ViewModel.
    /// </summary>
    public ClasseListViewModel()
    {
        ClasseDbRepository repo = new ClasseDbRepository(new AppDbContext());
        Classi = new ObservableCollection<Classe>(repo.Get());
    }
}
```

```
public abstract class BindableBase : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged = delegate { };

    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChangedEventHandler handler = PropertyChanged;
        if (handler != null)
        {
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }

    protected virtual void SetProperty<T>(ref T member, T value,
[CallerMemberName] string propertyName = null)
    {
        if (object.Equals(member, value))
            return;
        member = value;
        OnPropertyChanged(propertyName);
    }

    protected void OnPropertyChanged<T>(Expression<Func<T>> propertyExpression)
    {
        var body = propertyExpression.Body as MemberExpression;
        var expression = body.Expression as ConstantExpression;
        PropertyChanged(expression.Value, new
PropertyChangedEventArgs(body.Member.Name));
    }
}
```

Ora che ho capito il sistema dei ViewModel dovrei riuscire a concludere questo capitolo per questa settimana.

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

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Abbastanza in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Implementazione, documentazione.