18. Návrhové vzory (Design patterns)

Návrhový vzor nabízí obecné řešení nezávisle na použitém jazyku. Návrhář by měl správně používat tyto vzory s ohledem na konkrétní SW, podmínky a omezení.

Popis tří vzorů, které jsme se učili:

Singleton

Singleton se využívá, když je třeba zajistit pouze jednu instanci konkrétního typu v celém programu (většinou se používá pro práci s databází nebo hlavním menu). Hlavní myšlenkou je zamezení více instancí (private konstruktorem). Vytvoření se metoda, která bude vracet právě jedinou instanci (pokud nebude existovat zavolá právě private konstruktor). **Thread-safe vzor**:

```
public class Singleton
    {
        private static readonly object padlock = new object();

    Singleton() // private constructor
    {
    }

    private static Singleton instance = null;
    public static Singleton Instance
        {
            get {
                lock (padlock) { // thread-safe method with lock object
                    if (instance == null) {
                        instance = new Singleton();
                    }
                    return instance;
                }
            }
        }
    }
```

Dependency Injection

Dependency injection použijeme tehdy, pokud chceme navrhnout třídu, která bude mít různá specifika, podle kterých je budeme rozeznávat a pracovat s nimi. (např. Evropan a Asiat jsou lidé, budou mít jména, ale každý bude muset splňovat jiné podmínky).

V programování využijeme interface. Každá osoba bude mít v konstruktoru validátor, který bude zvolen podle potřeby. Aby taková osoba vznikla musí tedy projít námi vybraným validátorem.

Model-View-ViewModel

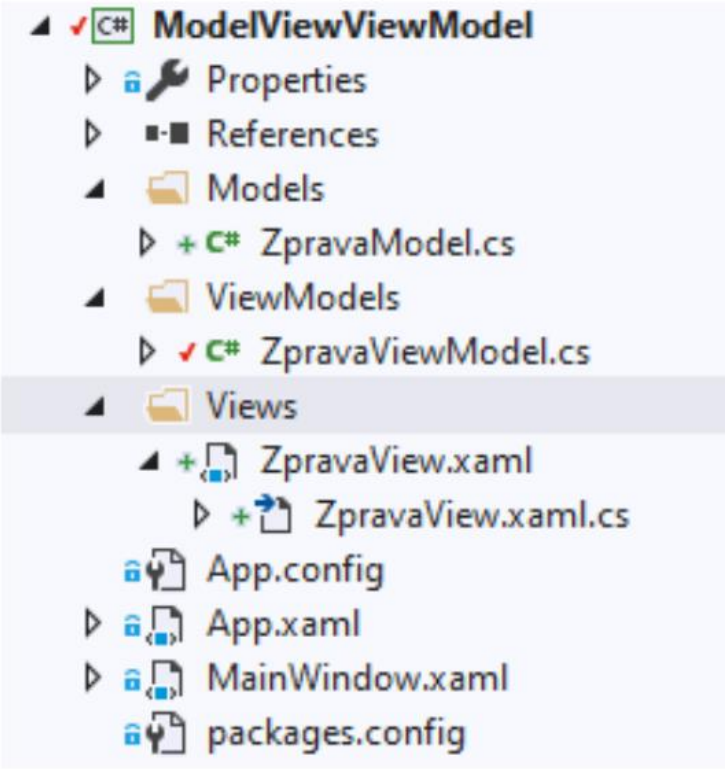
Zde se snažíme o rozdělení grafické stránky programu od funkcionální (např. chceme ověřit funkci Sečti aniž bychom to spojili s grafickým rozhraním programu). Hlavní myšlenka je důsledně separovat prezentaci informace a informaci samotnou.

Model – má pouze v sobě data (většinou nějaká databáze, úložiště), základní objekty

View – zná pouze uživatelské rozhraní, jde o čistou prezentaci (tlačítka, okna, labely)

ViewModel – obsahuje výpočty, transformace, validace, hlavní logiku programu

Konvencí v tomto modelu je vytvoření složek (se jmény jako Model, View, ViewModel) a do nich vkládání pouze informací a kódu, který koresponduje s informacemi napsané víš ↑.



Data Binding

Data binding je technika spojení (synchronizace) dat s interfacem. Napojíme totiž element v GUI přímo na proměnou, a když se změní tato proměnná, rovnou se nám to promítne v GUI. To v C# díky interface *INotifyPropertyChanged* a v jazyce XAML keywordu *Binding*. Potom lze také nastavit kdy se změní, jakým směrem a další parametry.