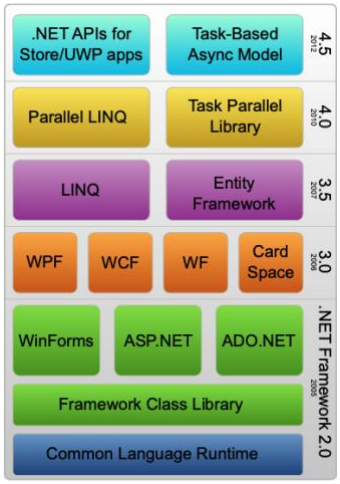
Architektury .NET

Prostředky architektury .NET pro cílové platformy, MVVM, Binding, Observer, událostmi řízené programování

Joseph Soukup – P4

.NET

- .NET framework (rok 2002) pro tvorbu aplikací pro operační systém MS Windows
- Dnes již přichází ve verzích .NET Core (rok 2016) a .NET Standart => multiplatformní
- 

- Podporovaný jazyky:
    - C Sharp
    - F Sharp
    - Visual Basic
    - IronPython (Python verze 2)

Prostředky architektury .NET pro cílové platformy

CLR

- Common Language Runtime
- Společné virtuální prostředí, ve kterém běží aplikace
- Jednotlivé jazyky jsou zkompilovány do CIL (Common Intermediate Language)
    - Následně interpretován
- Stará se o správu paměti, typovou kontrolu, výjimky, vlákna a bezpečnost

FCL

- Framework Class Library
- Soubor knihoven integrovaný do CLR
- Poskytování důležitých knihoven pro přístup ke:
    - Konzoli
    - Souborovém systému
    - Siťové konektivitě
- V případě .NET frameworku (Windows) je poskytnuta řada knihoven, či celých frameworků k tvorbě a správě okkeních aplikací (WPF, Windows Forms)

LINQ

- Language Integraded Query

- Jednotný jazyk pro dotazování nad kolekcemi
- Syntaxe připomíná SQL
- Příklad:
  - ```
    var results =  from c in SomeCollection
                   where c.SomeProperty < 10
                   select new {c.SomeProperty, c.OtherProperty};

    foreach (var result in results)
    {
            Console.WriteLine(result);
    }
    ```
- Linq v rámci vlastního jmenného prostoru poskytuje tyto rozšiřující metody, které lze volat standartním způsobem:
  - ```
    var results =
         SomeCollection
            .Where(c => c.SomeProperty < 10)
            .Select(c => new {c.SomeProperty, c.OtherProperty});

    results.ForEach(x => {Console.WriteLine(x.ToString());})
    ```

Jiné komponenty

- Další komponenty jsou prostředky pro tvorbu paralelních a asynchronních aplikací
- Velkým frameworkem je také ASP.NET pro vývoj webových aplikací

MVVM

- Model View ViewModel
- Softwarová architektura
- Cílem je oddělení logiky a uživatelského rozhraní
- Model:
  - Veškerá logika
  - Popisuje data, se kterými aplikace pracuje
  - Code-First Entity => třídy, které se mapují do databáze jsou Modely
  - Webová služba => třídy, které jsou vygenerovány, jsou taktéž Modely
  - Model nesmí o stavu ovládacích prvků nic vědět
- View
  - Reprezentuje uživatelské rozhraní
  - Okno aplikace, ovládací prvek nebo stránka
- ViewModel
  - Spojuje Model a View
  - Drží stav aplikace
  - Ovládací prvky jsou pomocí bindingu propojeny s ViewModelem a čerpají z něj svůj obsah
  - Filtrování dat v závislosti na stavu aplikace
  - Rozhraní
    - INotifyPropertyChanged
    - INotifyCollectionChanged

Binding

- Je technika, která svazuje datové zdroje poskytovatele a konzumenta
- Data jsou synchronizována
- V architektuře MVVM jsou vlastnosti ViewModelu (poskytovatel) svázány s vlastnostmi vizuálních komponent (konzument)
- V poskytovateli se nachází mechanismus, který při změně dat upozorní všechny konzumenty
- Ve WPF je toho docíleno právě implementací rozhraní INotifyPropertyChanged
  -
```
class ViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    5 references
    public void OnPropertyChanged([CallerMemberName]string propertyName = "")
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

Command

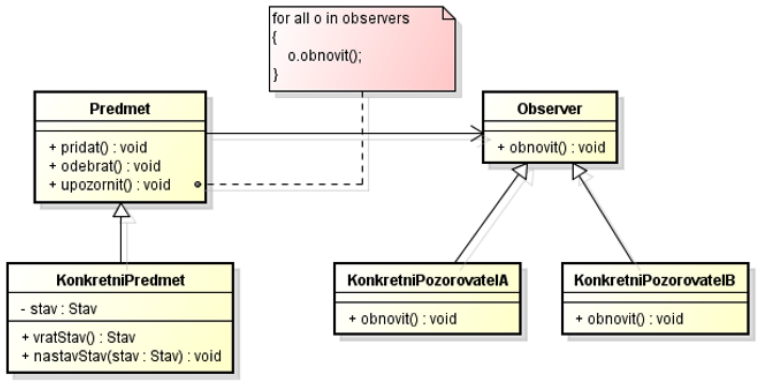
- Command je objekt reprezentující událost (také pracuje s delegáty)
- Command je taky svázán s vlastnostmi vizuálních prvků
- Metoda (Metody), respektive delegát, který v sobě uchovává a volá, pokud nastane nějaká událost
  -
```
#region Commands
1 reference
public Command SendCommand { get; private set; }
#endregion

0 references
public NewEmailViewModel()
{
    SendCommand = new Command(SendCommandExecute);
    ClearEmailRelatedProperties();
}

1 reference
public void SendCommandExecute()...
```
  -
```
<TextBox Grid.Column="1" Grid.Row="0" Margin="10" VerticalContentAlignment="Center" Text="{Binding Author}" />
<TextBox Grid.Column="1" Grid.Row="1" Margin="10" VerticalContentAlignment="Center" Text="{Binding Addressee}" />
<TextBox Grid.Column="1" Grid.Row="2" Margin="10" VerticalContentAlignment="Center" Text="{Binding Subject}" />
<Grid Grid.Column="2" Grid.Row="0" Grid.RowSpan="3" Margin="10">
    <Button Content="Send" Background="LightBlue" Height="50" Command="{Binding SendCommand}" />
</Grid>
<TextBox Grid.Column="1" Grid.ColumnSpan="2" Grid.Row="3" Margin="10"
         AcceptsReturn="True"
         TextWrapping="Wrap"
         Text="{Binding Message}" />
```

Converter

- Obekt umožňující přetypování svázané vlastnostmi na jiný typ
- Ve frameworku WPF musí taková třída implementovat rozhraní IValue Converter

Obeserver

- Observer (pozorovatel) je návrhový vzor, který umožňuje objektu spravovat své pozorovatele
- Pozorovatel čeká na změnu daného objektu a následně může reagovat



- Příkladem tohoto návrhového vzoru je právě binding
- Libovolný počet vizuálních prvků má svázáno vlastnost s ViewModelem
- Když se daná vlastnost ve ViewModelu změní, ViewModel na to upozorní a všichni pozorovatelé (nebo z předchozího odstavce konzumenti) zavolají getter dané vlastnosti a data aktualizují

Událostmi řízené programování

- Event-driven programování
- Programovací paradigma, ve kterém je chod programu řízen událostmi
- Událost může být:
  - Uživatelská akce (kliknutí na tlačítko, stisknutí klávesy)
  - Změny hodnoty senzory
  - Obdržení zprávy od ostatních programů či vláken
- Toto paradigma je dominantním v aplikacích s grafickým uživatelským rozhraním
- Byli pro to vyvinuty naslouchače (event listeners), které jednotlivým prvkům předávají delegáty rozlišené podle události
- Základní princip funguje tak, že v programu běží smyčka, která kontroluje, zda nastala událost (toto je realizováno podle nastavení určité hodnoty)
- Pokud událost zavolá tzv. Callback funkce, které přísluší dané události (delegáty v minulém odstavci)