Module 6
Data Binding a kolekce

MGR. TOMÁŠ HAVETTA - MCT

Obsah

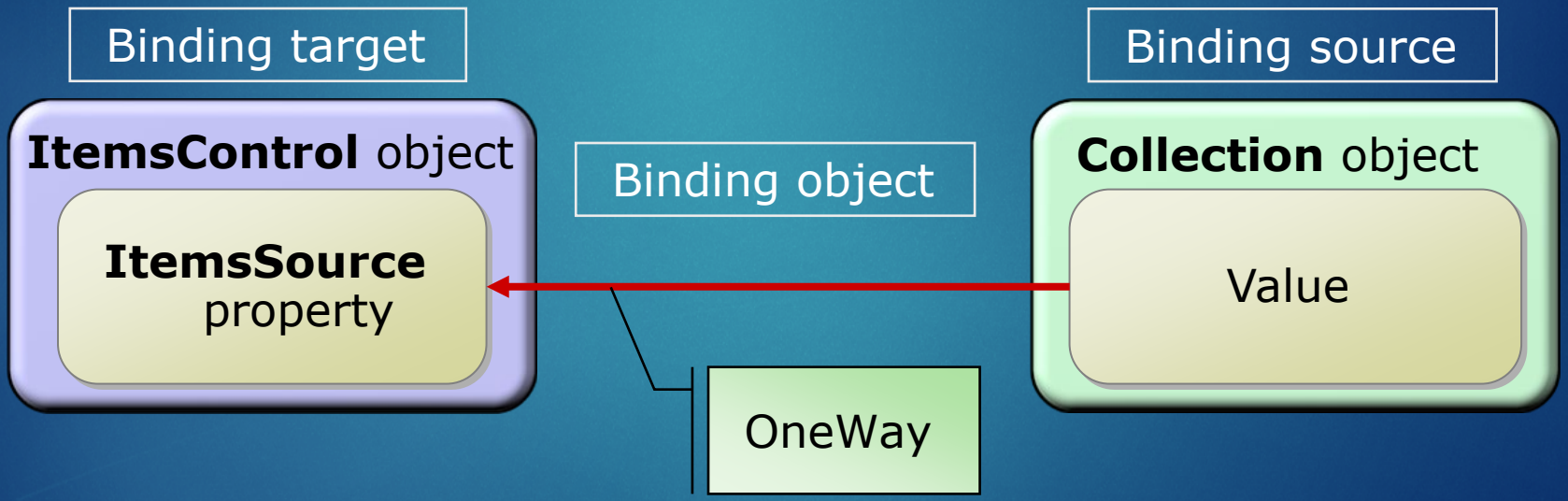
- Binding na kolekci objektů
- Collection View
- Master-Detail UI
- Data Templates

Blok 1: Binding na kolekci objektů

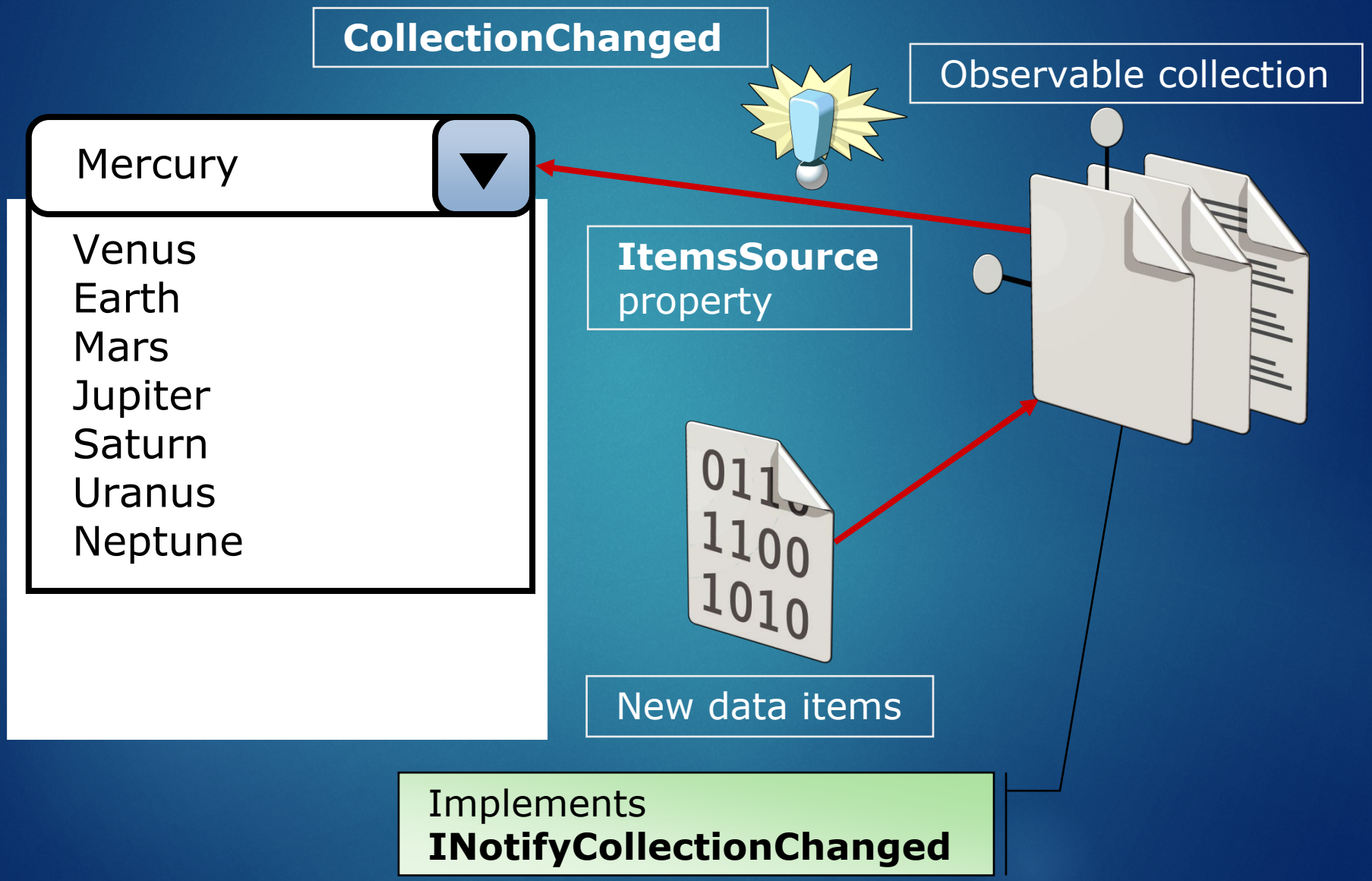
- Princip Bindingu kolekcí
- Observable kolekce
- Vlastní třídy odvozené z ObservableCollection<T>
- Použití LINQ

Princip Bindingu kolekcí

- Kolekce musí minimálně podporovat **IEnumerable**



Observable kolekce



Vlastní třídy odvozené z ObservableCollection<T>

- Rodič ObservableCollection<T>
- V konstruktoru načíst data
- Vytvořit instanci v Resources (není podmínkou, ale ulehčení)

```
public class SeznamOsob : ObservableCollection<Osoba>
{
  ...
  public SeznamOsob { ... this.Add(new Osoba(...); ... }
}
```

```
<Window ... xmlns:c="clr-namespace:WpfSample">
  <Window.Resources>
    <c:SeznamOsob x:Key="seznam"/>
  </Window.Resources>
  <ListBox ItemsSource =
              "{Binding Source={StaticResource seznam}}"/>
</Window>
```

Použití LINQ

Visual C#

Visual Basic

Others

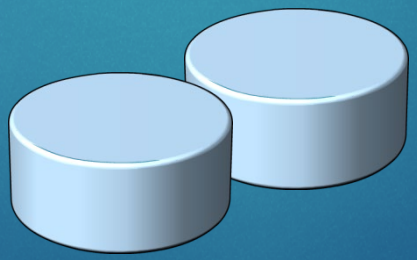
Language-Integrated Query

Standard query operators

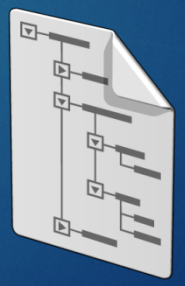
DLinq (ADO.NET)

XLinq (System.Xml)

CLR objects



Relational data

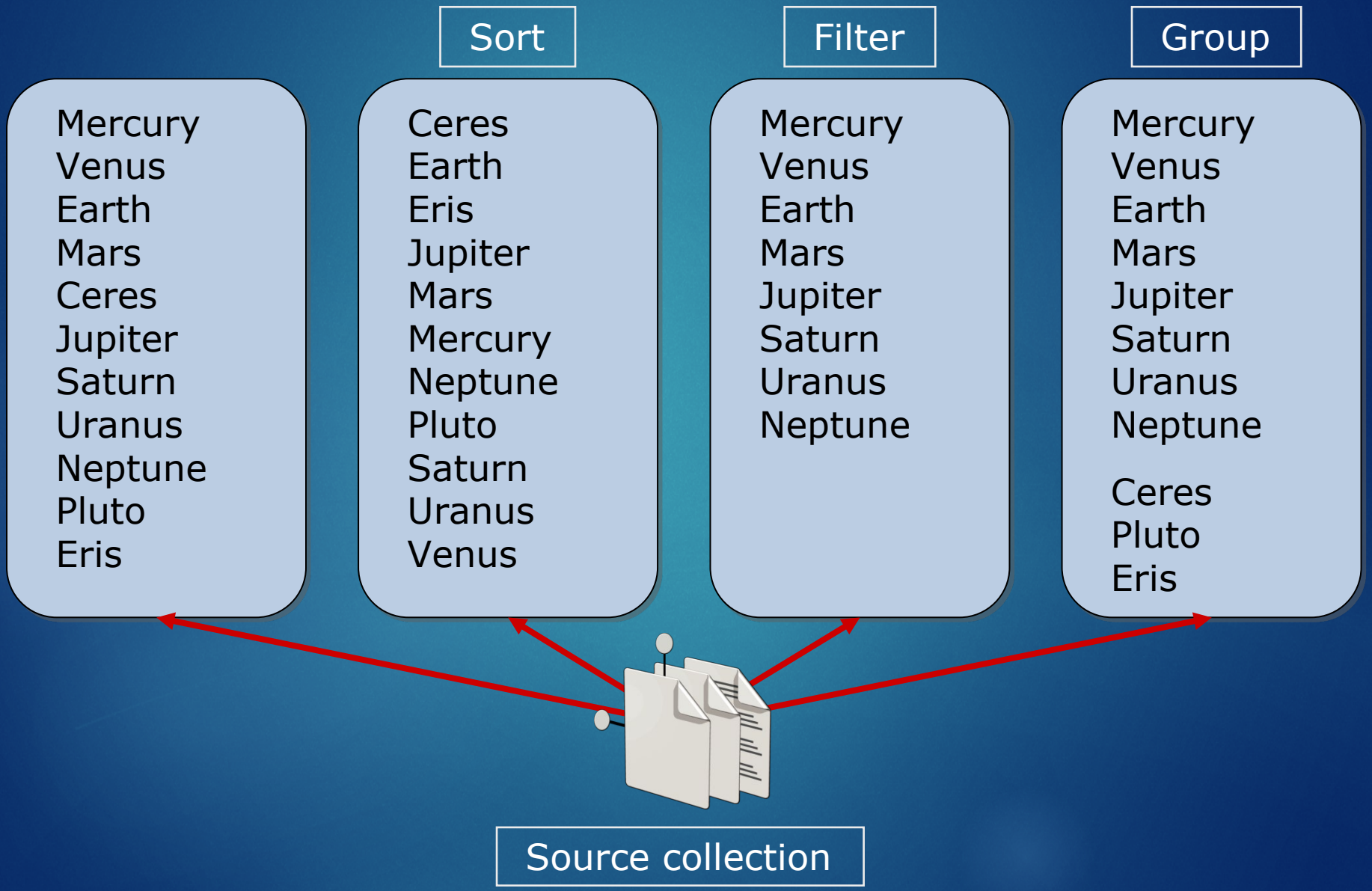


XML

Blok 2: Collection View

- Princip Collection View
- Vytvoření a použití
- Sortování dat
- Filtrování dat
- Grupování dat

Princip Collection View



Vytvoření a použití

- V Resource definovat CollectionViewSource
- Bindovat na tento element

```
<Window.Resources>
  <CollectionViewSource
    Source="{Binding Source={x:Static Application.Current},
                     Path=AuctionItems}"
    x:Key="listView" />
  ...
</Window.Resources>
```

```
<ListBox
  ItemsSource="{Binding Source={StaticResource listView}}">
  ...
</ListBox>
```

Sortování dat

- Vytvořit instanci SortDescription
- Vložit do kolekce SortDescriptions

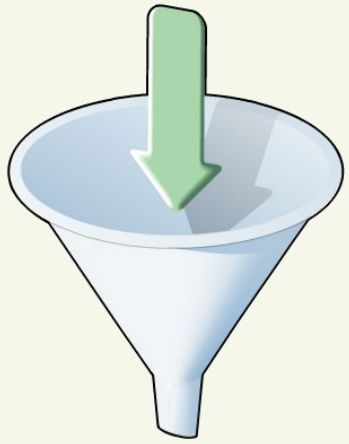
```
view.SortDescriptions.Add(
  new SortDescription("CategoryName",
                      ListSortDirection.Ascending));

view.SortDescriptions.Add(
  new SortDescription("ProductName",
                      ListSortDirection.Ascending));
```

Filtrování dat

- Reagovat na událost Filter
- V handleru nastavit e.Accepted

```
listView.Filter += new FilterEventHandler(ShowBargains);
...
private void ShowBargains(object s, FilterEventArgs e)
{
  Product p = e.Item as Product;
  if (p != null)
  {
    if (p.CurrentPrice >= 25)
      e.Accepted = false;
    else
      e.Accepted = true;
  }
}
```

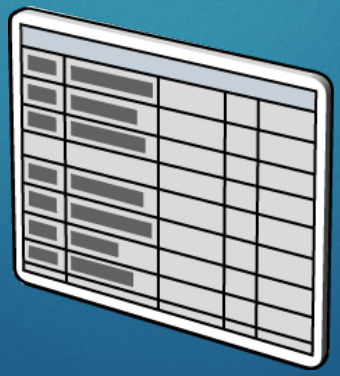


Grupování dat

- Vytvořit instanci PropertyGroupDescription
- Přidat do GroupDescriptions

```
PropertyGroupDescription desc = new
PropertyGroupDescription();
desc.PropertyName = "CategoryName";

listView.GroupDescriptions.Add(desc);
```



Blok 3: Master-Detail UI

- Princip Master-Detail UI
- Vytvoření Master-Detail UI v XAML

Princip Master-Detail UI

- Při procházení seznamu nabízí detail vybrané položky
- Umožňuje uživateli rychlý přehled obsahu
- Většinou podporuje sortování a filtraci

Vytvoření Master-Detail UI v XAML

- Definovat CollectionViewSource
- Napojit více elementů na stejný zdroj

```
<ListBox
  ItemsSource="{StaticResource listView}">
  ...
<ContentControl
  Content="{Binding Path=xxxx,
            Source={StaticResource listView}}">
  ...
```

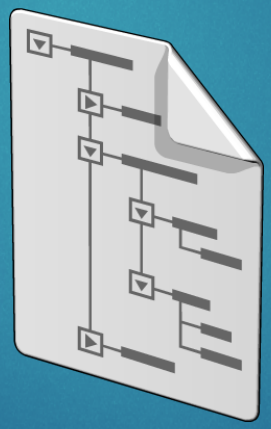
Musíte nastavit **IsSynchronizedWithCurrentItem** vlastnost na **True** pokud zdroj dat není CollectionView!

Blok 4: Data Templates

- Princip Data Templates
- Deklarace a použití
- Umístění do Resource
- Data Triggers
- Data Template Selector

Princip Data Templates

MySample.Person
MySample.Person
MySample.Person
MySample.Person



Data Template

Andy Jacobs
43
Robert Brown
34
Kelly Krout
63
Lola Jacobsen
23

Deklarace a použití

- Vytvořit **DataTemplate** element
- Přiřadit ho do **ItemTemplate** nebo **ContentTemplate**

```
<ListBox
  ItemsSource="{Binding Source={StaticResource list}}">
  <ListBox.ItemTemplate>
    <DataTemplate>
      <StackPanel>
        <TextBlock Text="{Binding Path=FirstName}" />
        <TextBlock Text="{Binding Path=LastName}"/>
      </StackPanel>
    </DataTemplate>
  </ListBox.ItemTemplate>
</ListBox>
```

Umístění do Resource

- Pro opakované použití DataTemplate je vhodné je umístit do Resources

```
<Window.Resources>
  <DataTemplate x:Key=„osobaTemplate" DataType="c:Osoba">
    <StackPanel>
      <TextBlock Text="{Binding Jmeno}" />
      <TextBlock Text="{Binding Prijmeni}"/>
    </StackPanel>
  </DataTemplate>
</Window.Resources>

<ListBox
  ItemsSource="{Binding Source={StaticResource list}}"
  ItemTemplate="{StaticResource osobaTemplate}" />
```

Data Triggers

- **Data Template** může obsahovat **Data Triggers**
- Nastavením **Binding** a **Value** vytvoříte podmínku
- Pomocí **Setter** nastavíte pro požadovanou část změnu

```
<DataTemplate x:Key="myDataTemplate">
  <DataTemplate.Triggers>
    <DataTrigger Binding="{Binding Oblast}"
                 Value="Morava">
      <Setter TargetName="border"
              Property="Foreground"
              Value="Blue"/>
    </DataTrigger>
  </DataTemplate.Triggers>
</DataTemplate>
```

Data Template Selector

- Vytvořit třídu odvozenou z **DataTemplateSelector**
- **SelectTemplate** metoda vrátí požadovaný template

```
public class MyDataTemplateSelector : DataTemplateSelector
{
    public override DataTemplate SelectTemplate(
        object item, DependencyObject container)
    {
        ...
    }
}
```

```
<ListBox
   ItemTemplateSelector="{StaticResource myTemplateSelector}">
  ...
</ListBox>
```

Lab: WPF a kolekce

- Vytvořte aplikaci pro zobrazení Osob v Master-Detail režimu
- Seznam osob vytvořte jako vlastní typ zděděný z ObservableCollection<T>
- Pro zobrazení osob v listboxu použijte DataTemplate umístěný v Window.Resources