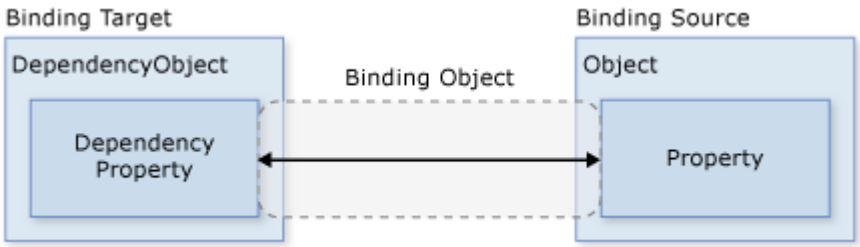
C# WPF – Povezivanje kontrola sa podacima

Katedra za informatiku
Fakultet tehničkih nauka
Univerzitet u Novom Sadu

Veza kontrole sa podacima

- Iz objektnog modela potrebno je prekopirati podatke u kontrole u kojima se podaci prikazuju
- Nakon izmene u kontrolama, potrebno je osvežiti objektni model
- Data Binding - WPF mehanizam koji automatski vrši ove operacije
- Generalno, Data Binding obezbeđuje automatsko sinhronizovanje vrednosti određenog objekta sa njegovim izvorom podataka

Data Binding



- Klasa **`System.Windows.Data.Binding`**
- Veza izvora podataka sa ciljnim podatkom
- Za prikaz imena studenta (definisanog u objektu klase Student) u TextBox komponenti
  - Objekat klase Student je izvor veze
  - Ime studenta je svojstvo (Property)
  - TextBox komponenta je cilj veze
  - Text svojstvo u TextBox komponenti je zavisno svojstvo (Dependency Property)

Smer povezivanja

- **`Binding.Mode`**
- Jednosmerno (OneWay) – izmena izvornog podatka automatski menja ciljni podatak, ali obrnuto ne važi
- Dvosmerno (TwoWay) – izmene nad izvornim podatkom menjaju ciljni podatak i obrnuto
- Jednosmerno ka izvornom podatku (OneWayToSource) – izmena ciljnog podatka automatski menja izvorni podatak, ali ne i obrnuto
- Samo jedanput (OneTime) – pri inicijalizaciji, ciljni podatak dobija vrednost izvornog podatka, ali se kasnije promene izvornog podatka ne reflektuju na vrednost ciljnog podatka

Interfejs INotifyPropertyChanged

- Obezbeđuje notifikaciju u trenutku kada se desi izmena objekta
- Objekat koji je izvor povezivanja, mora da implementira ovaj interfejs

Trenutak povezivanja

- **`Binding.UpdateSourceTrigger`**
- Svojstvo definiše u kom trenutku se podaci iz jednog objekta kopiraju u drugi
- Najčešće se kopiraju pri izmeni svojstva (**`UpdateSourceTrigger`** ima vrednost **`PropertyChanged`**)
- Može se vezati za druge događaje (npr. TextBox je napravljena tako da podrazumevano prebacuje podatke nakon što izgubi fokus)

DataBinding – Implementacija

- Prikaz imena studenta u TextBox komponenti
- TextBox koji prikazuje ime studenta – **`Path`** definiše izvorno svojstvo koje se kopira u ciljni objekat
  - XAML fajl

```
<TextBox Name="tbImeStudenta"
         Text="{Binding Path=Ime}"/>
```

- Definisanje izvora podataka za TextBox putem svojstva **`DataContext`**
  - Cs fajl

```
Student s1 = new Student();
s1.Ime = "Goran";
s1.Prezime = "Savic";

tbImeStudenta.DataContext = s1;
```

Data Binding – Implementacija

- Ako je izvor podataka drugi element u aplikaciji, može se koristiti svojstvo **`ElementName`**
- Lista sadrži objekte klase Student, a TextBox prikazuje prezime selektovanog studenta

```
<TextBox Name="tbPrezimeStudenta"
  Text="{Binding ElementName=lvStudenti,
  Path=SelectedValue.Prezime}"/>
```

- Path se ne mora specificirati ako je izvorni podatak ceo izvorni objekat

```
lvStudenti.ItemsSource = listaStudenata;
```

DataBinding – implementacija

- Moguće je kompletno povezivanje izvršiti programski u toku izvršavanja programa
- Prikaz imena studenta u TextBox komponenti

```
Binding b = new Binding();
//source property
b.Path = new PropertyPath("Ime");
//target and dependency property
tbImeStudenta.SetBinding(
  TextBox.TextProperty, b);
//binding source
tbImeStudenta.DataContext = s1;
```

Povezivanje sa kolekcijom

- **`ItemsControl`** – kontrola koja se može koristiti za prikaz kolekcije elemenata
- Naslednice klase **`ItemsControl`**: **`ListView`**, **`TreeView`**, **`DataGrid`**, ...
- Svojstvo **`ItemsSource`** za definisanje izvora podataka
- Kolekcija mora da implementira **`INotifyCollectionChanged`** interfejs

```
ObservableCollection<Student> studenti = new
  ObservableCollection<Student>();
lvStudenti.ItemsSource = studenti;
```

Manipulacija prikazom kolekcije

- Standardni zahtevi su sortiranje, filtriranje i grupisanje podataka u kolekciji
- Ove operacije realizuju se kreiranjem pogleda na kolekciju
- Pogled omogućuje različite prikaze podataka u kolekciji bez izmene sadržaja kolekcije

Kreiranje pogleda

- Korišćenjem **ICollectionView**

```
ICollectionView view =
  CollectionViewSource.
  GetDefaultView(listaStudenata);
lvStudenti.ItemsSource = view;
```

- Korišćenjem **CollectionViewSource**

```
CollectionViewSource view = new
  CollectionViewSource();
view.Source = listaStudenata;
lvStudenti.ItemsSource = view.View;
```

Sortiranje

- Pogled sadrži spisak objekata tipa **`SortDescription`**
- **`SortDescription`**
  - opis sortiranja po jednom atributu kolekcije
  - sadrži naziv svojstva i smer sortiranja
- Može se istovremeno sortirati po više podataka

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

Filtriranje

- Korišćenjem **`ICollectionView`**
  - Predikat koji definiše da li će objekat biti prikazan

```
public bool Uslov(object s)
{
    Student student = s as Student;
    return student.Ime.Length > 0 &&
           student.Ime[0] == 'A';
}
```

  - Postavljanje filtera

```
view.Filter = new Predicate<object>(Uslov);
```

Filtriranje

- Korišćenjem **`CollectionViewSource`**
  - Definišemo događaj

```
cvs.Filter += new FilterEventHandler(PocetnoA);
```

  - Definišemo obrađivač događaja

```
private void PocetnoA(object sender,
                      FilterEventArgs e)
{
   Student s = e.Item as Student;
   if (s != null)
         e.Accepted = s.Ime.Length > 0 &&
                      s.Ime[0] == 'A';
}
```

Grupisanje

- Organizacija podataka u logičke grupe po određenom kriterijumu
- Grupe studenata koji se isto prezivaju

```
PropertyGroupDescription groupDescription =
  new PropertyGroupDescription();
groupDescription.PropertyName = "Prezime";
view.GroupDescriptions.Add(groupDescription);
```

Trenutni element u pogledu

- **`ICollectionView.CurrentItem`**
- Povezivanje vrednosti u komponenti sa trenutnim elementom u pogledu

```
ICollectionView view = CollectionViewSource.
  GetDefaultView(listaStudenata);
lvStudenti.ItemsSource = view;
lvStudenti.IsSynchronizedWithCurrentItem = true;//!!!

Binding b1 = new Binding();
b1.Path = new PropertyPath("Prezime");
tbStudentRunTime.SetBinding(TextBox.TextProperty, b1);
tbStudentRunTime.DataContext = view;
```