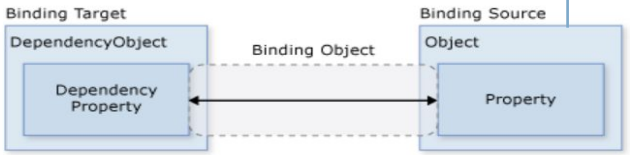
WPF - Binding

Binding

Binding **(povezivanje) je jednostavan i moćan način za automatsko ažuriranje podataka između poslovnog modela i korisničkog interfejsa. Svaki put kada se podaci vašeg poslovnog modela izmene, automatski će se ažurirati i korisnički interfejs i obrnuto.**

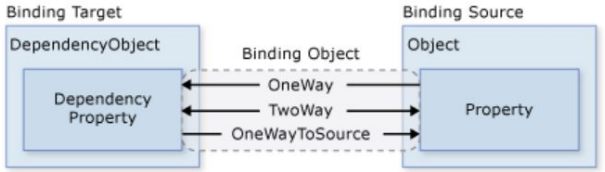
Primer *SimpleBinding*

INotifyPropertyChanged



Binding

- Model *Binding-a* možete videti slici. On se sastoji od:
  - Ciljnog objekta – element (kontrola) na koji vezujete podatak, odnosno izvor
  - Izvornog objekta – predstavlja objekat iz poslovnog modela koji vezujemo i on može biti ili .NET atribut ili zavisni atribut (*DependencyProperty*)
  - Ciljnog atributa – ovaj atribut mora da bude atribut zavisnosti (*DependencyProperty*) da bi za njega mogli da vežete neki podataka, osim onih za čitanje
  - *Path* atribut - predstavlja putanju do izvora koji se povezuje
- Da bi vezivanje podataka pravilno funkcionisalo, obe strane vezivanja moraju da obaveste o promeni koja ukazuje kada se ažurira ciljna vrednost. Ukoliko imate:
  - .NET atribute, onda se to vrši aktiviranjem događaja *PropertyChanges* iz interfejsa *INotifyPropertyChanged*
  - atribute zavisnosti (*DependencyProperty*) to se vrši pomoću *PropertyChanged* callback metapodatka atributa.



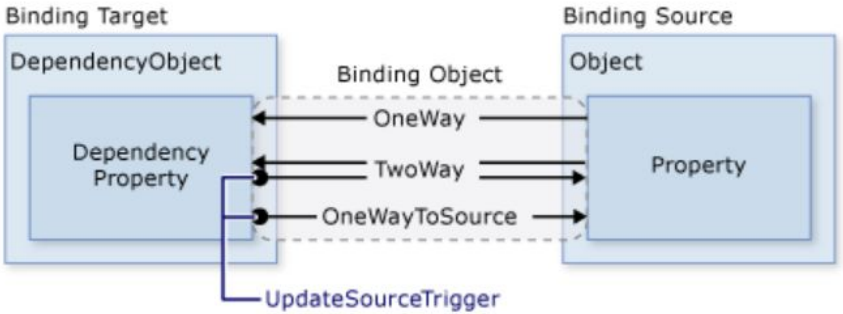
Binding – vrste vezivanja

- Postoje 3 vrste vezivanja u odnosu na smer:
  - Jednosmerno vezivanje (*OneWay*)
    - Kod ovog načina vezivanja podaci su vezani od izvora ka cilju, pa će svaka promena izvornog atributa da uzrokuje i promenu ciljnog atributa, obrnuto ne važi.
    - Pogodno ga je koristiti ako se vezana kontrola koristi samo za čitanje
  - Jednosmerno vezivanje ka izvoru (*OneWayToSource*)
    - Slično prvom načinu, samo što je sad u pitanju obrnut smer, tj. sad su podaci vezani od cilja ka izvoru, pa će svaka promena ciljnog atributa da uzrokuje promenu izvornog atributa
  - Dvosmerno vezivanje (*TwoWay*)
    - Kod ovog načina povezivanja podaci teku u oba pravca, pa će se promenom ciljnog atributa promeniti i izvorni atribut i obrnuto promenom izvornog atributa promeniće se ciljni atribut
    - Pogodno ga je koristiti za forme koje mogu da se uređuju

- *Binding* smer može da se specificira upotrebom atributa *Mode*. Podrazumevana vrednost ovog atributa zavisi od atributa zavisnosti koji se veže. Za kontrolne atribute koje mogu korisnici da uređuju, kao što su *TextBox.Text* ili *RangeBase.Value* ovaj atribut ima vrednost *TwoWay*, tj. podrazumeva dvosmerno vezivanje, a za većinu drugih atributa je to *OneWay*, tj. jednosmerno vezivanje.

Binding

- Događaj koji pokreće ažuriranje izvora
  - Pomoću atributa *Binding.UpdateSoureceTrigger* možete specificirati kada će se izmeniti vrednost izvornog atributa. Ako je vrednost ovog atributa postavljena na *PropertyChange*, onda se vrednost izvornog atributa menja čim se promeni ciljni atribut, a ukoliko ne navedemo vrednost atributa onda se izvorni atribut ažurira tek kada ciljni atribut izgubi fokus.



  - Ukoliko se atribut *UpdateSourceTrigger* eksplicitno ne navede, onda njegova podrazumevana vrednost zavisi od atributa za koji se izvor veže:
    - *PropertyChanged* – za većinu zavisnih atributa
    - *LostFocus* - za atribut *TextBox.Text*

Binding – resursi

- Resursi su definicije povezane sa nekim objektom za koji očekujete da će se koristiti više puta.
- Definisanje objekta kao resursa omogućava da mu pristupiš na drugom mestu, tj. da se neki objekat može koristiti ponovo.
- Resursi se definišu u rečnicima resursa i bilo koji objekat može da se definiše kao resurs koji ga efektivno čini deljenim resursom. Za XAML resurse se specificira jedinstveni ključ i pomoću njega možemo da vezujemo resurse. Vezivanje resursa se obavlja upotrebom *StaticResource* ili *DynamicResource* proširenja XAML jezika.
- Vrste resursa:
  - Statički resurs
    - pretražuje se u svim dostupnim rečnicima resursa i to se radi tokom učitavanja, što znači da će njegova vrednost da bude postavljena u toku učitavanja aplikacije
  - Dinamički resurs
    - obrađuje ključ resursa kreiranjem izraza koji se evaluira u toku izvršavanja aplikacije

Binding – resursi

- Statički vs dinamičiki
  - Statički resurs se dobavlja jednom u toku učitavanja XAML fajla i nije ga moguće kasnije promeniti, a dinamički resurs možemo da menjamo u toku izvršavanja aplikacije
  - Dinamički resurs vam omogućava da koristite resurse koji nisu prisutni tokom dizajniranja aplikacije

Binding – kreiranje resursa

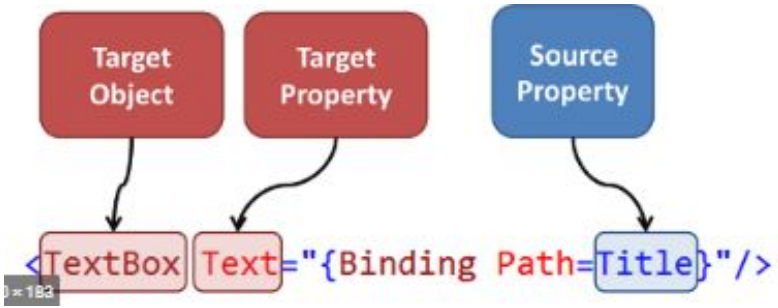
Na slici je dat primer kako kreirati resurs i povezati ga sa kontrolom.

1. kreiramo resurs koji je tipa String i tom resursu je dodeljen jedinstveni ključ, pomoću kojeg ćemo da ga referenciramo prilikom povezivanja sa kontrolom

2. prvo povezujemo kontrolu sa resursom upotrebom ključne reči *StaticResource* i tako kažemo da je naš resurs statički i da njegovu vrednost ne možemo menjati, a zatim isti taj resurs povežemo sa drugom kontrolom upotrebom *DynamicResource* i time kažemo da je on za ovu kontrolu dinamički i da može da se menja u toku izvršavanja programa

Binding – sintaksa

- Za definisanje povezivanja koristićemo proširenje XAML jezika, koje se zove *Binding*, ono se specificira unutar "{}" zagrada. *Binding* se dodeljuje atributu elementa za koji vežemo neki objekat,a pomoću atributa *Binding-a* određujemo izvorni podatak.



- Na slici vidimo da za atribut *Text* elementa *TextBox* pomoću *Path* atributa vezujemo *Title* podatak, što znači da će se podatak *Title* prikazati u *TextBox* elementu. Ovo je najjednostavniji primer vezivanja, a na narednim slajdovima razmotrićemo i ostale načine.

Binding – određivanje izvora podataka

- Vezni izvor je jedna od 4 potrebne komponente vezivanja. Stoga bez navođenja izvora, vezivanje ne bi radilo.
- Postoji nekoliko načina za određivanje izvora podataka, a to su:
  - Upotreba atributa *Path*
  - Upotreba atributa *DataContext*
  - Upotreba atributa *Source*
  - Upotreba atributa *RelativeSource*
  - Upotreba atributa *ElementName*

Binding – DataContext atribut

- *DataContext* atribut je koncept koji omogućava korisnicima da naslede informacije o izvoru vezivanja od svojih nadređenih elemenata. Koristan je za upotrebu na roditeljskom elementu kada vežete više atributa za isti izvor.
- Može direktno da se postavi na .NET objekat pri čemu se veze odnose na atribute tog objekta.
- *DataContext* atribut je atribut zavisnosti i on nasleđuje vrednosti atributa. Ako postoje podređeni elementi u kojima nije definisana vrednost ovog atributa, tada će sistem za njegovu vrednost postaviti vrednost *DataContext-a* najbližeg nadređenog elementa za koji je postavljena vrednost.
- U XAML-u najčešće se postavlja kao deklaracije vezivanja
- Treba voditi računa pri korišćenju ovog atributa da se ne naprave kružne reference za povezivanje.

Binding – Path atribut

- Atribut *Path* se koristi kada je izvor vezivanja objekat, da bi se odredio atribut sa kojim povezujete element korisničkog interfejsa.
- U najjednostavnijem slučaju vrednost atributa *Path* je ime atributa izvornog objekta koji će se koristiti za povezivanje.
  - *Path = PropertyName*
  - Takođe, možete specificirati i podatribute nekog atributa. Sintaksa je slična C# sintaksi.
    - Primer: *Path = ShoppingCart.Order*
- Ako je izvor vezivanja kolekcija onda trenutna stavka može da se specificira upotrebom kose crte (/).
  - Izraz: *Path = /* ukazuje da ce nam trenutna stavka kolekcije biti povezana sa ciljnim elementom
  - Takođe, kosu crtu možemo da kombinujemo i sa nazivom objekta
    - Primer: *Path = /Offices/ManagerName*
    - Izraz koji je naveden u primeru nam određuje trenutnu stavku kolekcije koja u sebi sadrži atribut Offices koji je kolekcija, a njegova trenutna stavka je objekat koji sadrži atribut ManagerName i taj atribut će nam biti povezan sa ciljnim elementom
- Ukoliko izostavimo atribut *Path* ili njegovu vrednost postavimo na tačku (.), onda to podrazumeva da se veže ceo izvorni objekat
  - *Text = "{Binding}"* -> *Text = "{Binding Path=.}"* . Ova dva izraza su ekvivalentna zato što i jedan i drugi vezuju element sa trenutnim izvorom.
  - Ovaj scenario vezivanja je koristan kada povezujete kolekciju, ceo objekat, a ne samo jedan njegov atribut ili kada je izvor tipa String

Binding – Source atribut

- Upotreba *Source* atributa je jedan od načina na koji možete eksplicitno da postavite izvor vezivanja i tako nadjačate atribut *DataContext*.
- Ako vam nije potrebna funkcionalnost uspostavljanja opsega u kojem nekoliko atributa nasleđuju isti kontekst podataka, možete direktno nad elementom iskoristiti *Source* atribut umesto *DataContext-a*

Binding – Path i Source atribut

- Primeri:

```
<ListBox ItemsSource="{Binding}"
         IsSynchronizedWithCurrentItem="true"/>
```

- U primeru iznad, vidite da nije naveden *Path* atribut što znači da vezujete atribut *ItemSource* za ceo objekat, koji je naveden u *DataContext-u*, jer *ListBox* nasleđuje *DataContex* iz roditeljskog elementa. Ovaj način vezivanja je poznat kao prazna sintaksa vezivanja.

```
<TextBlock Text="{Binding Source={StaticResource myDataSource}, Path=PersonName}"/>
```

- U primeru na slici iznad, vidite da Source atribut ukazuje na statički resurs sa ključem *myDataSource*, a *Path* atribut ukazuje na atribut *PersonName* koji pripada objektu na koji ukazuje resurs. A to znači da će vrednost *Text* atributa da bude vrednost *PersonName* objekta *myDataSource*. Upotrebom *Source* atributa direktno u elementu *TextBlock* specificirate da se resurs veže samo za taj element.

Binding – DataContext, Source i Path

Definisanje resursa koji je tipa *MyData*, koji je definisan u *SDKSample* prostoru imena

```
<DockPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
           xmlns:c="clr-namespace:SDKSample">
    <DockPanel.Resources>
        <c:MyData x:Key="myDataSource"/>
    </DockPanel.Resources>
    <DockPanel.DataContext>
        <Binding Source="{StaticResource myDataSource}"/>
    </DockPanel.DataContext>
    <Button Background="{Binding Path=ColorName}"
            Width="150" Height="30">
        I am bound to be RED!
    </Button>
</DockPanel>
```

Definisanje *DataContext* atributa upotrebom sintakse elementa za definisanje atributa. Svi elementi koji se nalaze u *DockPanel-u* će naslediti ovaj *DataContext*

Povezivanje atributa *Background* sa atributom *ColorName* upotrebom *Path* atributa, a *ColorName* je atribut objekta *MyData*

Određivanje izvora pomoć Source atributa. Izvor će u ovom slučaju da bude statički resurs koji smo prethodno napravili i dodelili mu ključ *myDataSource*

Binding – atribut RelativeSource

- Atribut *RelativeSource* se koristi za vezivanje jednog atributa objekta na drugi atribut istog objekta ili za definisanje vezivanja u stilu ili šablonu. Izvor se navodi u odnosu na to gde je vaš vezni cilj.
- Pomoću njega možete eksplicitno da postavite izvor vezivanja na određeni element i tako nadjačate nasleđeni *DataContext*

Binding – atribut ElementName

- Atribut *ElementName* se koristi kada želite da povežete dve kontrole, tj. kada želite da se povežete sa atributom drugog elementa u vašoj aplikaciji
- Pomoću njega možete eksplicitno da postavite izvor vezivanja na određeni element i tako nadjačate nasleđeni *DataContext*
- Kada postavite ovaj atribut, njegova vrednost se mora odnositi na element na jednoj od sledećih lokacija:
  - Trenutni XAML prostor imena
  - U prostoru imena predloženih roditelja ako je cilj vezivanje u šablonu podataka ili kontrolnom šablonu. Zbog ovog ograničenja se ne možete povezati na elemente koji nisu kreirani pomoću XAML-a. Za povezivanje na programski kreirane elemente koristite atribut *Source*.
  - Povezivanja koja koriste ovaj atribut gotovo uvek uključuju i jednostavan atribut *Path* koji imenuje svojstvo koje postoji u kontroli (elementu) na koju se povezuje

Binding – atribut ElementName

- Primer: Možete da ga koristiti ako želite da prikažete vrednost koju ste izabrali pomoću *Slider* kontrole ili ako želite da neki sadržaj kontrole povežete sa atributom *SelectedValue* u vašoj listi.

```
<Slider x:Name="slider1" Minimum="1" Maximum="100"/>
<TextBox Text="{Binding ElementName=slider1, Path=Value, Mode=TwoWay}"/>
```

Pomoću atributa *ElementName* definišemo sa kojom kontrolom se povezujemo, ovde je to konkretno kontrola slider

Specificiramo atribut slider-a sa kojim se povezujemo

Pomoću *Mode* atributa eksplicitno navodimo smer povezivanja, konkretno za ovaj primer je to *TwoWay*, što znači da će se promene jedne kontrole odraziti i na drugoj kontroli.

Binding – Source, RelativeSource i ElementName atributi

Ovi atributi su međusobno isključivi u povezivanju. Ako postavite jedan od ovih atributa, postavljanje bilo koja dva preostala u isto povezivanje izazvaće izuzetak!

Binding (XAML) – upotreba sintakse objektnog elementa

- Sintaksa objektnog elementa je alternativa za kreiranje povezivanja. Koristi se u slučajevima kada nije podržano povezivanje pomoću XAML proširenja.
- Obično se koristi kada imamo atribut čija vrednost nije tipa String i za koju ne postoji konverzija tipa, nego moramo sami da je implementiramo.

```
<TextBlock Name="myconvertedtext"
  Foreground="{Binding Path=TheDate,
                       Converter={StaticResource MyConverterReference}}">
  <TextBlock.Text>
    <Binding Path="TheDate"
             Converter="{StaticResource MyConverterReference}"/>
  </TextBlock.Text>
</TextBlock>
```

Kreiranje povezivanja upotrebom XAML proširenja

Kreiranje proširenja upotrebom sintakse objektnog elementa

Binding – C# kod

- Drugi način da uradimo povezivanje elemenata korisničkog interfejsa i izvornog podatka
- Potrebno je:

i. kreirati *Binding* objekat i u njemu postavite vrednosti atributa:

   - *Mode* - specificira smer povezivanja
   - *Source* – specificira izvor
   - *Converter* – Ovaj atribut služi za definisanje konvertora koji će se iskoristiti za pretvaranje vrednosti ukoliko su vrednosti izvornog i ciljnog objekta različitog tipa.
   - *ConverterCulture* – Ovaj atribut dobavlja ili postavlja kulturu u kojoj će se evaluirati pretvarač. Ukoliko ne navedemo ovaj atribut, onda će se iskoristiti atribut Language ciljnog objekta. U XAML-u je to obično "*en-US*" ili nasleđena vrednost iz matičnog elementa stranice ako je eksplicitno navedena.

ii. Pomoću klase *BindingOperations* i njenog atributa *Binding* povežete izvor, cilj i novokreirani *Binding* objekat

Binding – C# kod

```
// Make a new source, to grab a new timestamp
MyData myChangedData = new MyData();

// Create a new binding
// TheDate is a property of type DateTime on MyData class
Binding myNewBindDef = new Binding("TheDate");

myNewBindDef.Mode = BindingMode.OneWay;
myNewBindDef.Source = myChangedData;
myNewBindDef.Converter = TheConverter;
myNewBindDef.ConverterCulture = new CultureInfo("en-US");

// myDateText is a TextBlock object that is the binding target object
BindingOperations.SetBinding(myDateText, TextBlock.TextProperty, myNewBindDef);
```

Postavili smo atribut *Mode* na vrednost *OneWay*, što znači da će ovo biti jednosmerno vezivanje od izvora ka cilju, odnosno da će svaka promena izvornog podatka dovesti do promene prikaza ciljnog izvora

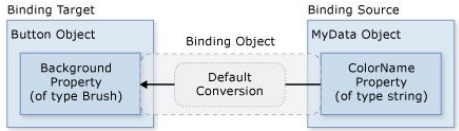
Za izvorni objekat smo postavili objekat *myChangedData*, koji je tipa *MyData*

U ovom slučaju ćemo postaviti i atribut *Converter*, zato što podatak koji prikazujemo nije tipa String, i atribut *ConverterCulture*, zato što se radi o podatku koji se različito prikazuje u zavisnosti od kulture

I na kraju nam je ostalo još samo da povežemo izvor i cilj i da specificiramo na koji način se povezuju i to ćemo uraditi pomoću atributa *Binding* klase *BindingOperations*

Binding – konverzija podataka

- Kada pravite povezivanje i ukoliko želite da povežete dva atributa koji nisu istog tipa, potrebno je da napravite konvertor, tj. da se uradi konverzija između tipova. Mnogi atributi već imaju ugrađene konvertore podataka, ali za one koji nemaju možete da napravite svoje konvertore.
- Kada kreirate svoje konvertore, potrebno je da implementirate *IValueConverter* interfejs i da klasu koja implementira ovaj interfejs anotirate sa anotacijom *ValueConversion* koja kao parametre prima tipove koji se konvertuju. Na sledećem slajdu je primer implementiranog konvertora.
- Konvertor će se pozvati svaki put kada dođe do povezivanja ciljnog elementa i izvornog podatka. Potrebno je samo da u *Binding* objekat dodamo i atribut *Converter*. Ukoliko eksplicitno ne navedemo koji konvertor da se koristi, tada će se kreirati podrazumevani konvertor koji će pokušati da izvrši pretvaranje i ako ne uspe, kao povratnu vrednost će vratiti *null*.



Primer

Binding – konverzija podataka

- Na slici je prikazano kako kreirati konvertor. Konvertor na slici pretvara vrednost tipa *Color* u vrednost tipa *SolidColorBrush*, a to smo specifirali upotrebom anotacija *ValueConversion*. Takođe, da biste napravili konvertor, potrebno je da implementirate *IValueConverter* interfejs.

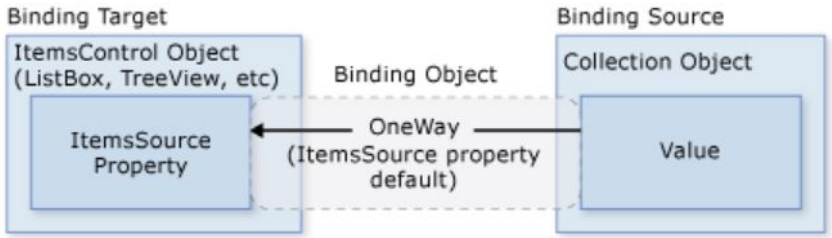
```
[ValueConversion(typeof(Color), typeof(SolidColorBrush))]
0 references
public class ColorBrushConverter : IValueConverter
{
    0 references
    public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        Color color = (Color)value;
        return new SolidColorBrush(color);
    }

    0 references
    public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        return null;
    }
}
```

Binding – konverzija podataka

- Upotreba:
  - Podaci koje prikazujete se različito prikazuju u zavisnosti od kulture
  - Podaci koji se prenose nisu namenjeni za prikaz, nego će se postaviti kao vrednost atributa neke kontrole
  - Isti podatak vezan za više kontrola ili više atributa kontola
  - Podatak koji prikazujete je kolekcija, tj. *MultiBinding*
  - Imate brojčanu vrednost, ali želite da na različite načine prikažete pozitivne i negativne brojeve
  - Želite da proverite *CheckBox* na osnovu vrednosti, ali vrednost je niz poput “da” ili “ne” umesto *Boolean* vrednosti

Binding - kolekcije

- Za prikaz kolekcije objekata obično koristimo neku od ItemsControl kontrola kao što su ListBox, ListView ili TreeView.



- Kao što je prikazano na dijagramu iznad, za povezivanje ItemsControl-a za objekat kolekcije, koristi se atribut ItemsSource. ItemsSource možete da smatrate sadržajem ItemsControl-a. ItemsSource podrzumevano podržava jednosmerno povezivanje.