

Interakcija čovek - računar

Vežbe 5

Fakultet tehničkih nauka
Univerzitet u Novom Sadu



Validacija podataka

Uvod

Gotovo sve aplikacije koje prihvataju ulazne podatke od strane korisnika u vidu forme danas koriste neki vid validacije ulaznih podataka. Validacija se radi u cilju proveravanja da li su ulazni podaci očekivanog formata i tipa. U sklopu validacije ulazi i davanje povratne informacije korisniku, koja je često u obliku tekstualne poruke čija je svrha da otkrije korisniku šta nije u redu sa input-om.

Validacija u WPF-u

Kako bi se razumela implementacija validacije podataka u WPF-u, potrebno je poznavati koncepte binding-a i data type konverzije u WPF-u koji su objašnjeni na prethodnim vežbama.

Bitno je napomenuti da kada se setuje vrednost Value polja neke kontrole (npr. *TextBox*) u WPF-u, inicijalni tip te vrednosti je string. *Binding*-om se vrednost Value polja ovih kontrola vezuje za neki *property* u *code-behind* koji ima svoj tip. Ukoliko je to primitivan tip podatka ili enumeracija, izvršiće se automatska konverzija. Ukoliko *XAML* parser ne prepozna *property* kao primitivan tip ili enumeraciju, potrebno je definisati *custom* konvertor tako što se implementira interfejs *IValueConverter*.

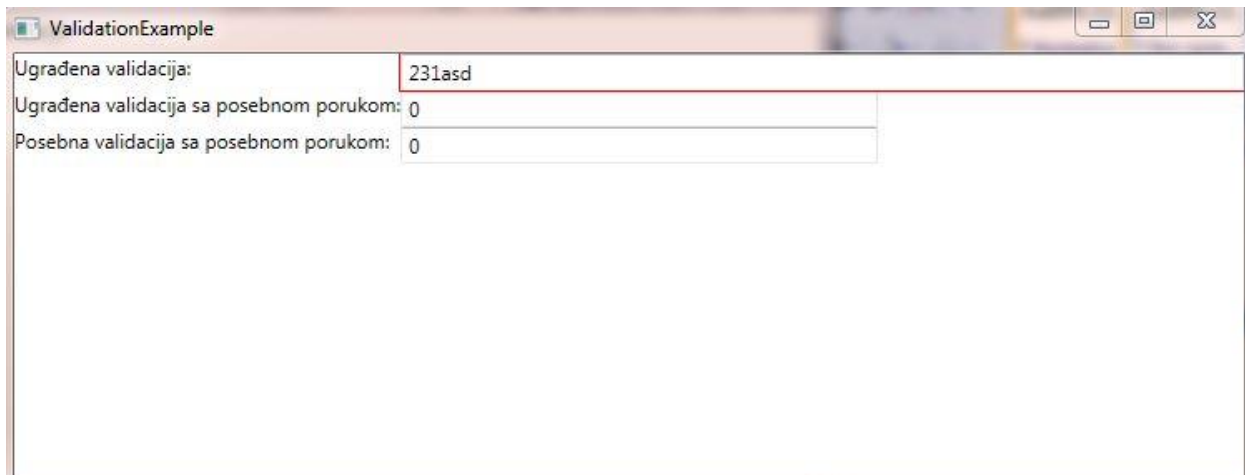
Ukoliko korisnik unese nevalidnu vrednost koja ne može biti konvertovana, nastaje greška i korisniku se vraća povratna informacija koja indicira grešku u validaciji. Prikaz greške zavisi od *control template*-a. Ukoliko se koristi *TextBox* kontrola, po default-u se kontrola uokviruje crvenom bojom čime se indicira korisniku da postoji greška u validaciji (*default control template*).

U primeru za vežbe 5, u prozoru *ValidationExample* (*ValidationExample.xaml* fajl) imamo *Grid Layout* sa tri reda i dve kolone, koja predstavljaju input polja (*TextBox*) sa labelama ispred. Prvi *TextBox* predstavlja primer za ugrađenu validaciju (slika 1). Ova kontrola je bind-ovana za *property Test1* iz *code-behind* (*ValidationExample.xaml.cs*) koji je tipa *double*. Ukoliko se unese string koji ne može da se konvertuje u *double*, polje se uokviruje crvenom bojom. Konverzija se vrši pre poziva *Set* metode *property*-ja, stoga se vrednost *property*-ja ne menja ukoliko je došlo do validacione greške. Moguće je odrediti vreme kad će se izvršiti validacione provere (pre konverzije, pre ili posle poziva *Set* metode *property*-ja) o čemu će biti reči u nastavku.

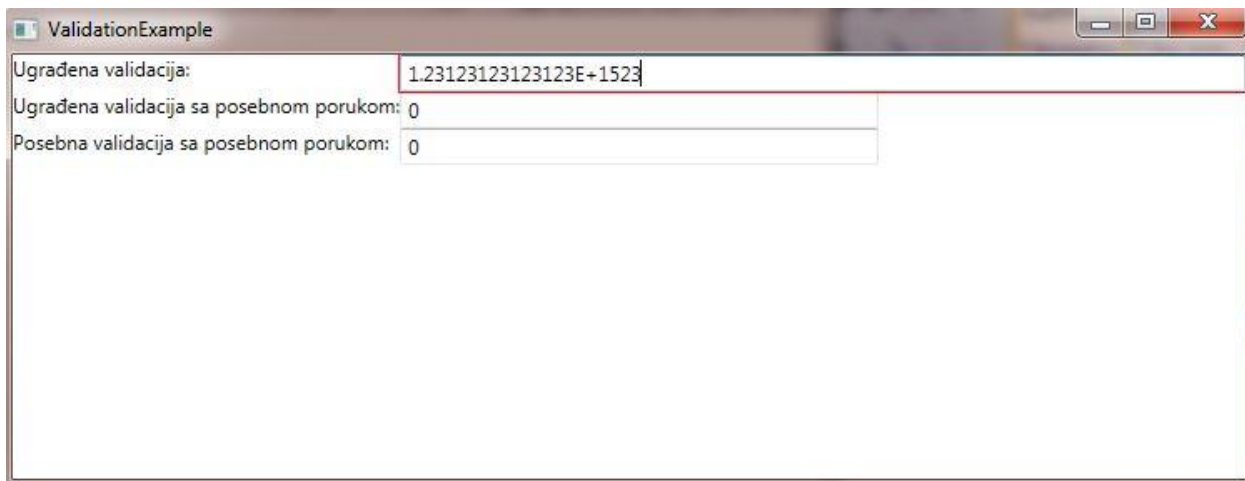
```
17 <TextBlock Grid.Column="0" Grid.Row="0">Ugrađena validacija: </TextBlock>
18 <TextBlock Grid.Column="0" Grid.Row="1">Ugrađena validacija sa posebnom porukom: </TextBlock>
19 <TextBlock Grid.Column="0" Grid.Row="2">Posebna validacija sa posebnom porukom: </TextBlock>
20
21 <TextBox Grid.Column="1" Grid.Row="0" Text="{Binding Path=Test1,UpdateSourceTrigger=PropertyChanged}"></TextBox>
22 <TextBox Grid.Column="1" Grid.Row="1" Margin="0,0,220,0" Text="{Binding Path=Test2,UpdateSourceTrigger=PropertyChanged}">
23 <Validation.ErrorTemplate>
```

Slika 1. *ValidationExample.xaml* - Primer za ugrađenu validaciju(*default control template*)

Povratna informacija o grešci se može videti na slici 2 i slici 3.

A screenshot of a Windows application window titled "ValidationExample". It contains three text input fields. The first field, labeled "Ugrađena validacija:", contains the text "231asd". The second field, labeled "Ugrađena validacija sa posebnom porukom:", contains the number "0". The third field, labeled "Posebna validacija sa posebnom porukom:", also contains the number "0". The first field has a red border, indicating a validation error.

Slika 2. Unos karaktera koji nisu cifre

A screenshot of the same "ValidationExample" window. The first text input field now contains the text "1.23123123123123E+1523". The other two fields remain the same, with "0" in each. The first field has a red border, indicating a validation error.

Slika 3. Izvan opsega double tipa

Control Template

Ovakva poruka je često nedovoljno precizna. Preciznost se obično postiže kratkim opisom validacione greške. Da bi se videla poruka, potrebno je umesto default *control template*-a (koji uokviruje element crvenom bojom) kreirati *custom control template*. To se radi podešavajući *Validation.ErrorTemplate property*. U primeru za vežbe 5 za *control template* postavljamo *grid layout*. Da bi se razumela sintaksa sa slike 4 sledi kratko objašnjenje.

Poruka koja opisuje grešku se nalazi u polju *ErrorContent* objekta *System.Windows.Controls.ValidationError*. Za svaku kontrolu koja koristi koncept *binding*-a se kači *Validation.Errors* (*attached property*) kolekcija koja sadrži listu objekata tipa *System.Windows.Controls.ValidationError*. U primeru se sa [0] pristupa prvom elementu *Validation.Errors* kolekciji kojem se u *runtime*-u dodaju *ValidationError* objekti ukoliko se dese validacione greške. *ErrorTemplate* se prikazuje u *adorned layer*-u koji se renderuje povrh ostalih elemenata. *Adorner* ima značenje "onaj koji ukrašava", stoga je u ovom primeru *TextBox* kontrola element koji se ukrašava. Iz ovog razloga se koristi element *AdornedElementPlaceholder* sa kojim se podešava pozicija *TextBox* kontrole.

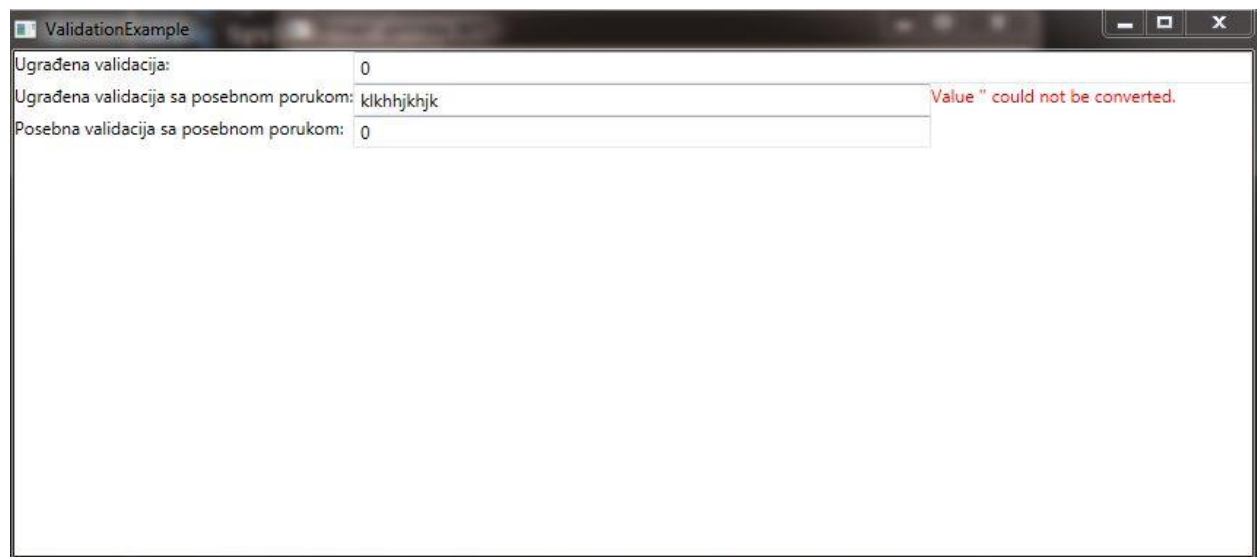
```

22 <TextBox Grid.Column="1" Grid.Row="1" Margin="0,0,220,0" Text="{Binding Path=Test2,UpdateSourceTrigger=PropertyChanged}">
23     <Validation.ErrorTemplate>
24         <ControlTemplate>
25             <Grid>
26                 <Grid.RowDefinitions>
27                     <RowDefinition />
28                 </Grid.RowDefinitions>
29                 <Grid.ColumnDefinitions>
30                     <ColumnDefinition />
31                     <ColumnDefinition Width="Auto" />
32                 </Grid.ColumnDefinitions>
33                 <AdornedElementPlaceholder Grid.Column="0" Grid.Row="0"/>
34                 <TextBlock Grid.Column="1" Grid.Row="0" Text="{Binding [0].ErrorContent}" Foreground="Red"/>
35             </Grid>
36         </ControlTemplate>
37     </Validation.ErrorTemplate>
38 </TextBox>

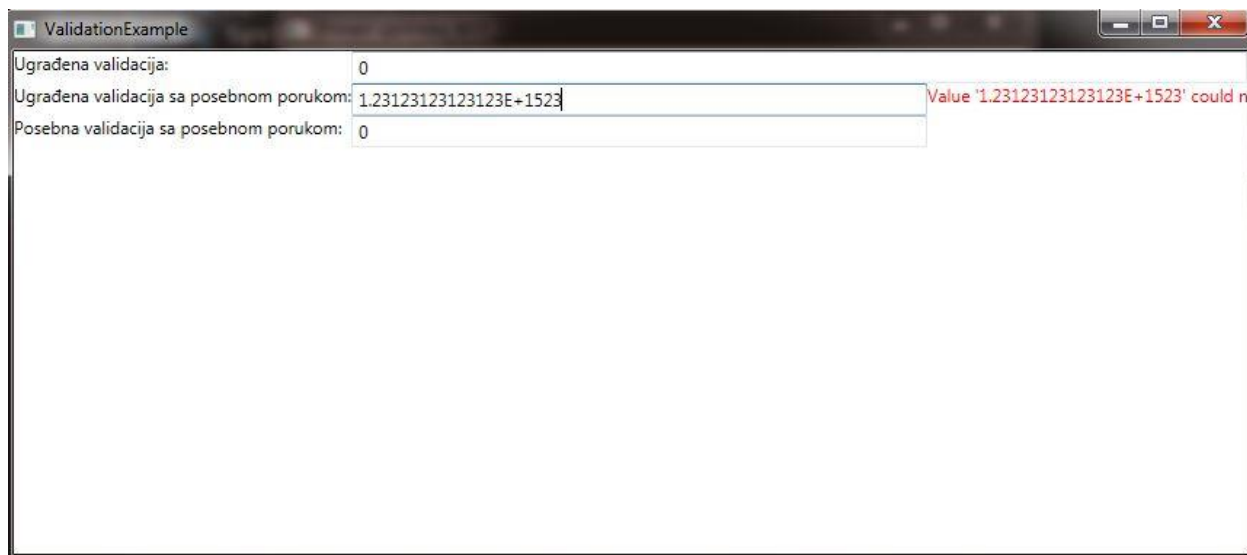
```

Slika 4. *ValidationExample.xaml*

Slika 5 i slika 6 prikazuju poruku o grešci. Na slici 6 se vidi da se elementi koji se renderuju u *Adorner Layer*-u ne uzimaju u obzir prilikom kreiranja *layout*-a.



Slika 5



Slika 6

Validaciona pravila

Poslednja *TextBox* kontrola u primeru za vežbe 5 uvodi koncept validacionih pravila koja su enkapsulirana abstraktnom klasom *ValidationRule*. Implementacijom validacionih pravila se otvara mogućnost prilagođavanja validacione poruke. Implementacija podrazumeva nasleđivanje abstraktne klase *ValidationRule* i implementacija metode *Validate* u kojoj se vrši provera validnosti input vrednosti. Za *ValidationRule* objekat je vezano polje *ValidationStep* kojim se podešava kada će se pozvati metoda *Validate*. Moguće vrednosti polja *ValidationStep* su:

- *RawProposedValue* - validaciono pravilo se pokreće pre konverzije. *Default*-na vrednost.
- *ConvertedProposedValue* - validaciono pravilo se pokreće posle konverzije, ali pre poziva *Set* metode *property*-ja.
- *UpdatedValue* - validaciono pravilo se pokreće nakon poziva *Set* metode *property*-ja.
- *CommittedValue* - validaciono pravilo se pokreće nakon što je vrednost source *property*-ja komitovana

U primeru za vežbe 5 u fajlu *StringToDoubleValidationRule.cs* su dati par validacionih pravila: *StringToDoubleValidationRule* (slika 5) i *MinMaxValidationRule* (slika 6).

```

9      public class StringToDoubleValidationRule : ValidationRule
10     {
11         1 reference
12         public override ValidationResult Validate(object value, System.Globalization.CultureInfo
13             cultureInfo)
14         {
15             try
16             {
17                 var s = value as string;
18                 double r;
19                 if(double.TryParse(s, out r))
20                 {
21                     return new ValidationResult(true, null);
22                 }
23                 return new ValidationResult(false, "Please enter a valid double value.");
24             }
25             catch
26             {
27                 return new ValidationResult(false, "Unknown error occured.");
28             }
29         }
30     }

```

Slika 7. Vršiti proveru mogućnosti konverzije stringa u double tip

```

30     public class MinMaxValidationRule : ValidationRule
31     {
32         1 reference
33         public double Min
34         {
35             get;
36             set;
37         }
38         1 reference
39         public double Max
40         {
41             get;
42             set;
43         }
44         1 reference
45         public override ValidationResult Validate(object value, System.Globalization.CultureInfo
46             cultureInfo)
47         {
48             if (value is double)
49             {
50                 double d = (double)value;
51                 if (d < Min) return new ValidationResult(false, "Value too small.");
52                 if (d > Max) return new ValidationResult(false, "Value too large.");
53                 return new ValidationResult(true, null);
54             }
55             else
56             {
57                 return new ValidationResult(false, "Unknown error occured.");
58             }
59         }
60     }

```

Slika 8. Vršiti proveru da li se vrednost nalazi u opsegu zadatom sa Min i Max

Metoda *Validate* prihvata dva parametra: *value* tipa *object* i *cultureInfo* tipa *System.Globalization.CultureInfo*. Parametar *value* predstavlja vrednost preuzetu iz WPF kontrole (*binding target*, u primeru *TextBox*), dok *cultureInfo* predstavlja "kulturu" (sadrži podatke o pismu, kalendaru, formatima datuma i slično) koja se koristi. Povratna vrednost metode je objekat tipa klase *ValidationResult*. Konstruktor ove klase prihvata dva parametra, gde prvi setuje polje *IsValid* koja daje informaciju o validnosti vrednosti, dok drugi setuje polje *ErrorContent* kojim se definiše poruka greške.

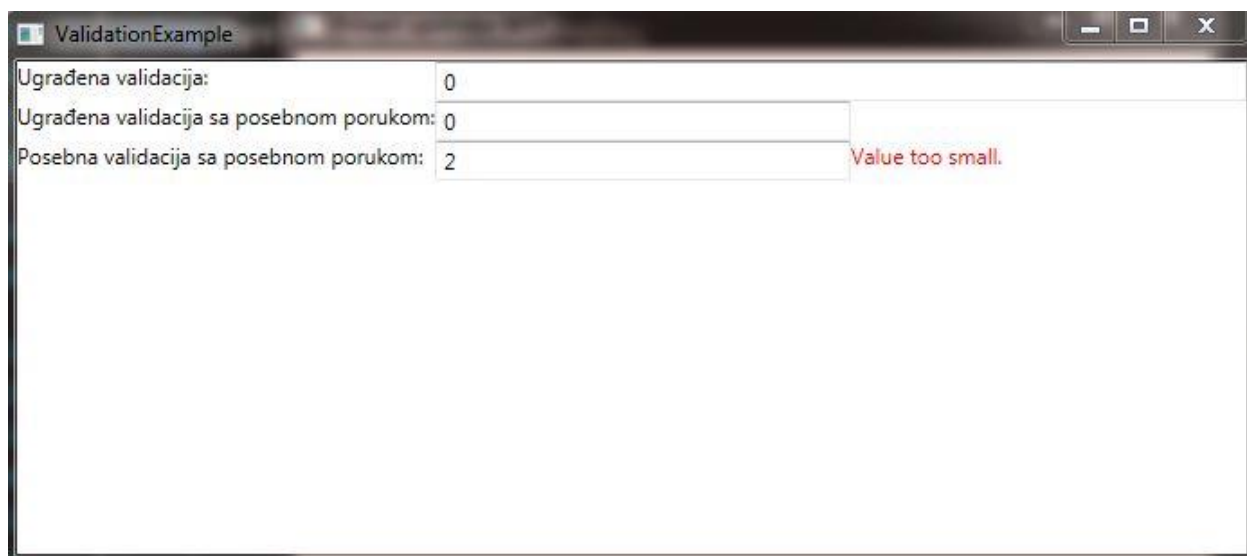
Na slici 9 je prikazana sintaksa za vezivanje(*bind*-ovanje) validacionih pravila za WPF kontrolu.

```
40 <TextBox Grid.Column="1" Grid.Row="2" Margin="0,0,220,0">
41   <TextBox.Text>
42     <Binding Path="Test3" UpdateSourceTrigger="PropertyChanged">
43       <Binding.ValidationRules>
44         <val:StringToDoubleValidationRule ValidationStep="RawProposedValue"/>
45         <val:MinMaxValidationRule ValidationStep="ConvertedProposedValue" Min="10" Max="350"/>
46       </Binding.ValidationRules>
47     </Binding>
48   </TextBox.Text>
49   <Validation.ErrorTemplate>
50     <ControlTemplate>
51       <Grid>
52         <Grid.RowDefinitions>
53           <RowDefinition />
54         </Grid.RowDefinitions>
55         <Grid.ColumnDefinitions>
56           <ColumnDefinition />
57           <ColumnDefinition Width="Auto" />
58         </Grid.ColumnDefinitions>
59
60         <AdornedElementPlaceholder Grid.Column="0" Grid.Row="0" x:Name="textBox"/>
61         <TextBlock Grid.Column="1" Grid.Row="0" Text="{Binding [0].ErrorContent}" Foreground="Red"/>
62       </Grid>
63     </ControlTemplate>
64   </Validation.ErrorTemplate>
65 </TextBox>
```

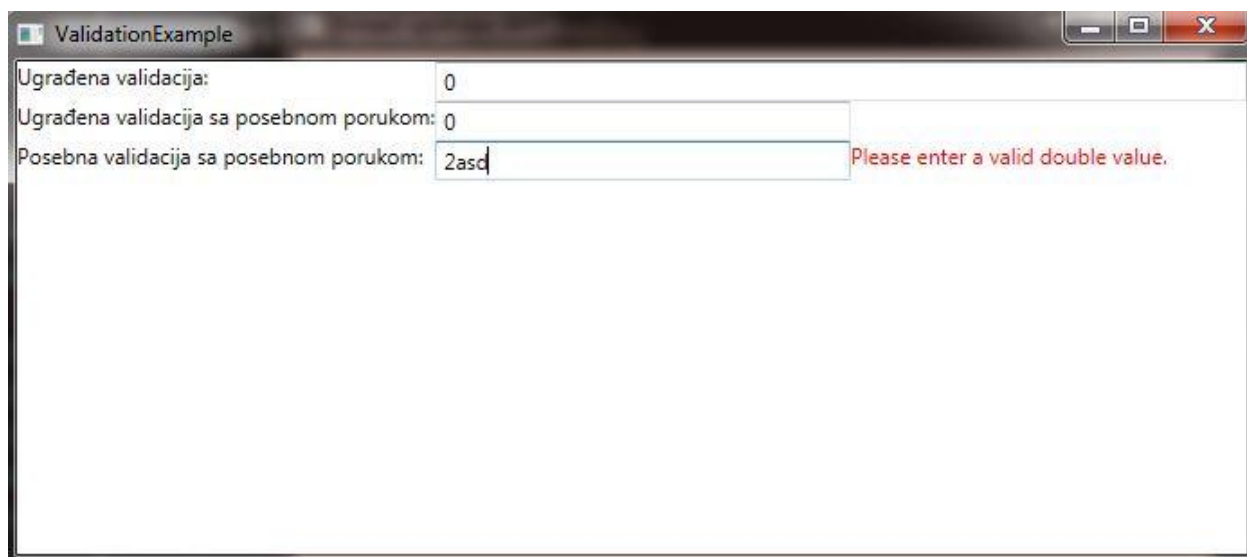
Slika 9. *ValidationExample.xaml*

Sa slike 9 se vidi da se *set*-ovanjem *ValidationStep* polja pravilo *StringToDoubleValidationRule* poziva pre konverzijem, dok se *MinMaxValidationRule* poziva nakon konverzije.

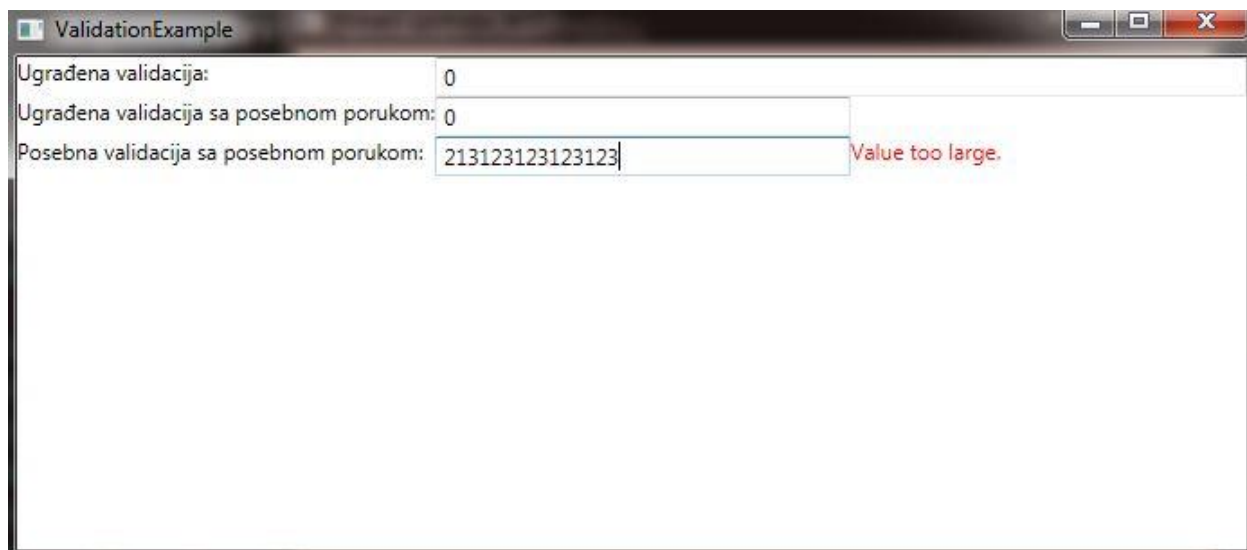
Na ovaj način je moguće napraviti *custom* poruke o grešci. Na narednim slikama su prikazane te poruke u primeru za vežbe 5.



Slika 10. Greška koju daje MinMaxValidationRule za vrednost koja je manja od Min



Slika 11. Greška koju daje StringToDoubleValidationRule



Slika 12. Greška koju daje MinMaxValidationRule za vrednost koja je veća od Max

KORISNI LINKOVI:

<https://docs.microsoft.com/en-us/dotnet/desktop-wpf/data/data-binding-overview>

<https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/how-to-implement-binding-validation>

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.validation?view=netframework-4.8>

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.validationrule?view=netframework-4.8>

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.validationresult?view=netframework-4.8>