Agent Assignment Lab3

Kommentarer til løsningsforslag

I alle debug-mapper ligger der en fil med det "rette" navn, så man kan vælge File-Open for at få indlæst "standard" agenterne.

Lab1

1. Tilføjer en datatemplate til <Window.Resources> i MainWindow.xaml

```
<DataTemplate x:Key="agentTemplate">

        <WrapPanel>
            <TextBlock Text="{Binding Path=ID}" Padding="5, 0"
                       Width="50"
                       />
            <TextBlock Text …
```

2. På listboxen fjernes DisplayMemberPath og i stedet sættes ItemTemplate:

```
ItemTemplate="{StaticResource agentTemplate}"
```

3. Som en service for brugeren har jeg tilføjet en kolonne i Grid-panelet og lagt en `GridSplitter` i denne kolonne:

```
<GridSplitter
      Grid.Column="1"
      HorizontalAlignment="Center"
      VerticalAlignment="Stretch"
      Width="5"
      />
```

Lab2

1. Tilføjer en ny klasse AgentIdToColorValueConverter til projektet som implementerer IValueConverter interfacet – se denne implementering i kodefilen, og bemærk min brug af Debug-klassen.

2. Opretter AgentIdToColorValueConverter som en resource i xaml-filen:

```
<local:AgentIdToColorValueConverter x:Key="IdToColorConverter" />
```

3. Binder Foreground property på alle controller i ListBoxens datatemplate til ID og bruger AgentIdToColorValueConverter:

```
Foreground="{Binding Path=ID, Converter={StaticResource
IdToColorConverter}
```

Lab3

1. Jeg laver en klasse til at indeholde mulige specialities:
   ```
   public class Specialities : ObservableCollection<string>
   ```
   Og i dens default konstruktor initierer jeg den med nogle specialities.

2. Jeg opretter den som en ressource i vinduet, så jeg kan binde til den.

3. Opretter en combobox og binder dens ItemSource til specialities klassen (de mulige valg i comboboxen), og binder dens SelectedItem til Speciality på Agenten:
   ```
   <ComboBox Margin="100,72,14,0"
             VerticalAlignment="Top"
             SelectedItem="{Binding Path=Speciality}"
             ItemsSource="{Binding Source={StaticResource specialities}}"
             />
   ```

Lab4

1. Tilføjer label og combobox til valg af sorteringsnøgle i toolbaren. Bemærk bruger af Tag-property for ikke at lave logikken afhængig af tekst på UI (vil fejle hvis sproget ændres).
2. I eventhandleren for comboboxens SelectionChanged event sættes SortDescription for bindingens CollectionViewSource til den valgte værdi:
   ```
   ComboBoxItem cbi = e.AddedItems[0] as ComboBoxItem;
   string newSortOrder;
      if (cbi != null)
      {
          if (cbi.Tag == null)
              newSortOrder = "None";
          else
              newSortOrder = cbi.Tag.ToString();
          SortDescription sortDesc = new SortDescription(newSortOrder,
                                         ListSortDirection.Ascending);
          ICollectionView cv = CollectionViewSource.GetDefaultView(
                                                          DataContext);

          if (cv != null)
          {
              cv.SortDescriptions.Clear();
              if (newSortOrder != "None")
                  cv.SortDescriptions.Add(sortDesc);
          }
   }
   ```

3. Trin 1 og 2 er nok for at få sorteringen til at virke, og Next og Previus kommandoerne virker stadig, da de er uafhængige af agenternes sortering. Men det har den bivirkning at man ved delete sletter den forkerte agent, hvis sorteringsrækkefølgen er anderledes en ved ”None”! Og ved Add vælges ikke den nye Agent.
   For at rette op på disse fejl tilføjes en property `CurrentAgent` på Agents klassen, listbox'ens property `CurrentItem` bindes til denne property og koden for `DeleteAgent` og `AddAgent`

ændres til at benytte `CurrentAgent`.

Lab5

1. Tilføjer label og combobox til valg af filter i toolbaren. ItemsSource bindes til en property i Agents klassen som returnerer en ReadOnly collection af specialities baseret på specialities klassen. Men da brugeren her også skal kunne vælge alle (det samme som ingen filter) tilføjes "All" som første element.

```
public IReadOnlyCollection<string> FilterSpecialities
 {
      get
      {
          ObservableCollection<string> result = new
                                      ObservableCollection<string>();
          result.Add("All");
          foreach (var s in new Specialities())
              result.Add(s);
          return result;
      }
  }
```

2. SelectedIndex bindes til en ny CurrentSpecialityIndex i Agents.
3. Selve filtreringen sker i CurrentSpecialityIndex's Setter – eller rettere i denne aktiveres CollectionViews filtreringsmekanisme. Dog ikke hvis selected index er 0 (= all). I dette tilfælde fjernes filterfunktionen:

```
public int CurrentSpecialityIndex
{
   get { return currentSpecialityIndex; }
   set
   {
      if (currentSpecialityIndex != value)
      {
         ICollectionView cv = CollectionViewSource.GetDefaultView(this);
         currentSpecialityIndex = value;
         if (currentSpecialityIndex == 0)
            cv.Filter = null; // Index 0 is no filter (show all)
         else
         {
            filter = FilterSpecialities.ElementAt
                                              (currentSpecialityIndex);
            cv.Filter = CollectionViewSource_Filter;
         }
         NotifyPropertyChanged();
      }
    }
  }
```

4. Selve filter-funktionen (predicate) som insættes til CollectionView'et:

```
private bool CollectionViewSource_Filter(object ag)
{
   Agent agent = ag as Agent;
   return (agent.Speciality == filter);
}
```

5. Da nye “tomme” agenter vil blive sorteret fra af et filter, så fjerner jeg filtreringen ved at sætte `CurrentSpecialityIndex = 0 i AddAgent`.