GUI Lab: Agent Assignment 1

Kommentarer til løsningsforslag

Lab1

Det sværeste i denne opgave er nok at vælge, hvordan man vil angive databinding-source. Jeg har valgt at sætte vinduets Datacontext i koden for konstruktoren til en instans af agentklassen som oprettes i koden.

For alle relevante kontroller skal jeg så binde deres Text property til relevant property på Agentklassen. F.eks.:

```
Text="{Binding Path=ID}"
```

Lab2

I denne opgave har jeg valgt at sætte Datacontext på Grid'et i stedet for på vinduet (det er mest for at vise, at det kan man også). Det kræver blot at man har givet grid'et et Name i XAML-koden:

```
agentGrid.DataContext = agents;
```

Det store valg i denne opgave er hvordan man vil binde de enkelte kontroller til listen af agenter, og hvordan man vil synkronisere selectedItem i listboksen med det der vises i de individuelle kontroller. Da jeg har sat DataContext til listen af agenter, så kan det gøres meget smart ved at binde listboksens ItemSource (uden at angive Path, da den skal bindes til selve objektet i listen) samt sættes dens property IsSynchronizedWithCurrentItem til True.

```
<ListBox ItemsSource="{Binding}"
         IsSynchronizedWithCurrentItem="True"
         DisplayMemberPath="CodeName"
```

Alternativ kunne jeg have gjort som MacDonald og sat listboksens ItemSource i koden, og så i Xaml-koden bundet Grid'et til ListBoxens SelectedItem via elementBinding.

Lab3

I denne opgave er der 3 udfordringer:

1. Man skal sætte DataContext i XAML-koden,
2. Man skal kunne navigerer frem og tilbage i listen ved tryk på en knap
3. Man skal kunne oprette nye agenter.

1: I hjælp til opgaven står der at man skal bruge en ViewModel til løsning af opgaven, så jeg tilføjer klassen MainWindowViewModel, og i den opretter jeg en ObservableCollection af Agents, og en property, som kan returnere denne liste.
I XAML-koden oprettes en instans af MainWindowViewModel i Winduets DataContext:

```
<Window.DataContext>
    <local:MainWindowViewModel />
</Window.DataContext>
```

For at vist listen af agenter i listboxen, så skal dens ItemSource bindes til Agents-propertyen:

```
ItemsSource="{Binding Path=Agents}"
```

For at få textboxene til at virke igen, så skal MainWindowViewModel tilføjes en property CurrentAgent - og det er vigtigt at den notifier.

```
public Agent CurrentAgent
{
    get { return currentAgent; }
    set
    {
        if (currentAgent != value)
        {
            currentAgent = value;
            NotifyPropertyChanged();
        }
    }
}
```

Og så skal listboksens SelectedItem bindes til den, og det samme skal tekstboxene.

2: I knappernes eventhandlere implementerer jeg navigeringen ved at ændre SelectedIndex for listboksen.

```
void btnForward_Click(object sender, RoutedEventArgs e)
{
    if (lbxAgents.SelectedIndex < lbxAgents.Items.Count - 1)
        lbxAgents .SelectedIndex = ++lbxAgents.SelectedIndex;
}
```

3: Det store problem med at indsætte en ny agent er, at listen er oprettet i Xaml-koden, og at man ikke kan give den et navn, således at den kan tilgås som et almindeligt datamedlem i koden. Men vi ved at Winduets DataContext indeholder vores MainWindowViewModel objekt så vi kan derigennem få en reference som vi kan typecaste til MainWindowViewModel og på den kan vi kalde en metode, som kan tilføje vores nye agent.
Af hensyn til brugervenlighed flytter jeg selected index til den nye agent og flytter focus til det første indtastningsfelt.

```
private void BtnAddNew_Click(object sender, RoutedEventArgs e)
{
    var vm = DataContext as MainWindowViewModel;
    vm.AddNewAgent();
    lbxAgents.SelectedIndex = lbxAgents.Items.Count - 1;
    tbxId.Focus();
}
```