Process of transferring data from the Update View to the Main View

1. **The ViewModelLocator must register and create instances of the UpdateViewModel and the MainViewModel. The registration is for finding the viewmodel during data context binding.**

public ViewModelLocator()

{

ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

//Register the View Models

SimpleIoc.Default.Register<MainViewModel>();

SimpleIoc.Default.Register<AddViewModel>();

SimpleIoc.Default.Register<UpdateViewModel>();

}

1. **The Data context must be set in a high-level container (in this case the window itself) to find out where we want to bind our data.**
   1. **For UpdateViewModel:**

DataContext="{Binding UpdateViewModel, Source={StaticResource Locator}}"

* 1. **For MainViewModel:**

DataContext="{Binding MainViewModel, Source={StaticResource Locator}}"

1. **The Update button must be bound to the UpdateViewModel UpdateCommand relay command.**

<i:Interaction.Triggers>

<i:EventTrigger EventName="Click">

<i:InvokeCommandAction Command="{Binding UpdateCommand}"/>

</i:EventTrigger>

</i:Interaction.Triggers>

1. **When the user clicks the update button, the UpdateCommand relay command is called in the UpdateViewModel.**
   1. **The relay command calls the UpdateCommandAction method to execute the command.**

UpdateCommand = new RelayCommand(UpdateCommandAction);

1. **If the data is valid, the UpdateCommandAction will send a message …**

private void UpdateCommandAction()

{

if (ValidateData())

{

var messageViewModel = new MessageViewModel()//use message class

{

FirstText = FirstTextBox,

LastText = LastTextBox,

EmailText = EmailTextBox

};

Messenger.Default.Send(messageViewModel);

}

}

1. **The MainViewModel must set up a receiver for UpdateViewModel messages**

Messenger.Default.Register<MessageViewModel>(this, OnReceiveNewData);

1. **The MainViewModel receives the message and calls OnReceiveNewData to execute.**

//Receiving data

void OnReceiveNewData(MessageViewModel obj)

{

if(obj != null)

{

firstContext = obj.FirstText;

lastContext = obj.LastText;

emailContext = obj.EmailText;

}

}

1. **OnReceiveNewData sets the dataContext variables to the message object’s variables and raises a PropertyChanged flag.**

// Property get/set

public string FirstContext

{

get { return firstContext; }

set

{

firstContext = value;

RaisePropertyChanged("FirstContext");

}

}

1. **The dataContext variables are used to update a new object into the member list. First Name for Example:**

var i = memberList.IndexOf(selectedMember);

memberList[i] = new Member(firstContext, lastContext, emailContext);

1. **The newest member in the list will now show in the Main View.**