WPF - MVVM : Gestion de pages virtuelles

Auteur : Hassan KANDIL
Date de publication :01/10/2009
Revisitée : 01/02/2017

Une listView ou un DataGid peuvent contenir un nombre important de données en Pages virtuelles et en se basant sur le DataBinding. Le principe est d"une simplicité effarante. Seuls les habitués à une programmation classique y compris fonctionnelle pourront saisir l'intérêt de MVVM et WPF dans cet exemple.

Sans s'attarder sur le lancement d'un projet de type MVVM, nous allons supposer la présence d'une fenêtre principale et de trois dossiers Model-ModelView-Views.
L'objectif étant de remplir une DataGrid avec des données et d'utiliser un groupe de RadioButton pour passer à une autre page c'est à dire remplir la DataGrid par d'autres données sans perdre les données de la page précédente.
Il est évident que la sauvegarde dans une programmation classique assurera cette transition, mais avec le DataBinding, la tâche semble être prise en charge directement par le FrameWork .net.

Ainsi, pouvons nous commencer par créer un contrôle utilisateur qui contient la DataGrid et qui se base sur un model définissant ses colonnes, et les attributs par la même occasion. Le ModelView de ce contrôle fera le lien entre le model et la vue.

Notre Contrôle s'appellera DataGridPages. La DataGrid dans l'exemple aura 3 colonnes: Opération-Resumé-Taille. Cela ne vaut rien dire de particulier, mais suffira pour la démonstration que nous cherchons à faire.

Il est plus intéressant d'utiliser une source de données de type SQL que de mettre des données figées dans notre liste .. Dans un autre article nous donnons un exemple d'exploitation SQL via ADO.net et une lecture d’un fichier texte.

Dans la démonstration on se contentera des données constantes.

Deux boutons seront nécessaires pour accomplir la tâche, le premier est pour la lecture, le second permet d'effacer le contenu de la DataGrid.

1. Le fichier modèle.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;

namespace Demonstration.Model
{
    class DataGridPagesModel : INotifyPropertyChanged
    {
        #region Constructers
//l’utilisation des paramètres nommés est très utile pour l(ordre des paramètres et
//les valeurs par défaut
        public DataGridPagesModel(int operation = 0, string resume = "", long taille =
0)
        {
            this.operation = operation;
            this.resume = resume;
            this.taille = taille;
```

```
        }
        #endregion
        #region Attributs - Fields
        private int operation;
        private string resume;
        private long taille;
        #endregion
        #region Properties
        public int Operation { get => operation; set { operation = value;
TrtEvent("Operation"); } }
        public string Resume { get => resume; set { resume = value;
TrtEvent("Resume"); } }
        public long Taille { get => taille; set { taille = value; TrtEvent("Taille");
} }
        #endregion
        #region Events
        public event PropertyChangedEventHandler PropertyChanged;

        public void TrtEvent(String property)
        {
            //PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(property));
            if (PropertyChanged != null) PropertyChanged(this, new
PropertyChangedEventArgs(property));
        }
        #endregion
    }

}
```

2. DataGridPagesCiewModel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections.ObjectModel;
using Demonstration.Model;
using Demonstration.Views;

namespace Demonstration.ViewModel
{
        class DataGridPagesViewModel
        {
            public Commandes LectureCommand { get; }
            public Commandes ClearCommand { get; }
            public static ObservableCollection<DataGridPagesModel> PagesSet;
            public static List<ObservableCollection<DataGridPagesModel>> PagesSets =
new List<ObservableCollection<DataGridPagesModel>>();
            public DataGridPagesViewModel()
            {
                LectureCommand = new Commandes(OnLecture);
                ClearCommand = new Commandes(OnClear);
                LoadPagesSet();
            }
            public static void PositionPartSet(int indice)
            {
                PagesSet = PagesSets[indice];
                DataGridPages ActuelView = DataGridPages.thisSave;
                ActuelView.dataGrid.ItemsSource = PagesSet;
            }
```

```
            public static void LoadPagesSet()
            {
                for (int i = 1; i <= 3; i++)
// 3 = nombre de pages qui pourrait être en donnée constante surtout qu'il va être
utilisé dans la partie RadioButton
                {
                    PagesSets.Add(new ObservableCollection<DataGridPagesModel>());
                }
                PagesSet = PagesSets[1];
                DataGridPages ActuelView = DataGridPages.thisSave;
                ActuelView.dataGrid.ItemsSource = PagesSet;
            }
            private DataGridPagesModel _selectedDataPart;
            public DataGridPagesModel SelectedDataPart
            {
                get
                {
                    return _selectedDataPart;
                }
                set
                {
                    _selectedDataPart = value;
                }
            }
            private void OnDelete()
            {
                PagesSet.Remove(SelectedDataPart);
            }
            private void OnLecture()
            {
                LoadData.LoadConstantes();
            }
            private void OnClear()
            {
                PagesSet.Clear();
            }
        }
}
```

3. Le XAML du DataGrid

```
<UserControl x:Class="Demonstration.Views.DataGridPages"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Demonstration.Views"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid Margin="0,0,-80,0">
        <StackPanel Margin="10,0,20,0">
            <DataGrid Height="300" Width="500" CanUserAddRows="True"
CanUserDeleteRows="True" AutoGenerateColumns="False"
                      x:Name="dataGrid" ItemsSource="{Binding PartSet}"
                      SelectedItem="{Binding SelectedDataPart}"
AlternatingRowBackground="LightBlue"
                      AlternationCount="2" HorizontalAlignment="Left">
                <DataGrid.Columns>
                    <DataGridTextColumn Header="Opération" Binding="{Binding
Operation, Mode=TwoWay}" />
```

```
                    <DataGridTextColumn Header="Resumé" Binding="{Binding Resume,
Mode=TwoWay}" />
                    <DataGridTextColumn Header="Taille" Binding="{Binding Taille,
Mode=TwoWay}" />
                </DataGrid.Columns>
            </DataGrid>
            <StackPanel Orientation="Horizontal">
                <DockPanel>
                    <Button Name="btnLecture" Content="Lecture" Command="{Binding
LectureCommand}" HorizontalAlignment="Left" VerticalAlignment="Center" Width="75" />
                    <Button Name="btnClear" Content="Clear" Command="{Binding
ClearCommand}" HorizontalAlignment="Left" VerticalAlignment="Center" Width="75" />
                </DockPanel>
            </StackPanel>
        </StackPanel>
    </Grid>
</UserControl>
```

4. Le code behind

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections.ObjectModel;
using Demonstration.ViewModel;

namespace Demonstration.Views
{
    /// <summary>
    /// Logique d'interaction pour DataGridPages.xaml
    /// </summary>
    public partial class DataGridPages : UserControl
    {
        public static DataGridPages thisSave;
        public DataGridPages()
        {
            InitializeComponent();
            thisSave = this;
            this.DataContext = new Demonstration.ViewModel.DataGridPagesViewModel();
        }
    }
}
```

La gestion de pages

1. Un autre contrôle utilisateur est crée pour la gestion de pages avec des RadioButton au nombre des pages à gérer.

```
<UserControl x:Class="Demonstration.Views.LesPages"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Demonstration.Views"
             xmlns:data="clr-namespace:Demonstration.Model"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <UserControl.Resources>
        <DataTemplate DataType="{x:Type data:LesPagesModel}">
            <StackPanel Orientation="Vertical">
                <RadioButton GroupName="AllPages" Name="rbScreen" IsChecked="{Binding
Path=ScreenChecked, Mode=TwoWay}" Content="{Binding Path=ScreenContent,Mode=TwoWay}"
HorizontalAlignment="Left" Height="20.034" Margin="3 5 3 5" VerticalAlignment="Top"
Width="74"/>
            </StackPanel>
        </DataTemplate>
    </UserControl.Resources>
    <Grid>
        <Frame  HorizontalAlignment="Right" Margin="10,0,0,0" Content=""
BorderThickness="3" BorderBrush="Black" Height="301" VerticalAlignment="Top"
Width="140.5" >
        </Frame>
        <StackPanel Orientation="Vertical">
            <ListBox Name="lbPages"  Margin="12,0,0,2" Height="300"
VerticalAlignment="Top" Width="140"/>
        </StackPanel>
    </Grid>
</UserControl>
```

2. **Code behind**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Demonstration.ViewModel;

namespace Demonstration.Views
{
    /// <summary>
    /// Logique d'interaction pour LesPages.xaml
    /// </summary>
    public partial class LesPages : UserControl
    {
        public LesPages()
        {
            InitializeComponent();
            lbPages.ItemsSource = LesPagesVM.OurPage;
```

```
        }
    }
}
```

3. ViewModel : LesPagesVM

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Demonstration.Model;
using System.Collections.ObjectModel;

namespace Demonstration.ViewModel
{
    class LesPagesVM
    {
        public LesPagesVM()
        {
            LoadPages();
        }
        public static ObservableCollection<LesPagesModel> OurPage = new
ObservableCollection<LesPagesModel>();
        public void LoadPages()
        {
            for (int i = 1; i <= 3; i++) // nombre de pages
            {
                OurPage.Add(new LesPagesModel { ScreenContent = "Page N° " +
i.ToString(), ScreenChecked = (i == 1) ? true : false, ScreenNumber = i });
            }
        }

    }
}
```

4. Le fichier Model:LesPagesModel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;
using Demonstration.ViewModel;

namespace Demonstration.Model
{
    public class LesPagesModel : INotifyPropertyChanged
    {
            private string screenContent;
            private bool screenChecked;
            private int screenNumber;
            public string ScreenContent { get => screenContent; set { screenContent =
value; RaisePropertyChanged("ScreenContent"); } }
            static int indice;
            public bool ScreenChecked
            {
```

```
                get => screenChecked;
                set
                {
                    screenChecked = value;
                    if (value == true) indice = screenNumber;
                    if (indice > 0) ChangePage(indice);
                    RaisePropertyChanged("ScreenChecked");
                }
            }
            public int ScreenNumber
            {
                get => screenNumber;
                set
                {
                    screenNumber = value;
                    RaisePropertyChanged("ScreenNumber");
                }
            }
            public void ChangePage(int indice)
            {
                DataGridPagesViewModel.PositionPartSet(indice);
            }
            #region Notif
            public event PropertyChangedEventHandler PropertyChanged;
            public void RaisePropertyChanged(string property)
            {
                PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(property));
            }
            #endregion
    }
}
```

Des Classes Extra

Deux classes sont ajoutées pour compléter le projet, dont une dérive de l'interface ICommande et qui servira à définir les commandes du premier contrôle utilisateur. La deuxième est faite pour isoler les données du programme de présentation.

En principe, et quand on manipule des données SQL ou autres, cette classe sera dans une bibliothèque de classes qui gérera l'accès aux données.

1. Commandes : voir LectureCommand et ClearCommand

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace Demonstration
{
    public class Commandes : ICommand
    {
        Action _TargetExecuteMethod;
        Func<bool> _TargetCanExecuteMethod;
        public Commandes(Action executeMethod)
        {
            _TargetExecuteMethod = executeMethod;
        }
```

```
        public Commandes(Action executeMethod, Func<bool> canExecuteMethod)
        {
            _TargetExecuteMethod = executeMethod;
            _TargetCanExecuteMethod = canExecuteMethod;
        }
        public event EventHandler CanExecuteChanged = delegate { };
        public void RaiseCanExecuteChanged()
        {
            CanExecuteChanged(this, EventArgs.Empty);
        }
        bool ICommand.CanExecute(object parameter)
        {
            if (_TargetCanExecuteMethod != null)
            {
                return _TargetCanExecuteMethod();
            }
            if (_TargetExecuteMethod != null)
            {
                return true;
            }
            return false;
        }
        void ICommand.Execute(object parameter)
        {
            if (_TargetExecuteMethod != null)
            {
                _TargetExecuteMethod();
            }
        }
    }
}
```

2. **LoadData : Une classe qui doit servir dans l'architecture 3 tiers**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Demonstration.Model;
using Demonstration.ViewModel;

namespace Demonstration
{
    class LoadData
    {
        public static void LoadConstantes()
        {
            DataGridPagesViewModel.PagesSet.Add( new DataGridPagesModel(operation:
DataGridPagesViewModel.PagesSet.Count()+1,  resume: "Constante", taille: 10));
        }
    }
}
```

La fenêtre principale

Elle contient tout simplement les deux contrôles.
Dans un projet où un menu oriente la présentation, c'est une fenêtre qui sera lancée selon le besoin en show ou showDialog (modeless et modal).

1. XAML

```
<Window x:Class="Demonstration.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Demonstration"
        xmlns:views="clr-namespace:Demonstration.Views"
        mc:Ignorable="d"
        Title="MainWindow" Height="400" Width="630">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>

        <Grid x:Name="MultiCol" Grid.Row="0" Grid.Column="1">
            <views:DataGridPages x:Name="DGControl"/>
        </Grid>
        <Grid x:Name="ScrenPages" Grid.Row="0" Grid.Column="2">
            <views:LesPages x:Name="PagesControl" Loaded="PagesControl_Loaded"/>
        </Grid>
    </Grid>
</Window>
```

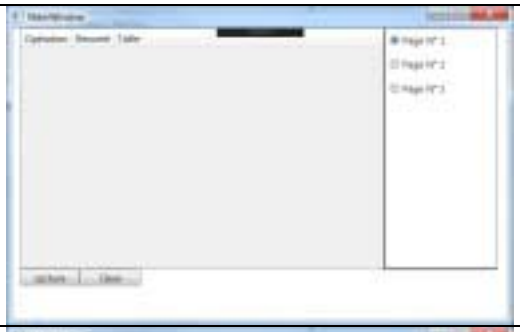
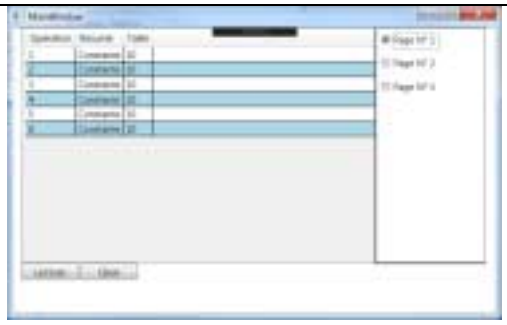
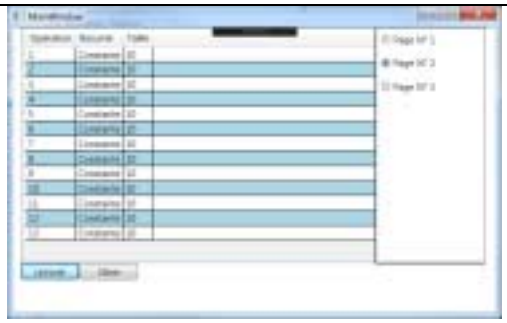
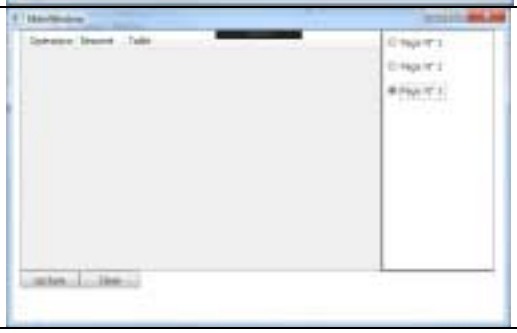
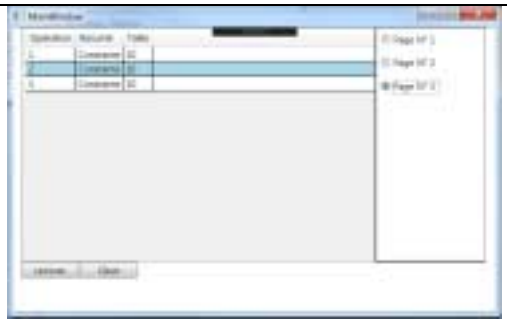
2. Le code behind

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Demonstration.ViewModel;

namespace Demonstration
{
    /// <summary>
    /// Logique d'interaction pour MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void PagesControl_Loaded(object sender, RoutedEventArgs e)
        {
            LesPagesVM PagesModelObject = new LesPagesVM();
            PagesControl.DataContext = PagesModelObject;
```

```
            }
        }
}
```

Résultat de la présentation

Le bouton lecture alimente la DataGrid par des lignes constants (à part le numéro de la ligne – operation).
Chaque click génère une ligne

Page N°	Premier passage	Après n click-Lecture
1		
2		
3		

On peut utiliser les RadioButtons pour passer d'une page à l'autre.
Si on efface avec Clear une des pages les autres restent aptes à afficher leurs contenus.