Ecole Nationale des Sciences Appliquées de Safi

A.U:2023-2024

PlateForme C#.Net

Ing. Meryem OUARRACHI

Plan

- **L’environnement .Net**
- **Initiation à la programmation C#**
- **Programmation Orienté Objet C#**
- **Programmation avancée en C#**

WPF

Plan du chapitre

1-Les contrôles WPF

2-Les triggers en wpf

3-Les animations en wpf

4-Le binding en WPF

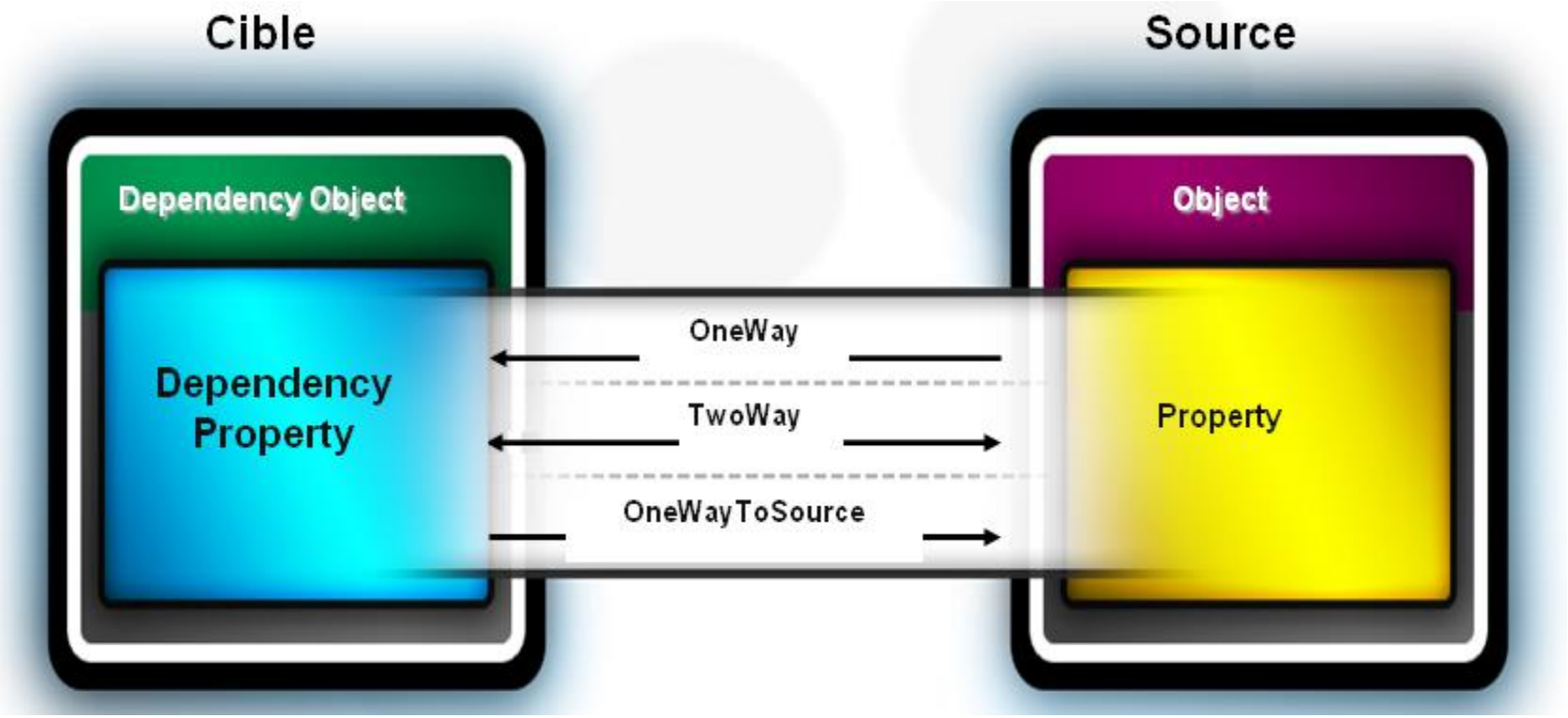
5-Les validateurs en wpf

6-Les bibliothéques en wpf

Binding

DataBinding

Data binding est une technique permettant de lier un contrôle avec une ou plusieurs sources de données.



DataBinding

- **Les propriétés de Binding:**

-ElementName: définit le nom de l'élément à utiliser comme objet de source de liaison.

-Path: Obtient la propriété de source de liaison.

-Mode: OneWay,OneWayToSource, TwoWay

DataBinding

- **Exemple:** Lier le contenu de deux TexteBox

```
<TextBox Height="23" HorizontalAlignment="Left" Margin="12,12,0,0" Name="textBox1"
  VerticalAlignment="Top"   Width="120" />
```

```
<TextBox Height="23" HorizontalAlignment="Right" Margin="0,12,40,0" Name="textBox2"
  VerticalAlignment="Top" Width="120" Text="{Binding ElementName=textBox1,Path=Text, Mode=OneWay}"/>
```

Static Ressource

Objectif: Lier les composants de WPF à des objets.Net

-Méthode classique: Passer par le code behind

Exemple:

datagrid1.itemsSources=operation.getFiliere();

Static Ressource

-Méthode2:Passer par le fichier XAML

Si on veut ajouter une ressource externe à partir de laquelle on va faire le Binding de nos composants:

1. Ajouter une ligne de référence dans la balise Windows

<xmlns:NomProjet="clr-namespace:VotreNamespace">

2.Pour utiliser la source

<DataGrid HorizontalAlignment="Left"

ItemsSource="{Binding Source={x:Static NomProjet:maListe}}"

Static Ressource

Exemple:

1.

```
ObservableCollection<Filiere> fl= new ObservableCollection<Filiere>(db.Filiere.ToList());
```

2.

```
DataContext="{Binding Source={x:Static y:FiliereOperation.fl}}">
```

3.

```
<DataGrid  ItemsSource="{Binding Source={x:Static y:FiliereOperation.fl}}"/>
```

DataContext

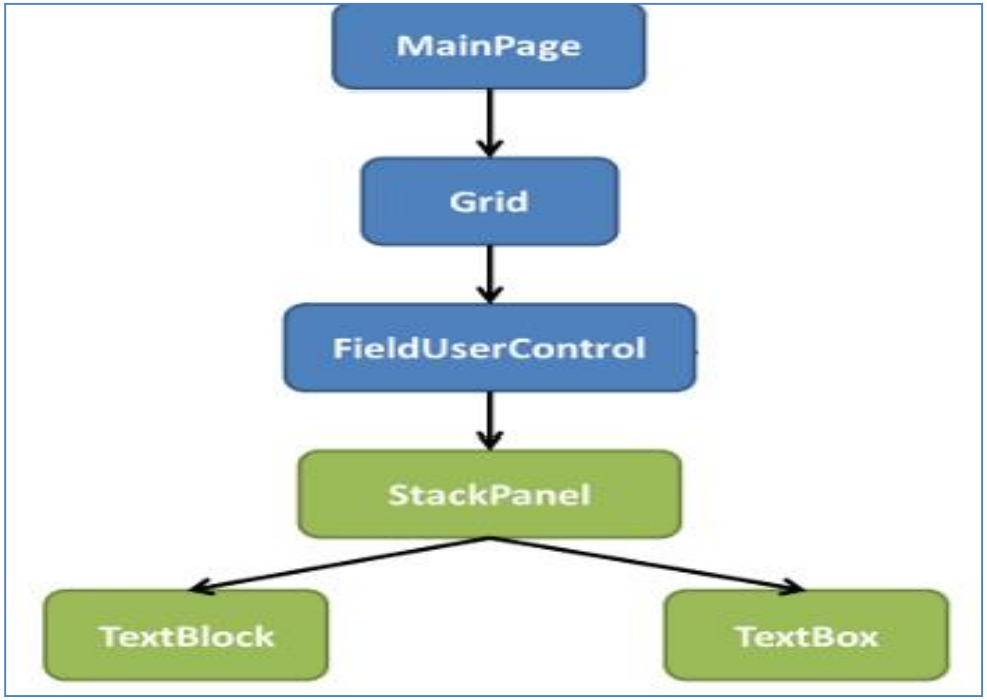
-La propriété DataContext est la source de nos fixations par défaut,elle est définie dans FrameworkElement.

→ elle est donc à la disposition de tous les contrôles WPF

-Elle est automatiquement héritée par tous les enfants de l'élément où elle est définie

→ il n'est donc pas nécessaire de la redéfinir pour chaque élément qu'on veut lier aux données.

DataContext



-Son type de données est System.Object .

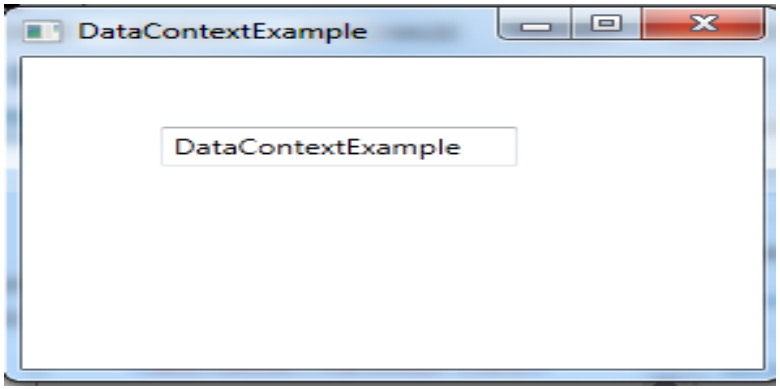
DataContext

-Exemple

1. public partial class DataContextExample : Window

```
{ public DataContextExample ( )
    { InitializeComponent ();
        this.DataContext = this;  } }
```

2. <TextBox **Text = "{Binding Title}"** width = "50" />



DataContext

-Remarque: on a la possibilité de définir le Datacontext dans le code XAML

```
DataContext="{Binding Source={x:Static
y:FiliereOperation.liste1}}">
```

-Pour l'appeler

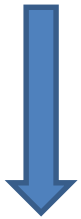
```
<DataGrid ItemsSource="{Binding}">
<DataGrid.Columns>
 <DataGridTextColumn Binding="{Binding Id_filiere}"
  Header="ID"/>
<DataGridTextColumn Binding="{Binding Nom_filiere}"
Header="Nom"/>
```

Validateur /Convertisseur

Convertor

-Convertisseurs de valeur sont utilisés dans la liaison de données. Lorsque le type d'objet source est différent du type d'objet cible.

→La source doit être convertie avant qu'elle atteint le cible.



Définir une classe Convertor

Convertor

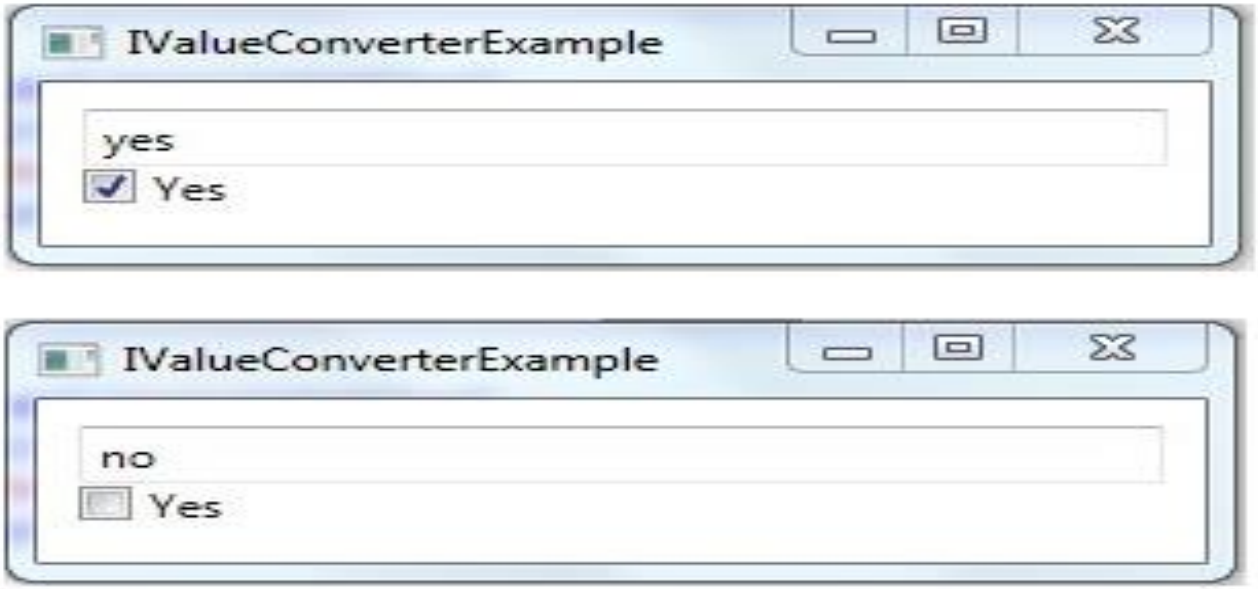
-La classe convetor doit implémenter l’interface IValueConvertor

-La IValueConverter interface se compose de deux méthodes:

- **Convert:** est appelée lorsque la source met à jour l'objet cible.
- **ConvertBack:** est appelée lorsque la cible met à jour l'objet source.

Convertor

Exemple



IsChecked(boolean) # Text (string)

Convertor

-Définir la classe de conversion

```
public class YesNoToBooleanConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter,
            System.Globalization.CultureInfo culture)
    {
        switch (value.ToString().ToLower())
        {
            case "yes":
                return true;
            case "no":
                return false;

            default:
                return Binding.DoNothing; }
    }
```

Convertor

-Définir la classe de conversion

```
public object ConvertBack(object value, Type targetType, object parameter,
        System.Globalization.CultureInfo culture)
{
    if (value is bool)
    {
        if ((bool)value == true)
            return "yes";
        else
            return "no";
    }
    return "no";
}
```

Convertor

-Utiliser la classe de conversion lors de Binding

```
<Window x:Class="WpfNotification.Window3"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Window3" Height="300" Width="300"
        xmlns:local="clr-namespace:WpfNotification"
        >
    <Window.Resources>
        <local:YesNoToBooleanConverter x:Key="YesNoToBooleanConverter"/>
    </Window.Resources>
    <Grid>
        <TextBox Name="txtValue" HorizontalAlignment="Left" Height="23" Margin="10,32,0,0" TextWrapping="Wrap"
                 Text="TextBox" VerticalAlignment="Top" Width="120"/>
        <CheckBox IsChecked="{Binding ElementName=txtValue,
                                      Path=Text,
                                      Converter={StaticResource YesNoToBooleanConverter}}"  Content="Yes"/>
```

Validation des données

Pour valider les données des champs en wpf:

Etape 1: Créer modèle de données avec IDataErrorInfo.

Etape 2: Lier les données à nos champs d'entrée.

Etape 3: Personnaliser le modèle d’affichage d'erreur.

Validation des données

Etape 1:Création du modèle de données avec IDataErrorInfo.

```
public class Employee : IDataErrorInfo
{
    public string Name { get; set; }

    public string Position { get; set; }

    public int Salary { get; set; }

    public string Error
    {
        get { throw new NotImplementedException(); }
    }
```

Validation des données

Etape 1:Création du modèle de données avec IDataErrorInfo.

```
public string this[string columnName]
{get  {string result = null;

    if (columnName == "Name")
        {
            if (string.IsNullOrEmpty(Name) || Name.Length < 3)
                result = " enter un  Nom"; }

    if (columnName == "Position")
        {
            if (string.IsNullOrEmpty(Position) || Name.Length < 3)
                result = "enter une Position"; }

    if (columnName == "Salary")
        {
            if (Salary <= 1000 || Salary >= 50000)
                result = "enter un saliare valide";  }
        return result;
       }
```

Validation des données

Etape2: Lier les données à nos champs d'entrée.

```
<TextBox x:Name="textBox_Name"
   Text="{Binding  UpdateSourceTrigger=PropertyChanged,Path=Name,
            ValidatesOnDataErrors=true}" />
```

Validation des données

Etape3: Personnaliser le modèle d'affichage d'erreur

-Par défaut l'erreur est signaler par une bordure rouge sur le champ concerné.

-On a la possibilité de changer le style d'affichage d'erreur

Exemple: afficher le message dans le toolTip quand la souris passe sur le champ.

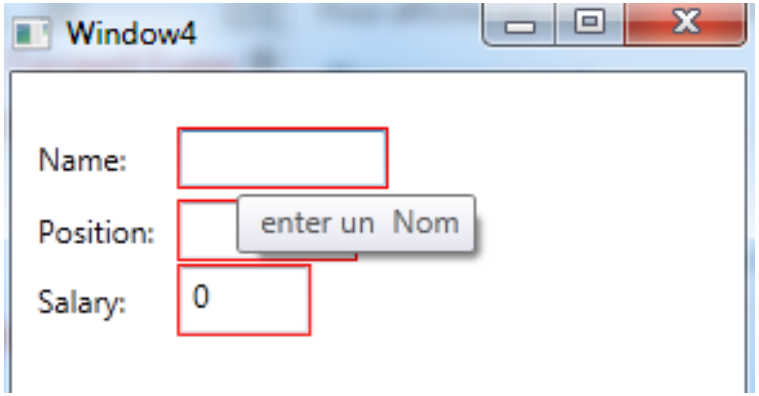
Validation des données

Etape3: Personnaliser le modèle d'affichage d'erreur

-Par défaut l'erreur est signalé par une bordure rouge sur le champ concerné.

-On a la possibilité de changer le style d'affichage d'erreur

Exemple: afficher le message dans le toolTip quand la souris passe sur le champ.



Validation des données

Etape3: Personnaliser le modèle d'affichage d'erreur

```
<Window.Resources>
     <Style TargetType="{x:Type TextBox}">
        <Style.Triggers>
            <Trigger Property="Validation.HasError" Value="true">
                <Setter Property="ToolTip"
            Value="{Binding RelativeSource={RelativeSource Self},
        Path=(Validation.Errors)[0].ErrorContent}"/>
            </Trigger>
        </Style.Triggers>
    </Style>
</Window.Resources>
```

Notification de changement

Objectif: Le composant wpf devra être notifier automatiquement par les modifications effectuées dans la source de données

Activer la notification dans le composant

Notification de changement

1.En implémentant l'interface INotifyPropertyChanged et ajouter un événement OnPropertyChanged à chaque changements.

```
public partial class Filiere : INotifyPropertyChanged
{
    private int _Id_filiere;
    public int Id_filiere
    {
        get { return _Id_filiere; }
        set { _Id_filiere = value; OnPropertyChanged("Id_filiere"); }
    }
```

Notification de changement

2. L’événement PropertyChanged est déclenché lorsqu'une propriété est modifiée sur un composant.Un objet PropertyChangedEventArgs spécifie le nom de la propriété modifiée.

- L'événement PropertyChanged peut indiquer que toutes les propriétés de l'objet ont été modifiées à l'aide de null

Notification de changement

```
public event PropertyChangedEventHandler PropertyChanged;
//evenement se produit en cas de modification d'une valeur
protected void OnPropertyChanged(string name)
{
    PropertyChangedEventHandler handler = PropertyChanged;
    //handler est un delegate
    if (handler != null)
    {
        handler(this, new PropertyChangedEventArgs(name));
    }
}
```

Notification de changement

-Puis, pour chaque propriété pour laquelle vous souhaitez obtenir des notifications de modification, vous appelez OnPropertyChanged à chaque fois que la propriété est mise à jour.

Remarque: Lors de l'utilisation de Linq to sql cette interface s'hérite dans les classes générées.

UpdateSourceTrigger

-La propriété UpdateSourceTrigger contrôle le minutage des mises à jour de la source de liaison.

La propriété TextBox.Text a une valeur par défaut UpdateSourceTrigger de LostFocus. Cela signifie que si une application a un TextBox avec une propriété TextBox.Text liée aux données, le texte que vous saisissez dans le TextBox ne met pas à jour la source jusqu'à ce que le TextBox perde le focus (par exemple, lorsque vous cliquez en dehors de TextBox).

UpdateSourceTrigger

Si on souhaite que la source soit mise à jour pendant notre saisie, on doit défnir le UpdateSourceTrigger par la valeur PropertyChanged.

```
<TextBox  Text="{Binding Nom,UpdateSourceTrigger=PropertyChanged}"
  Name="textBox5" Margin="74,91,0,0"  VerticalAlignment="Top" Width="120" Grid.Column="1" />
```

ObservableCollection

C’est une collection qui permet au code en dehors de la collection être conscient lors de modifications apportées à la collection (ajouter, déplacer, supprimer) se produit.

Résumé pour les notifications

Type de notification	Méthode
Notification de changement de valeur	Utiliser l'interface InotifyPropertyChanged
Notification d'ajout/suppression	Utiliser la collection ObservableCollection

Bibliothèque en WPF

Telerik for wpf

- **Définition:**

-Telerik for wpf est une bibliothèque qui offre un ensemble des contrôles pour créer une interface graphique ergonomique.

-Ces contrôles permettent d'avoir une couche présentation de haute performance et très attrayante.

Telerik for wpf

- **Installation:**

A partir de site officiel installer telerik for wpf

https://www.telerik.com/download-trial-file/v2/ui-for-wpf

→Une fois installer les contrôles vont être ajouté automatiquement aux projets wpf de Visual studio

Telerik for wpf

- **Documentation***:*

-visiter leur site il y'a des exemples pour tous les contrôles:

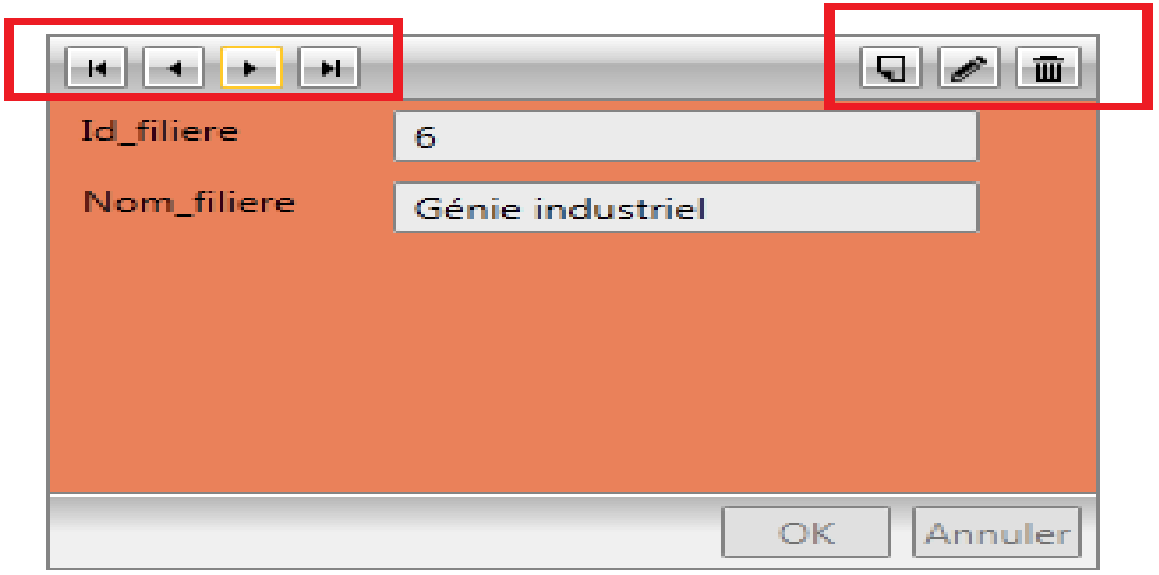
http://docs.telerik.com/devtools/wpf/introduction

-Cliquer sur le bouton Demos afin de télécharger des démonstrations avec les codes sources

Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadDataForm:** Composant qui affiche les données et facilite les mises à jour dans la base de donnée



Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadDataForm:**

-Pour le remplissage:

```
private void RadDataForm_Loaded(object sender, RoutedEventArgs e)
{    FiliereOperation f = new FiliereOperation();
     RadDataForm1.ItemsSource = f.getAllFiliere();
}
```

-Pour valider la suppression:

```
private void RadDataForm1_DeletingItem(object sender, System.ComponentModel.CancelEventArgs e)
{ Filiere f1 = RadDataForm1.CurrentItem as Filiere;
  FiliereOperation f = new FiliereOperation();
  f.deleteFiliere(f4.Id_filiere);
}
```

Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadDataForm:**

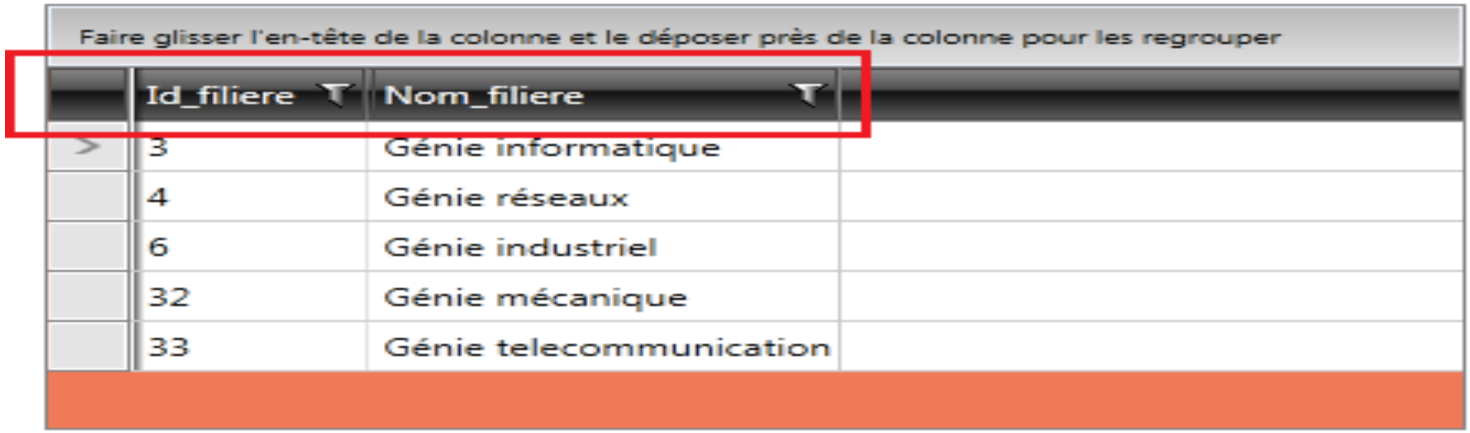
  -Pour la validation des modifications(Add+Update) dans la BD:

  -Créer une méthode qui transforme le contenu de RadDataForm vers la table à modifier dans la base de donnée.

Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadGridView:** Composant qui affiche les données



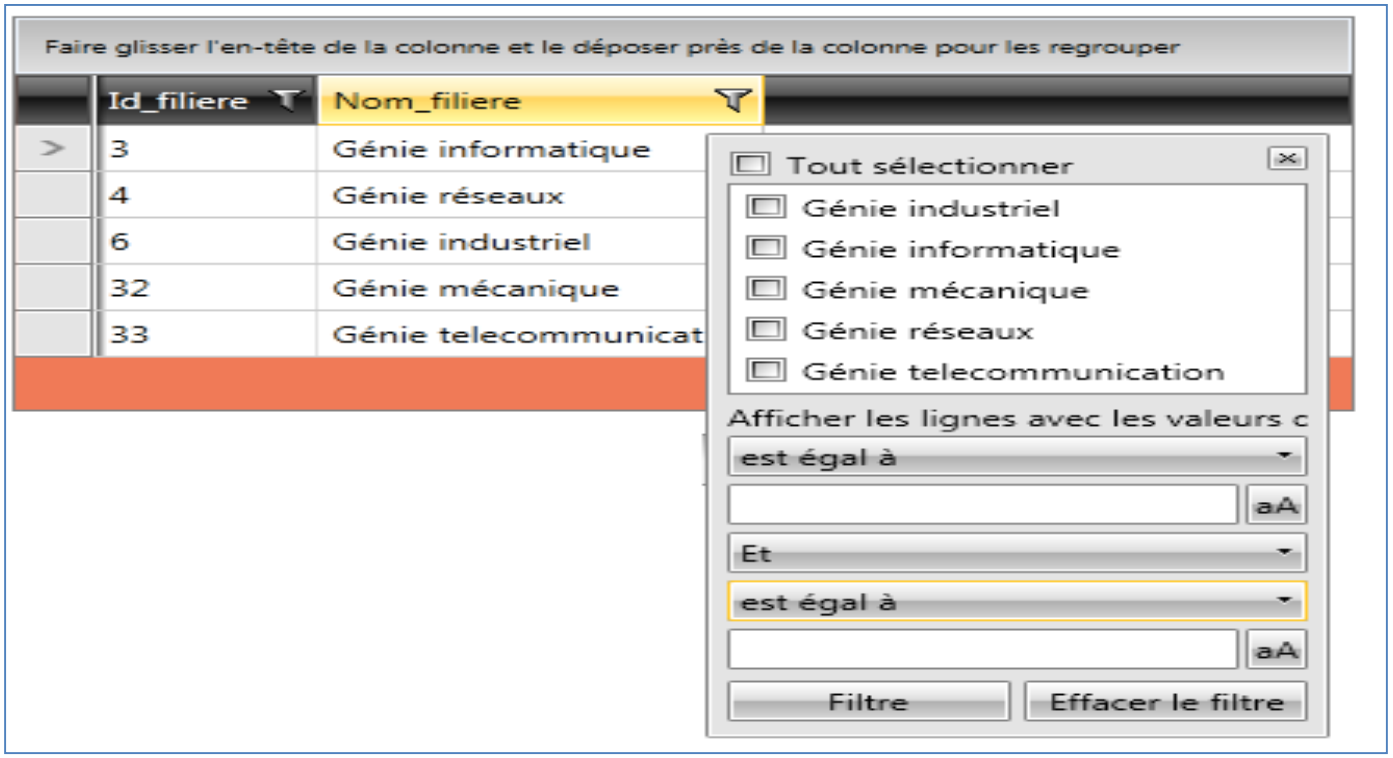
-Pour le remplissage:

```
private void RadGridView_Loaded(object sender, RoutedEventArgs e)
{   FiliereOperation f = new FiliereOperation();
    RadGridView1.ItemsSource = f.getAllFiliere();
}
```

Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadGridView:** permettant de trier les données et faire des filtres sur ces données.



Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadDataPage:** Composant qui organise la pagination d'un autre composant.

Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadDataPage:**

-Exemple: Lier RadDataPage à un DataGridView.


```
<telerik:RadGridView  Name="RadGridView1"   Background="#FFF07A57"
                      Loaded="RadGridView_Loaded" />
<telerik:RadDataPager Source="{Binding Items, ElementName=RadGridView1}"
                  PageSize="5" />
```

Exemple des contrôles Telerik for wpf

1.Contrôles de données

- **RadCarousel:** RadCarousel pour WPF est un contrôle innovant pour la navigation interactive des données, en utilisant des trajectoires de mouvement circulaire

Exemple des contrôles Telerik for wpf

1.Contrôles de données

-Pour le remplissage:

```
private void RadCarousel_Loaded(object sender, RoutedEventArgs e)
{   FiliereOperation f = new FiliereOperation();
    RadCarousel.ItemsSource = f.getAllFiliere3();
}
```

Exemple des contrôles Telerik for wpf

1.Contrôles de données

-Personnaliser le contenu:

```
<Window.Resources>
 <Style TargetType="{x:Type telerik:CarouselDataRecordPresenter}">
   <Setter Property="Template">
    <Setter.Value>
        <ControlTemplate >
        <!-- le contenu -->
        </ControlTemplate>
    </Setter.Value>
   </Setter>
 </Style>
</Window.Resources>
```

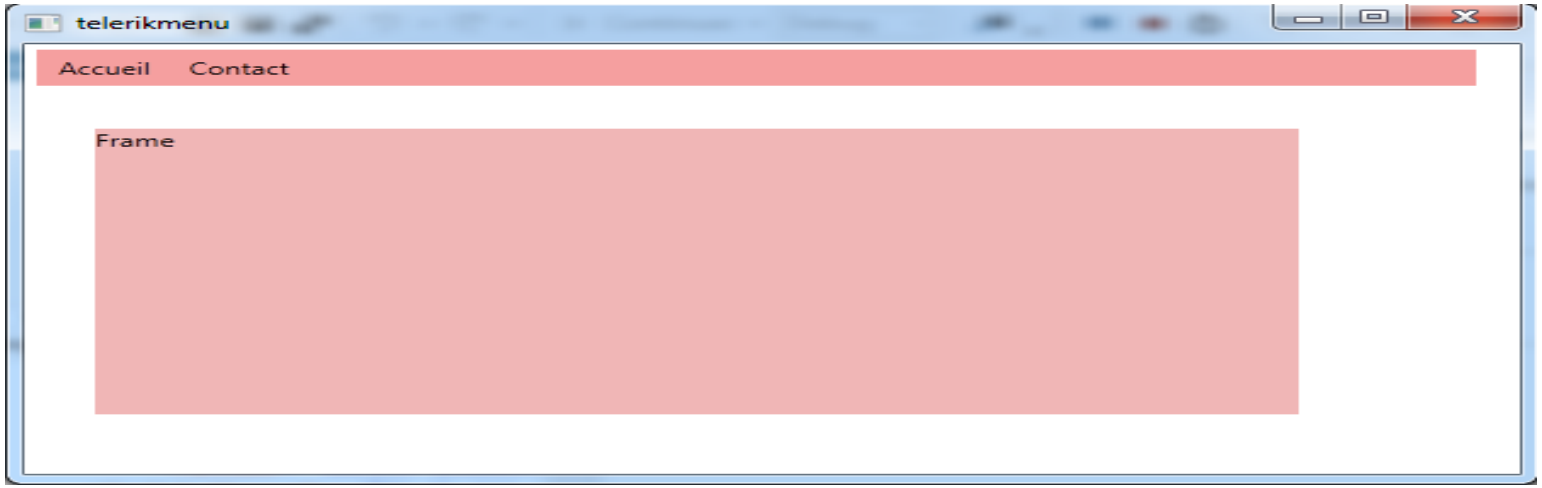
Exemple des contrôles Telerik for wpf

2.Navigation.

- **Frame:** est un contrôle de contenu qui prend en charge la navigation.(contrôle de wpf).

-Le contenu de Frame est un usercontrol

```
UserControl1 a = new UserControl1();
frame1.Content = a;
```

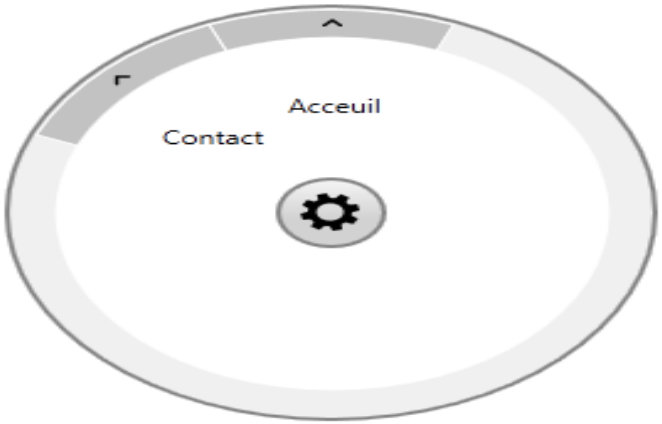


Exemple des contrôles Telerik for wpf

2.Navigation.

- **RadMenu:** Composant qui permet de créer un menu horizontal et Vertical (selon l'orientation)

- **RadRadialMenu:**



Autres Bibliothèques for WPF

On trouve aussi:

- Bibliothèque: syncfusion for wpf

https://www.syncfusion.com/products/wpf

- Bibliothèque: Toolkit