

WPF

Programmation événementielle client lourd
Partie 2

Sommaire

- **Binding**
- **Styles**
- **Validation des données**
- **Persistiance / Base de données**

Concept de binding

- À maîtriser pour une utilisation optimale des capacités de WPF.
- Liaison **unidirectionnelle** ou **bidirectionnelle** entre un élément XAML et un contexte de données (DataContext) et permet la propagation des données du code C# vers le code XAML (ou inversement) sans être obligé de faire du code C#.
- Mécanisme qui est la clé du **découpage** entre le code métier et l'interface graphique.
- La mise en œuvre de cette séparation est le plus souvent effectuée à l'aide du patron de conception **MVVM (Model – View – ViewModel)**

Binding

- Répercuter les modifications vue/C# dans les deux sens !
- 3 Objets : source, cible, binding
- 3 modes :
 - OneWay : source vers cible
 - OneWayToSource : cible vers source
 - **TwoWays : les 2 sens (par défaut)**

Binding : exemple simple

➊ Une classe métier Personne(nom,age)

```
3 références
public class Personne
{
    private String nom;
    2 références
    public String Nom
    {
        get { return nom; }
        set { nom = value.ToUpper(); }
    }

    private int age;
    2 références
    public int Age
    {
        get { return age; }
        set { age = value; }
    }

    1 référence
    public Personne(string nom, int age)
    {
        Nom = nom;
        Age = age;
    }
}
```

B. DIARD

Binding : exemple simple

➋ Une vue avec 2 champs de saisie :



```
<Label Content="Nom : " HorizontalAlignment="Left" Margin="20,10,0,0"
<TextBox x:Name="nomTexte" Text="{Binding Nom}" HorizontalAlignment="L
<Label Content="Age : " HorizontalAlignment="Left" Margin="20,50,0,0"
<TextBox x:Name="ageTexte" Text="{Binding Age, Mode = TwoWay}" Horizon
<Button Content="Verif" Click="Button_Click" HorizontalAlignment="Left"
```

➌ Un binding sur le nom et l'âge

Pas obligatoire : par défaut

Binding : exemple simple

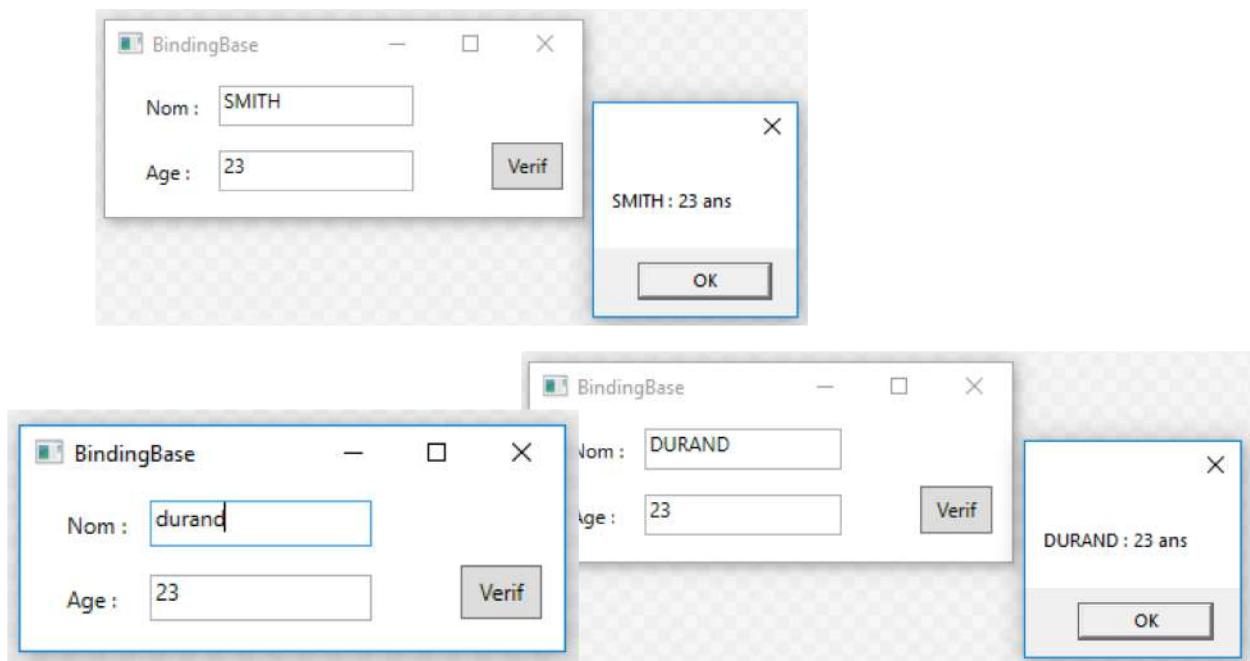
Code behind :

```
public partial class MainWindow : Window
{
    Personne p;
    public MainWindow()
    {
        InitializeComponent();
        p = new Personne("Smith", 23);
        this.DataContext = p; ← Définition de la source de données de façon explicite grâce au contexte de données
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show(p.Nom + " : " + p.Age + " ans");
    }
}
```

Binding : exemple simple

Propagation des données de façon bidirectionnelle :



Binding : interface INotifyPropertyChanged

- ➊ On peut utiliser aussi l'interface **INotifyPropertyChanged**
- ➋ Il faut alors ajouter la définition de la propriété dans la classe **MainWindow**.
- ➌ On applique le **DataContext** à la fenêtre en cours (permet de gérer plusieurs composants)
- ➍ Puis il faut gérer l'événement **PropertyChanged** pour permettre la propagation de la modification

Binding : interface INotifyPropertyChanged

```
public partial class MainWindow : Window,INotifyPropertyChanged
{
    private String nom;

    2 références
    public String Nom
    {
        get { return nom; }
        set { nom = value.ToUpper();
              OnPropertyChanged();
        }
    }

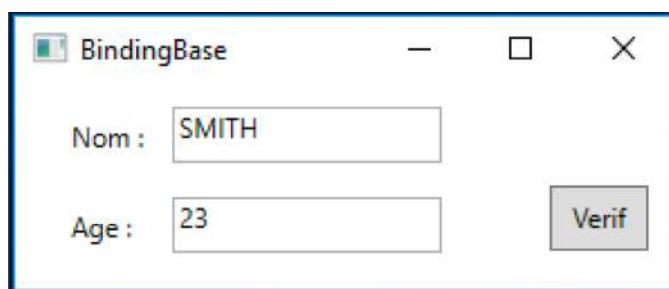
    private int age;

    2 références
    public int Age
    {
        get { return age; }
        set { age = value;
              OnPropertyChanged();
        }
    }
}
```

Binding : interface INotifyPropertyChanged

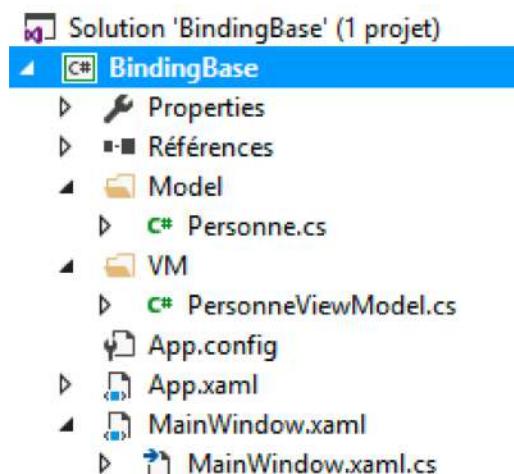
```
public MainWindow()
{
    InitializeComponent();
    this.DataContext = this;
    this.Nom = "Smith";
    this.Age = 23;
}

1 référence
private void Button_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show(Nom + " : " + Age + " ans");
}
```



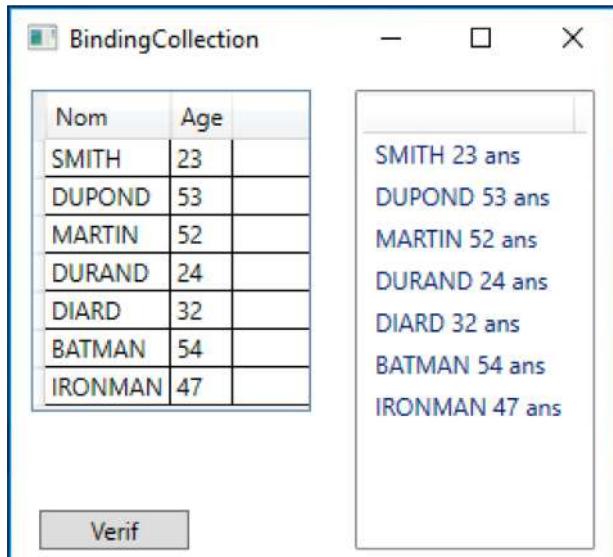
Binding : interface INotifyPropertyChanged

- ➊ Inconvénient : et si on veut utiliser des classes métier ?
- ➋ Il faut créer une classe intermédiaire VM (Vue - Model)
- ➌ Principe du pattern MVVM
- ➍ Exemple complet (démonstration) :



Binding : Collections

- ➊ Beaucoup plus simple.
- ➋ On utilise une ObservableCollection.



B. DIARD

13

Binding : Collections

```
<Grid>
    <DataGrid ItemsSource="{Binding ListePersonnes}" />
    <ListView ItemsSource="{Binding ListePersonnes}" />
    <Button Content="Verif" HorizontalAlignment="Left" />
</Grid>

public partial class MainWindow : Window
{
    9 références
    public ObservableCollection<Personne> ListePersonnes { get; set; }

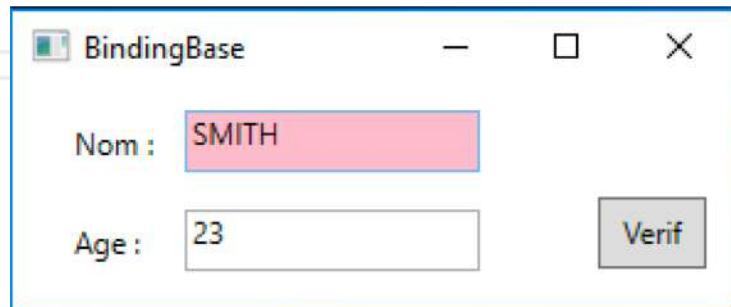
    0 références
    public MainWindow()
    {
        InitializeComponent();
        this.DataContext = this;
        ListePersonnes = new ObservableCollection<Personne>();
        ListePersonnes.Add(new Personne("Smith", 23));
        ListePersonnes.Add(new Personne("Dupond", 53));
        ListePersonnes.Add(new Personne("Martin", 52));
        ListePersonnes.Add(new Personne("Durand", 24));
        ListePersonnes.Add(new Personne("Diard", 32));
        ListePersonnes.Add(new Personne("Batman", 54));
        ListePersonnes.Add(new Personne("IronMan", 47));
    }
}
```

B. DIARD

Définir des styles

Style générique sur un TextBox avec changement d'arrière plan au survol

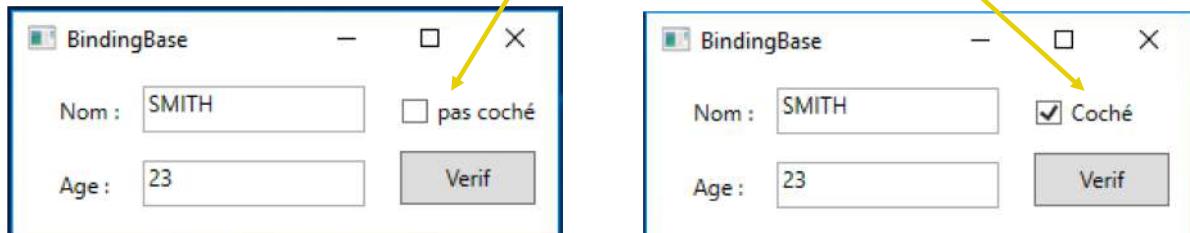
```
<Window.Resources>
    <Style TargetType="{x:Type TextBox}">
        <Setter Property="Height" Value="25"/>
        <Setter Property="Width" Value="120"/>
        <Style.Triggers>
            <Trigger Property="Control.IsMouseOver" Value="True">
                <Setter Property="Background" Value="Pink"/>
            </Trigger>
        </Style.Triggers>
    </Style>
</Window.Resources>
```



Définir des styles

Déclenché sur un changement de valeur :

```
<Style TargetType="{x:Type CheckBox}">
    <Setter Property="Content" Value="pas coché"/>
    <Style.Triggers>
        <Trigger Property="IsChecked" Value="True">
            <Setter Property="Content" Value="Coché"/>
        </Trigger>
    </Style.Triggers>
</Style>
```



Définir des styles

- ➊ Définir un style en fonction d'une valeur :
- ➋ Créer une classe implémentant **IValueConverter**

```
public class AgeToColorConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        int age;
        Int32.TryParse(value.ToString(), out age);
        return (age >= 18 ? Brushes.LightGreen : Brushes.OrangeRed);
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }
}
```

B. DIARD

17

Définir des styles

- ➊ Ajouter une ressource à la classe en XAML :

```
<local:AgeToColorConverter x:Key="AgeToColorConverter" />
</Window.Resources>
```

- ➋ Puis un binding sur l'arrière plan faisant appel à la classe :

```
<local:AgeToColorConverter x:Key="AgeToColorConverter" />
</Window.Resources>
<Grid>
    <Label Content="Nom : " HorizontalAlignment="Left" Margin="20,10,0,0" VerticalAlignment="Top"/>
    <TextBox x:Name="nomTexte" Text="{Binding Nom}" HorizontalAlignment="Left" Margin="70,10,0,0" TextWrapping="Wrap" />
    <Label Content="Age : " HorizontalAlignment="Left" Margin="20,50,0,0" VerticalAlignment="Top"/>
    <TextBox x:Name="ageTexte" Background="{Binding Path=Age, Converter={StaticResource AgeToColorConverter}}"/>

```

B. DIARD

18

Définir des styles

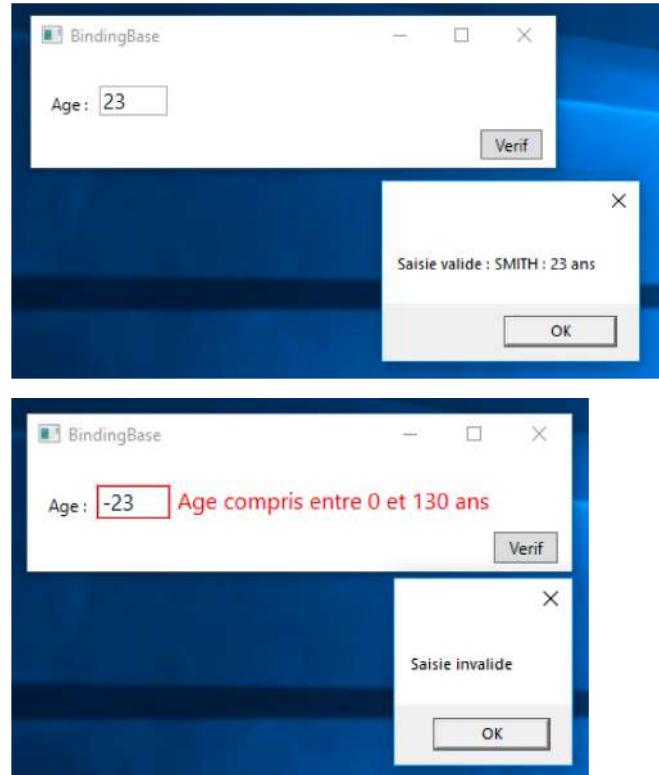
The image displays two screenshots of a Windows application window titled "BindingBase". Both screenshots show a form with fields for "Nom" (Name) and "Age" (Age). A checkbox labeled "pas coché" (unchecked) is also present. In the first screenshot, the "Age" field contains the value "12", which is highlighted with a red background, indicating an error. In the second screenshot, the "Age" field contains the value "23", which is highlighted with a green background, indicating that the validation has passed.

Validation des données

- ➊ Première possibilité : coder le setter du modèle pour déclencher une exception traitée dans le binding
- ➋ Seconde possibilité : Utiliser Validation Rules (plus générique)
 - ➌ Exemple Microsoft : <https://docs.microsoft.com/fr-fr/dotnet/framework/wpf/data/how-to-implement-binding-validation>

Validations Rules

Exemple :



Validation Rules

Création d'une classe qui implémente l'interface ValidationRules :

```
public class AgeRangeRule : ValidationRule
{
    private int _min;
    private int _max;

    0 références
    public AgeRangeRule()...

    2 références
    public int Min...

    2 références
    public int Max...

    0 références
    public override ValidationResult Validate(object value, CultureInfo cultureInfo)
    {
        int age = 0;
        try
        {
            if (((string)value).Length > 0)
                age = Int32.Parse((String)value);
            catch (Exception e)
            {
                return new ValidationResult(false, "caractere illégal");
            }

            if ((age < Min) || (age > Max))
            {
                return new ValidationResult(false,
                    "Age compris entre " + Min + " et " + Max + " ans");
            }
            else
            {
                return ValidationResult.ValidResult;
            }
        }
    }
}
```

Validation Rules

- Création d'un template TextBlock pour l'affichage du message :

```
<ControlTemplate x:Key="txtError">
    <StackPanel Orientation="Horizontal">
        <Border BorderBrush="Red" BorderThickness="1">
            <AdornedElementPlaceholder x:Name="element"/>
        </Border>
        <TextBlock x:Name="tbError" Foreground="Red" Margin="5,0,0,0" FontSize="12pt"
            Text="{Binding ElementName=element, Path=AdornedElement.(Validation.Errors)[0].ErrorContent}"/>
    </StackPanel>
</ControlTemplate>
</Window.Resources>
<Grid>
```

Validation Rules

- Liaison de la classe créée avec la vue :

```
<TextBox Name="textBox1" FontSize="15"
    HorizontalAlignment="Left" Margin="50,21,0,0" VerticalAlignment="Top" Width="50"
    Validation.ErrorTemplate="{StaticResource txtError}">
    <TextBox.Text>
        <Binding Path="Age" UpdateSourceTrigger="PropertyChanged">
            <Binding.ValidationRules>
                <local:AgeRangeRule Min="0" Max="130"/>
            </Binding.ValidationRules>
        </Binding>
    </TextBox.Text>
</TextBox>
```

- Rq : Détection des erreurs :

```
Oréférences
private void Button_Click(object sender, RoutedEventArgs e)
{
    if (Validation.GetHasError(textBox1) == false)
        MessageBox.Show("Saisie valide : " + p.Nom + " : " + p.Age + " ans");
    else
        MessageBox.Show("Saisie invalide");
}
```

Persistante

Rappel : Utilisation de JSON (Sérialisation / désérialisation)

The screenshot shows a Windows application window titled "Personnes.json" with the tab "MainWindow.xaml.cs" selected. The status bar indicates "Schéma : <Aucun schéma sélectionné>". The code editor displays the following JSON content:

```
1   [  
2     {"Nom": "Durand", "Age": 32},  
3     {"Nom": "Dupond", "Age": 54},  
4     {"Nom": "Batman", "Age": 72},  
5     {"Nom": "Diard", "Age": 32}  
6   ]
```

To the right, a small window titled "B..." shows a table with the following data:

Nom	Age
DURAND	32
BATMAN	54
SMITH	72
DIARD	32

A "Sauve" button is at the bottom of the small window.

Persistante

```
public void ChargeListePersonnes()  
{  
    try  
    {  
        ListePersonnes = JsonConvert.DeserializeObject<ObservableCollection<Personne>>(File.ReadAllText("./Data/Personnes.json"));  
    }  
    catch  
    {  
        MessageBox.Show("Lecture fichier impossible, l'application va se fermer"  
            , "erreur", MessageBoxButton.OK, MessageBoxImage.Error);  
        Environment.Exit(0);  
    }  
}  
  
1 référence  
public void SauveListePersonnesJson()  
{  
    try  
    {  
        File.WriteAllText("./Data/Personnes.json", JsonConvert.SerializeObject(ListePersonnes));  
    }  
    catch  
    {  
        MessageBox.Show("Ecriture fichier impossible", "erreur", MessageBoxButton.OK, MessageBoxImage.Error);  
    }  
}
```

Base de données

- Pour des données plus complexes, il est préférable d'utiliser une base de données.
- Il est possible d'utiliser une base de données interne (sqlexpress) ou un serveur sql externe.
- En local il suffit d'ajouter un objet de type Base de données.
- L'explorateur de serveur permet de configurer la base de données.

Base de données

Exemple : Base de données locale

The screenshot shows the SSMS interface with the following components:

- Object Explorer (Left):** Shows the project structure under "BindingCollection". It includes "Properties", "Références", "Data", "Model", "App.config", "App.xaml", and the database "Database1.mdf" which contains "Database1_Log.ldf".
- Server Explorer (Top Right):** Shows "Connexions de données" and "Database1.mdf" which contains "Tables", "Vues", "Procédés", "Fonctions", "Synonymes", "Types", and "Assemblages". A table named "PERSONNES" is selected.
- Table Definition (Center):** A detailed view of the "PERSONNES" table structure. It has three columns: "Id" (int, primary key, identity), "Nom" (nchar(50)), and "Age" (int). The "Nom" column has a checkmark in the "Autoriser les valeurs NULLS" checkbox.
- Script Editor (Bottom Left):** Displays the T-SQL code for creating the "PERSONNES" table:

```
CREATE TABLE [dbo].[PERSONNES]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Nom] NCHAR(50) NULL,
    [Age] INT NULL
)
```
- Properties Window (Bottom Right):** Shows properties for the "Id" column of the "PERSONNES" table, including "Début du compteur" set to 1 and "Incrément d'identité" set to 1.

Base de données

- Comme pour la persistance, un accès en lecture/écriture permet de charger/sauver les données.

- Ex : Lecture → chargement dans notre liste de personnes

```
public void ChargeListePersonnes()
{
    SqlConnection conn =
        new SqlConnection("Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename=|DataDirectory|\Database1.mdf; Integrated Security = True");
    conn.Open();
    string sql = "SELECT NOM,AGE FROM PERSONNES";
    SqlCommand requeteSQL = new SqlCommand(sql, conn);
    DataTable dataTable = new DataTable();
    SqlDataReader rdr = requeteSQL.ExecuteReader();
    dataTable.Load(rdr);
    for (int i = 0; i < dataTable.Rows.Count; i++)
    {
        Personne p = new Personne(dataTable.Rows[i]["NOM"].ToString(), Convert.ToInt32(dataTable.Rows[i]["AGE"]));
        ListePersonnes.Add(p);
    }
    rdr.Close();
    conn.Close();
}
```

Base de données

	Id	Nom	Age
	1	DIARD	32
	2	BATMAN	45
	3	DUPOND	54
	4	MICHEL	32
	5	SMITH	34
	NULL	NULL	NULL

Prolongements

- ➊ On peut utiliser une base de données distante, mais il faut disposer d'un serveur SQL (ex : \\jupiter)
 - ➌ Vu en M2104 → AmphiQuizz
- ➋ Pour simplifier le travail, on peut utiliser aussi le framework Entity Framework pour éviter d'accéder directement à la base de données. Il crée une couche d'accès aux données. La communication se fait à travers d'entités définies dans un modèle EDM. Ce modèle est utilisé pour les opérations sur la base de données et la génération des requêtes SQL
- ➌ Autre prolongement : Le patron de conception MVVM qui permet de bien séparer Modèle et vue, mais nécessite une couche intermédiaire VM (binding)
 - ➌ Il existe un framework facilitant le déploiement : MVVM Light Toolkit