Chapitre 5
DataBinding

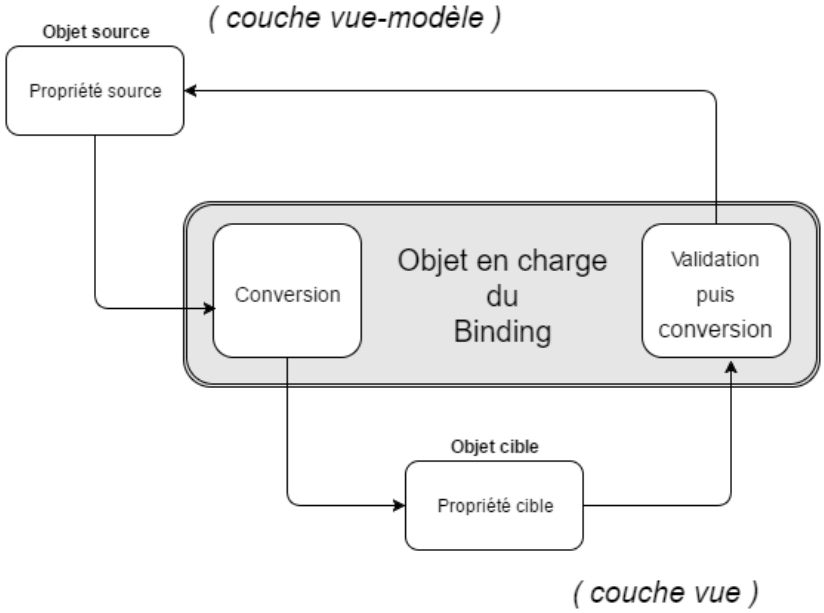
1. Introduction

Le *binding* ou le *binding de données*, que l'on pourrait traduire littéralement par « liaison de données », est un mécanisme absolument central en WPF. Ce mécanisme est d'ailleurs directement hérité de l'implémentation de MVVM par cette technologie.

S'il fallait le définir de façon très générale, le binding peut s'envisager comme le mécanisme qui lie deux sources de données et qui en assure la synchronisation mutuelle.

Ce mécanisme est au cœur de ce qui permet une réelle séparation entre vue et vue-modèle et à ce titre, est réellement le bras armé de MVVM dans le cadre d'un projet codé en WPF.

Ci-dessous, un schéma résumant le mécanisme de binding ainsi que les principaux objets impliqués.



En aucun cas le schéma précédent ne propose un cycle. Il s'agit bien là de mettre en exergue la synchronisation d'une propriété de la vue-modèle vers la vue et réciproquement.

Le schéma précédent utilise plusieurs dénominations qu'il s'agit ici de préciser :

– L'objet source situé dans la couche vue-modèle contient une propriété source qui fait l'objet d'une synchronisation *via* le binding.
– Côté vue, un objet, pas nécessairement visuel, contient une propriété cible qui elle aussi peut faire l'objet d'une synchronisation *via* le binding.
– L'objet Binding à même de réaliser le binding dans les deux sens.

1.1 Binding côté vue exclusivement

Côté vue, dans le code XAML, l'objet ou la propriété bindé utilise une expression « `Binding` » pour réaliser cette synchronisation.

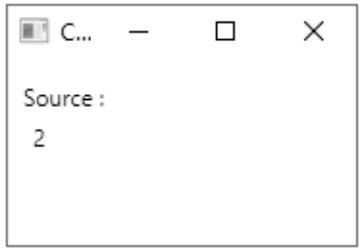
L'objet `Binding` utilise trois propriétés, `Source`, `RelativeSource` ou `ElementName` pour se lier avec l'objet source. À ce stade, il faut préciser que le binding ne se réalise pas nécessairement avec la vue-modèle. Classiquement, la vue-modèle va fréquemment être affectée au `DataContext` de la vue et le binding s'effectue alors préférentiellement avec la vue-modèle. Mais le binding peut également s'appliquer sans lien aucun avec la vue-modèle. Ce type de binding statique utilise l'une des trois propriétés explicitées ci-dessus.

1.1.1 Propriété Source

Cette propriété permet de spécifier un binding dont le chemin (`Path`) est totalement connu et ne dépend pas du `DataContext`. Par exemple ci-dessous, on affiche le mois courant (celui de la date du jour) dans un `TextBlock`.

```
<Window x:Class="CH5_BINDING1.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:sys="clr-namespace:System;assembly=mscorlib"
        xmlns:local="clr-namespace:CH5_BINDING1"
        mc:Ignorable="d"
        Title="CH5_BINDING1 (Source)" Height="200" Width="200">

        <StackPanel Margin="10" Background="White">

            <TextBlock Text="Source :" />
            <TextBlock Margin="5"
            Text="{Binding Source={x:Static sys:DateTime.Today},
Path=Month}" />

    </StackPanel>
</Window>
```

On exécute ce programme au mois de février (deuxième mois de l'année) ce qui donne l'affichage suivant :



1.1.2 Propriété RelativeSource

Cette propriété est relative contrairement à la propriété précédente, c'est-à-dire qu'elle envisage comme contexte l'arbre visuel défini par le code XAML dans lequel le contrôle courant s'insère.

Dans cet arbre visuel, on peut rechercher des informations sur le contrôle courant (grâce au mot-clé `Self`) ou sur un ancêtre, une balise située plus haut dans l'arbre en quelque sorte, ceci grâce au mot-clé `FindAncestor`. Une dernière utilisation possible est de recourir au mot-clé `TemplatedParent` qui permet d'appliquer un binding aux éléments soumis à un template donné, par exemple définis *via* une balise `DataTemplate`.

L'exemple suivant reprend chacune des valeurs possibles de `RelativeSource` :

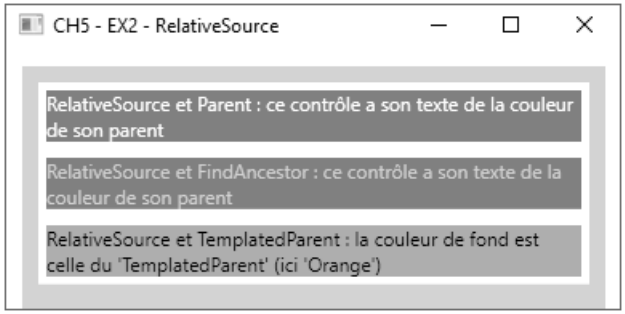
- `Self` : une première écriture voit sa couleur de texte identique à la couleur de fond de son parent.
- `FindAncestor` : une seconde écriture voit sa couleur de texte identique à la couleur de fond d'un ancêtre d'un type donné (`StackPanel`).
- `TemplatedParent` : une troisième écriture voit sa couleur de fond issue du contrôle qui accueille son *Template*.

Voici le code :

```
<Window x:Class="CH5_EX2_RELATIVESOURCE.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:CH5_EX2_RELATIVESOURCE"
        mc:Ignorable="d"
        Title="CH5 - EX2 - RelativeSource" Height="200" Width="400">
    <StackPanel Margin="10,10,10,0"  Background="LightGray" Height="249"
VerticalAlignment="Top">

        <StackPanel Margin="10" Background="White">

            <TextBlock Margin="5"
            Background="Gray"
            Text="RelativeSource et Parent :
ce contrôle a son texte de la couleur de son parent"
            TextWrapping="Wrap"
            Foreground="{Binding RelativeSource={RelativeSource Self},
Path=Parent.Background}"/>

            <TextBlock Margin="5"
            Background="Gray"
            Text="RelativeSource et FindAncestor :
ce contrôle a son texte de la couleur de son parent"
            TextWrapping="Wrap"
            Foreground="{Binding RelativeSource={RelativeSource
FindAncestor, AncestorType={x:Type StackPanel}, AncestorLevel=2},
Path=Background}"/>

            <ItemsControl Background="Orange" Margin="5">
                <TextBlock TextWrapping="Wrap">RelativeSource et
TemplatedParent : la couleur de fond est celle du 'TemplatedParent'
(ici 'Orange')</TextBlock>
                <ItemsControl.ItemTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <TextBlock Background="{Binding
RelativeSource={RelativeSource TemplatedParent}, Path=Background}"/>
                        </StackPanel>
                    </DataTemplate>
                </ItemsControl.ItemTemplate>
            </ItemsControl>
```

```
        </StackPanel>
    </StackPanel>

</Window>
```

Cela donne la fenêtre suivante :



Remarque

Lors de l'utilisation de `FindAncestor`, il est possible de spécifier le niveau d'ascendance auquel se trouve le contrôle « ancêtre » grâce à la propriété `AncestorLevel`.

1.1.3 Propriété ElementName

`ElementName` permet de référencer un élément de la vue par son nom. Ainsi, dans l'exemple d'école suivant, une première `TextBlock` affiche le nom d'une seconde `TextBlock` et réciproquement. Le référencement de chaque contrôle se fait via le nom de chacune d'elles.

```
<Window x:Class="CH5_EX3_ElementName.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:CH5_EX3_ElementName"
        mc:Ignorable="d"
        Title="CH5 - EX3 - ElementName" Height="350" Width="525">
    <StackPanel Margin="10">
        <TextBlock Margin="10" Name="Premier_TextBlock"
Text="{Binding ElementName=Second_TextBlock, Path=Name}" />
```