Chapitre 4
PowerShell et XAML

1. Introduction

Dans le deuxième chapitre, nous avons vu ce qu'était le cœur de notre interface, XAML.

Nous avons également vu comment créer et éditer un fichier XAML.

Dans ce chapitre, nous allons voir comment lier notre interface avec notre code PowerShell.

Nous allons également voir comment ajouter des actions sur nos Controls afin de créer une interactivité avec l'utilisateur.

Il est important de bien comprendre le rôle de chaque partie dans son projet WPF et PowerShell.

Ce projet est composé de deux parties :

– PowerShell

– XAML

2. Rôles des fichiers

2.1 XAML pour notre interface

La partie XAML, comme mentionné au chapitre XAML, le cœur de notre interface, représentera la partie graphique, et uniquement graphique, de notre projet.

Nous intégrerons dans cette partie tous nos Controls et aspects graphiques.

2.2 PS1 pour notre code

La partie PowerShell quant à elle représentera la partie code de notre projet. C'est la partie qui permettra de gérer toutes les actions sur les Controls liées à notre interface.

2.3 Comment organiser son projet ?

Les deux parties XAML et PS1 peuvent s'utiliser de deux façons :

– Deux fichiers distincts (méthode que je recommande pour une meilleure lisibilité et davantage de maniabilité).
– Tout inclure dans le fichier PowerShell.

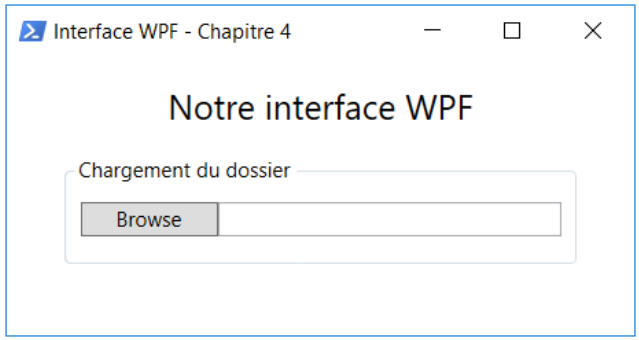
Ces deux méthodes seront traitées dans ce chapitre.

© Editions ENI - All rights reserved

2.4 Notre projet dans ce chapitre

Le projet qui servira d'exemple dans ce chapitre aura pour nom **MonProjet**.

L'interface sera telle que ci-dessous :



Cette interface est représentée par le code XAML ci-dessous :

```
<Window
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="Interface WPF - Chapitre 4" Width="380" Height="200">
<Grid>
<StackPanel Orientation="Vertical" HorizontalAlignment="Center"
Margin="0,10,0,0">
<Label Content="Notre interface WPF" FontSize="20"
HorizontalAlignment="Center"/>
<GroupBox Header="Chargement du dossier" Margin="0,10,0,0"
Height="65" Width="300">
<StackPanel Orientation="Horizontal" Margin="0,0,0,0"
HorizontalAlignment="Center">
<Button Name="MonBouton" Content="Browse" Width="80" Height="20"/>
<TextBox Name="MonTextBox" Width="200" Height="20"/>
</StackPanel>
</GroupBox>
</StackPanel>
</Grid>
</Window>
```

3. Première méthode : deux fichiers distincts

Notre projet sera composé de deux fichiers tels que ci-dessous :

– MonProjet.xaml : notre interface
– MonProjet.ps1 : notre code

3.1 Lier PowerShell et XAML

Ouvrons donc le fichier PowerShell et ajoutons-y le code suivant afin de charger notre fichier **MonProjet.xaml** :

```
[xml]$MonXAML = get-content -path "MonProjet.xaml"
$Reader=(New-Object System.Xml.XmlNodeReader $MonXAML)
$Interface = [Windows.Markup.XamlReader]::Load($Reader)
$Interface.ShowDialog() | Out-Null
```

Que fait ce code ?

– Nous allons d'abord transformer notre XAML en un objet XML en utilisant **`System.Xml.XmlNodeReader`**.
– Ensuite, nous chargeons ce dernier en utilisant **`[Windows.Markup.XamlReader]::Load`**.
– Enfin, nous affichons notre interface en utilisant **`$Interface.ShowDialog()`**.

© Editions ENI - All rights reserved

Si nous lançons le fichier PowerShell, nous obtenons l'erreur suivante :

```
Unable to find type [Windows.Markup.XamlReader].
At C:\Users\damien.van robaeys\Documents\Livre\CE04PS1WPF\Exemples\04EP01 - Deux fichiers\MonProjet.ps1:5 char:14
+ $Interface = [Windows.Markup.XamlReader]::Load($Reader)
+              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (Windows.Markup.XamlReader:TypeName) [], RuntimeException
    + FullyQualifiedErrorId : TypeNotFound

You cannot call a method on a null-valued expression.
At C:\Users\damien.van robaeys\Documents\Livre\CE04PS1WPF\Exemples\04EP01 - Deux fichiers\MonProjet.ps1:7 char:1
+ $MonBouton = $Interface.FindName("MonBouton")
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : InvokeMethodOnNull

You cannot call a method on a null-valued expression.
At C:\Users\damien.van robaeys\Documents\Livre\CE04PS1WPF\Exemples\04EP01 - Deux fichiers\MonProjet.ps1:8 char:1
+ $MonTextBox = $Interface.FindName("MonTextBox")
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : InvokeMethodOnNull
```

Pourquoi ?

La raison est simple, il nous manque l'assembly permettant le fonctionnement de WPF.

Présentation des assemblies

L'assembly à utiliser ici est l'assembly ***presentationframework*** et se présente sous la forme d'un fichier .dll.

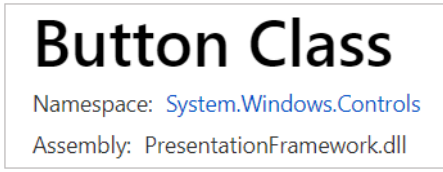
C'est cette assembly qui contiendra tous les Controls que nous voulons utiliser.

C'est également par elle que notre interface WPF pourra s'afficher.

Dans le chapitre XAML, le cœur de notre interface, nous avons retrouvé une référence à cette assembly à deux reprises.

En effet, lors de la création d'un projet WPF via Visual Studio, en cliquant sur [F12] sur un Control, par exemple un TextBox, nous obtenons l'information ci-dessous :

Si vous cherchez des informations sur un Control sur le site de Microsoft, par exemple un Button, vous obtiendrez l'information ci-dessous.



Remarque

Vous trouverez également une explication sur ces assemblies au chapitre Interface graphique et PowerShell.

Ce qui nous amène donc à notre seconde étape qui en réalité est donc la première.

3.2 Chargement de l'assembly

3.2.1 Comment charger l'assembly ?

Pour charger l'assembly **presentationframework**, nous utiliserons la ligne ci-dessous :

```
[System.Reflection.Assembly]::LoadWithPartialName
('presentationframework')
```

En exécutant cette ligne dans une console PowerShell, nous obtenons le résultat suivant :

```
GAC    Version        Location
---    -------        --------
True   v4.0.30319     C:\windows\Microsoft.Net\assembly\GAC_MSIL\presentationframework\v4.0_4.0.0.0__31bf3856ad364e35\presentationframework.dll
```

Nous pouvons remarquer que le chemin de cette assembly est local.

En parcourant ce dossier, nous retrouvons également d'autres assemblies.

© Editions ENI - All rights reserved