Exercice c# - WPF

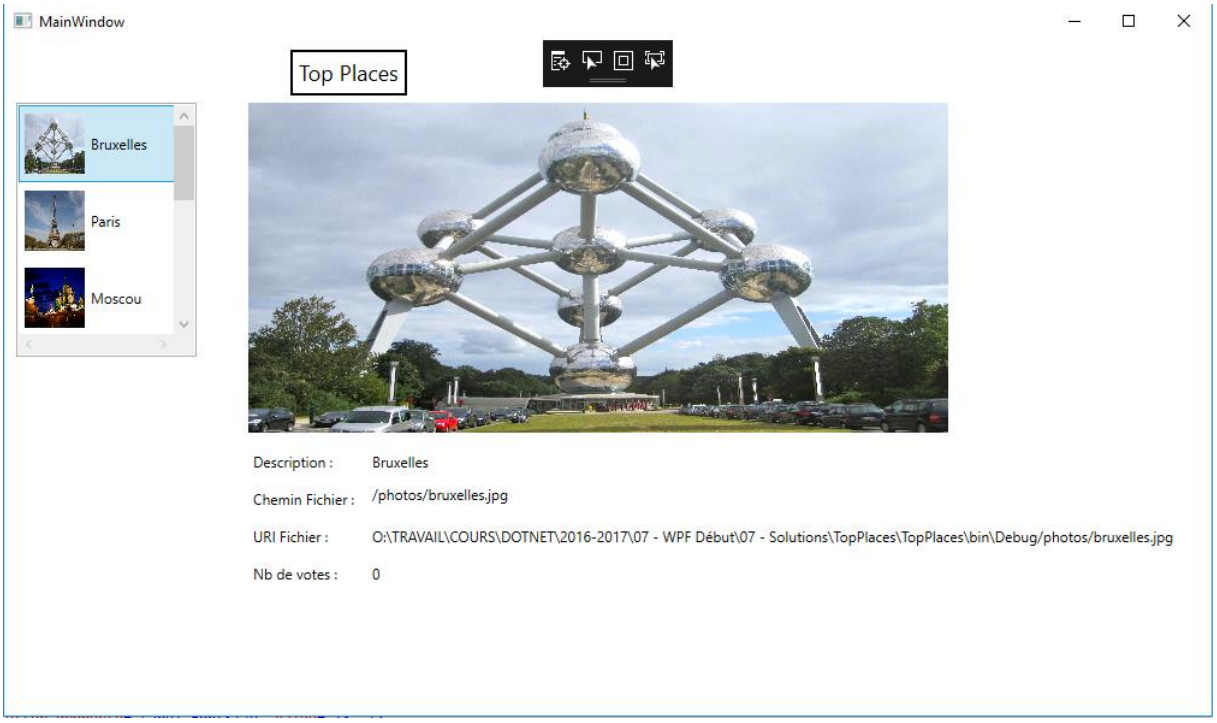
Objectifs

- ✓ Découvrir WPF
- ✓ Découvrir le XAML (Ressources, Styles, Converters)
- ✓ Découvrir le Binding

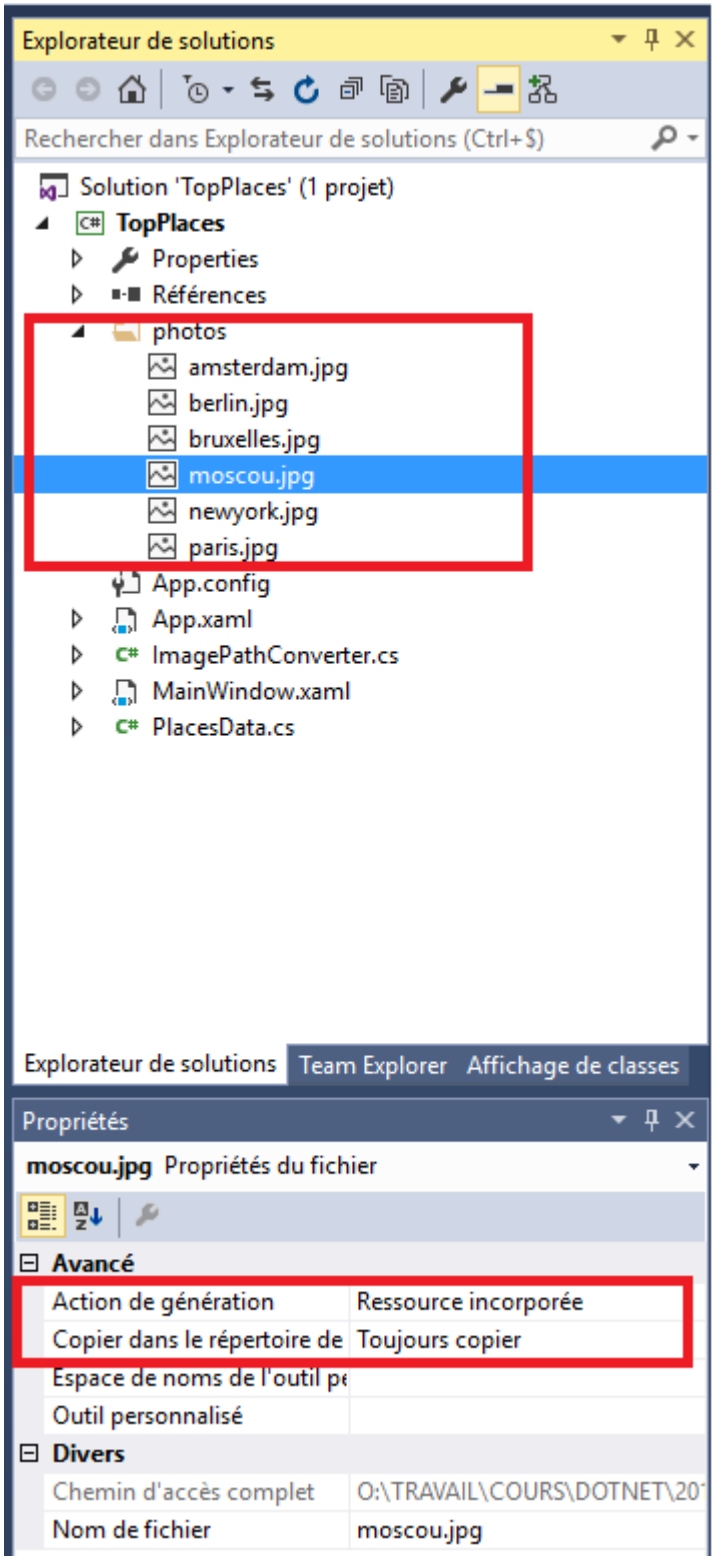
Exercice 1

Pour ce premier exercice, nous allons créer une application « TopPlaces » qui est une application permettant d'afficher et de voter pour son endroit préféré. Il s'agit d'une application très simple pour vous exposer les rudiments de WPF. De plus pour ce premier exercice, vous serez extrêmement guidé, le but étant que vous fassiez vos armes sur ce premier exercice avant de passer au second exercice, qui lui sera sans guidage !

Voici un aperçu de l'application à réaliser :



1. Créez un nouveau projet « Application WPF » que vous nommerez « TopPlaces ».
2. Récupérez sur Moodle le fichier « images.zip » et « photos.zip » que vous décompresserez et placerez dans votre Projet dans un répertoire nommé « photos ». Configurez Visual Studio pour qu'il recopie toujours ces fichiers dans le répertoire de génération (bin/debug ou bin/release). Voir ci-dessous.



3. Ajoutez une « listbox » à gauche de l'écran que vous nommerez « `listBoxPhotos` ».
    a. Pour donner un nom à un objet (nom qui pourra être utilisé dans le code) -> allez dans le XAML et ajoutez x :Name= «nom»
    b. Exemple : `<ListBox x:Name="listBoxPhotos" …..`

4. Ajoutez un champ image que vous nommerez « image1 ». Ce champ correspond à l'image de l'Atomium sur l'aperçu.
    a. Positionnez la propriété « Stretch » de ce champ à « Fill » pour que l'image source s'adapte à l'espace de votre champ.
5. Ajoutez un label de titre qui s'intitule « Top Places » au dessus du champ image.
    a. C'est la propriété « Content » qui permet de définir un texte
6. Définissez un style dans la partie Window.Resources du XAML comme ceci :

```
<Style x:Key="TitreStyle">
          <Setter Property="Label.FontSize" Value="18" />
          <Setter Property="Label.BorderBrush" Value="Black"/>
          <Setter Property="Label.BorderThickness" Value="2"/>
</Style>
```

    a. Remarquez le x:Key qui permet de donner une clé à ce style pour pouvoir l'utiliser par la suite !
7. Utilisez le style « TitreStyle » défini ci-dessus sur le label de titre
    a. `<Label x:Name="Titre" Content="Top Places" Style="{StaticResource TitreStyle} …."`
    b. Remarquez l'utilisation des accolades, du StaticRessource et de la clé « TitreStyle »
8. Lancez votre projet et vérifiez que le style est bien appliqué !
9. Créez, positionnez et nommez 4 Labels(Description, Chemin fichier, URI fichier, nb votes) en dessous de votre champ image.
10. Définissez/Utilisez un style dans le XAML pour que la police « courier new » soit appliquée aux 4 labels définis.
11. Créez, positionnez et nommez 4 Labels(DescriptionValeur, CheminfichierValeur, URIfichierValeur, nbVotesValeur) en regard de vos 4 labels précédents.
12. Créer une classe « PlacesData » que nous compléterons par la suite.
13. Créer une nouvelle classe « Places » dans le fichier « PlaceData.cs » qui contiendra les champs suivants :
    a. _description : string
    b. _pathImageFile : string
    c. _nbVotes : int
    d. _uri : Uri
    e. _image : BitmapFrame
14. Définissez un constructeur comme ceci :

```
public Place(string path,string description)
{
     _description = description;
     _pathImageFile = path;
     _nbVotes = 0;
     _uri = new Uri(_pathImageFile);
     _image = BitmapFrame.Create(_uri);
}
```

15. Définissez des propriétés (get/set) pour chacun des champs de la classe « Places »
16. Dans la classe « PlacesData » ajoutez une liste de « Places »
    a. `private IList<Place> placesList;`

17. Définissez un constructeur « PlacesData » comme ceci :

```
string pathProject = Environment.CurrentDirectory;
Place p1 = new Place(pathProject +"/photos/bruxelles.jpg", "Bruxelles");
Place p2 = new Place(pathProject + "/photos/paris.jpg", "Paris");
Place p3 = new Place(pathProject + "/photos/moscou.jpg", "Moscou");
Place p4 = new Place(pathProject + "/photos/amsterdam.jpg", "Amsterdam");
Place p5 = new Place(pathProject + "/photos/newyork.jpg", "New York");

placesList = new List<Place>();
placesList.Add(p1);
placesList.Add(p2);
placesList.Add(p3);
placesList.Add(p4);
placesList.Add(p5);
```

    a. Remarquez comment on peut récupérer le chemin où se trouve le projet.

18. Définissez une propriété (get) pour le champ « placeList ».
19. Dans le fichier « MainWindow.xaml.cs », après le « InitializeComponent » ajoutez ceci :

```
PlacesData placesData = new PlacesData();
this.listBoxPhotos.DataContext = placesData.PlacesList;
```

    a. Notre listbox a maintenant un Datacontext (des données) qui nous permettra de faire du binding !

20. Modifiez ou ajoutez dans le XAML la propriété ItemsSource de la listbox comme ceci
    a. `ItemsSource="{Binding}"`
21. Lancez votre projet, vous devriez voir dans votre listbox 5 lignes « TopPlaces.Place »
    a. La listbox affiche ici les objets Places que nous avons défini
22. Modifions maintenant l'affichage de ces objets dans la liste en définissant un template (dans Window.Resources) pour les élements de la listbox :

```
<DataTemplate x:Key="ItemTemplate">
<StackPanel  Orientation="Horizontal" Margin="0 5 0 5">
   <Image Width="50" Height="50" Stretch="Fill" Source="{Binding Image}"
   VerticalAlignment="Center" HorizontalAlignment="Center"/>
   <Label Content="{Binding Description}" VerticalAlignment="Center"
   HorizontalAlignment="Center"/>
</StackPanel>
</DataTemplate>
```

    a. Décortiquons ce template : nous indiquons que chaque élément d'une listbox sera un stackpanel constitué d'une image et d'un texte et nous « bindons » ces éléments avec les propriétés définies dans la classe « Place » **Attention ceci doit correspondre !** La propriété « Image » par ex. est définie dans la classe « Place et renvoie une image BitmapFrame ! **Vérifiez chez vous la cohérence !**

23. Utilisons maintenant le « template »
    a. `<Listbox … ItemTemplate="{StaticResource ItemTemplate} …."`

24. Lancez votre projet, vous devriez avoir une liste de photos avec les descriptions à côté.
25. Occupons-nous maintenant d'afficher l'image d'un endroit lorsque l'on clique dessus dans la listbox. Pour ce faire, double-cliquez sur la listbox et ajouter ce code :

```
Place place = (Place)listBoxPhotos.SelectedItem;
BitmapSource photo = BitmapFrame.Create(new Uri(place.Path));
Image1.Source = photo;
```

a. Vous êtes normalement déjà familier avec le système d'événement vu lors de la séance consacré à Winforms

26. Testez votre projet (affichage de l'image sélectionnée lors d'un clic sur la listbox)
27. Affichons maintenant dans le champ « DescriptionValeur » la description également lors d'un clic sur une image de la listbox. Nous allons faire ceci avec du « binding » en liant directement le label « DescriptionValeur » à la valeur sélectionnée dans listbox
    a. `<Label x:Name="DescriptionValeur" Content="{Binding ElementName=listBoxPhotos,Path=SelectedItem.Description}" …`
    b. Remarquez que SelectedItem représente un objet Place et que l'on accède à la description via une propriété "Description" définie dans la classe "Place"
28. Faites pareil pour « cheminFichierValeur, URIFichierValeur, et nbVotesValeur ».
29. Testez votre projet
30. Nous avons utiliser le « binding » jusque ici sans modifier la valeur reçue par la listbox. Cependant nous avons la possibilité en WPF d'appliquer un traitement pour transformer cette valeur. C'est ce que WPF appelle un converter. Nous allons utiliser un « converter » pour le champ « cheminfichiervaleur » , nous allons afficher le chemin à partir du répertoire « photos ».
31. Créer une classe « ImagePathConverter » comme ceci :

```
public class ImagePathConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter,
    System.Globalization.CultureInfo culture)
    {
              if (value == null) return "rien";
              // index début
              int indexDebDirPhotos = value.ToString().IndexOf("/photos");
              return value.ToString().Substring(indexDebDirPhotos);        }

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

    a. Remarquez que nous implémentons l'interface IvalueConverter
    b. Remarquez qu'un converter peut effectuer des conversions dans les 2 sens (source <-> cible)
32. Utilisons le « converter » sur le label « cheminfichierValeur ».
    a. Dans le fichier XAML, sous Window ajoutez
        i. `xmlns:local="clr-namespace:TopPlaces">`
        ii. Nous indiquons au XAML où trouver le code de notre converter
    b. Dans le fichier XAML, sous Window.Resources ajoutez
        i. `<local:ImagePathConverter x:Key="imgPathConverter"/>`
        ii. Nous définissons notre converter comme une ressource
    c. Modifiez le label "cheminFichierValeur"
        i. `Content="{Binding ElementName=listBoxPhotos ,Path=SelectedItem.Path,UpdateSourceTrigger=PropertyChanged, Converter={StaticResource imgPathConverter}}"`
        ii. Remarquez le UpdateSourceTrigger et l'application du converter

33. Finalement ajoutons un bouton personnalisé comme ceci :

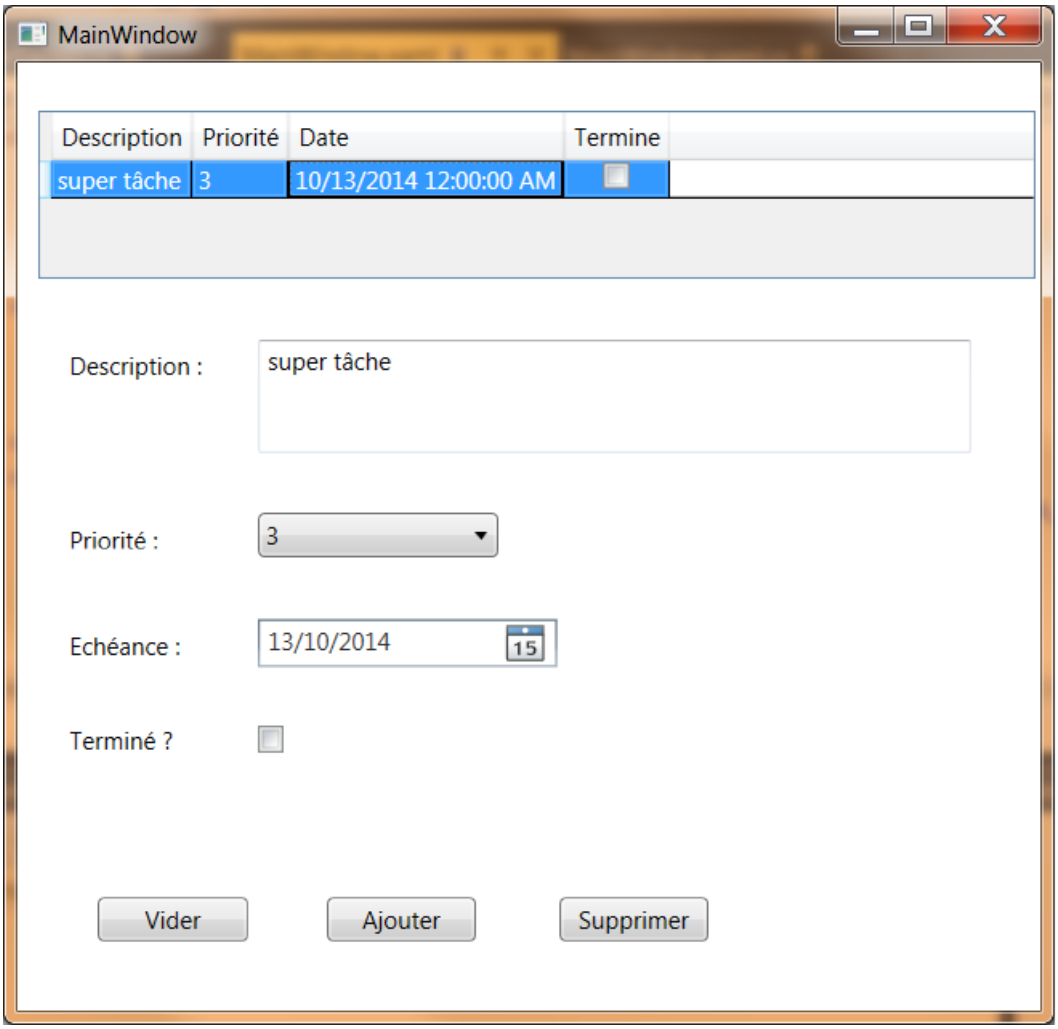
```
<Button Height="30" Margin="378,484,438,58" Click="Button_Click">
   <StackPanel Orientation="Horizontal" Width="113">
   <Image Source="\images\jaime.png" Height="25" Width="45" Margin="0,0,0,0"
      HorizontalAlignment="Left"/>
   <TextBlock Text="Je vote pour" VerticalAlignment="Center" Width="74"/>
   </StackPanel>
</Button>
```

   a. Comme pour la listbox, nous avons défini un panel à l'intérieur d'un bouton permettant de personaliser l'aspect du bouton. Le bouton sera ici composé d'une image fixe et d'un bloc de texte.
   b. L'image « jaime.png » est à placer dans un répertoire «images » à la racine du projet.
   c. Remarquez qu'ici (dans le XAML) vous faites référence directement à la racine du projet.

34. Réalisez le code permettant d'augmenter le nombre de votes pour une « Place ».

Exercice 2

Ecrivez une application WPF de gestion de tâches. Cette application permettra l'ajout, suppression et modifications de tâches présentes dans une lsite en mémoire. Voici un aperçu de l'application à réaliser :



Indices de programmation

1. Utilisez un datagrid pour afficher la liste des tâches
    a. Ce composant a des similarités avec la listbox utilisée dans l'exercice 1
2. N'hésitez pas à consulter Internet !
3. Utilisez le binding !
4. Pour actualiser le datagrid (refresh) -> vous pouvez utiliser « datagrid.items.refresh() » après avoir ajouté une tâche par exemple.