



Cégep de Saint-Hyacinthe
Département d'informatique

Programmation orientée objet
420-2DP-HY
(3-3-3)

Introduction à WPF #1 - Base

(Version 1.3)

1.5 à 3 heures

Préparé par
Martin Lalancette

Comprendre les éléments suivants:

- Qu'est-ce que WPF?
- Qu'est-ce que XAML?
- Normes
- Disposition des contrôles et des conteneurs
- Les images

Table des matières

Introduction.....	3
Qu'est-ce que le WPF?	3
Que veut dire XAML?.....	4
Quelques propriétés et événements de base (Contrôles WPF)	5
Quelques événements en WPF associés au formulaire	6
Normes pour l'attribution de nom pour les composantes visuelles	7
Nos premiers programmes de type WPF	8
Comment créer un projet.....	8
Interface associée.....	9
Structurer un projet WPF selon MVVM	9
MainWindow.xaml et MainWindow.xaml.cs	10
App.xaml et App.xaml.cs	10
L'objet MessageBox.....	11
Les conteneurs	13
Grid	14
Gestion des images.....	15
Ajouter des images à un projet WPF	15
Les propriétés appropriées.....	16
Changer l'image d'un objet <i>Image</i> par programmation	16
Classe Uri (Uniform Resource Identifier).....	17
Classe BitmapImage	17
Changer l'image d'un objet Button, Ellipse ou Label par programmation.....	17
Classe Brush.....	17
Autres conteneurs (à découvrir au besoin - FACULTATIF)	Erreur ! Signet non défini.
Bibliographie.....	19

Introduction

Cette séquence a pour but de vous initier aux notions de base nécessaires à l'utilisation d'un projet WPF afin de concevoir une application. Nous commencerons par énoncer les éléments théoriques appuyés d'exemples simples et faciles à reproduire. Afin d'axer l'attention sur la compréhension de ces notions, il y aura des exercices à faire tout au long de cette séquence. **La présente séquence va traiter de la création de projet (programme) de type formulaire WPF.** Pour bien suivre les instructions, une préparation de base s'impose. Il est important de créer un répertoire de travail (sur votre C: ou clé USB). Voici une suggestion d'arborescence:



Préparation : S'assurer d'avoir créé l'arborescence ici haut mentionnée sur votre C ou votre clé USB.

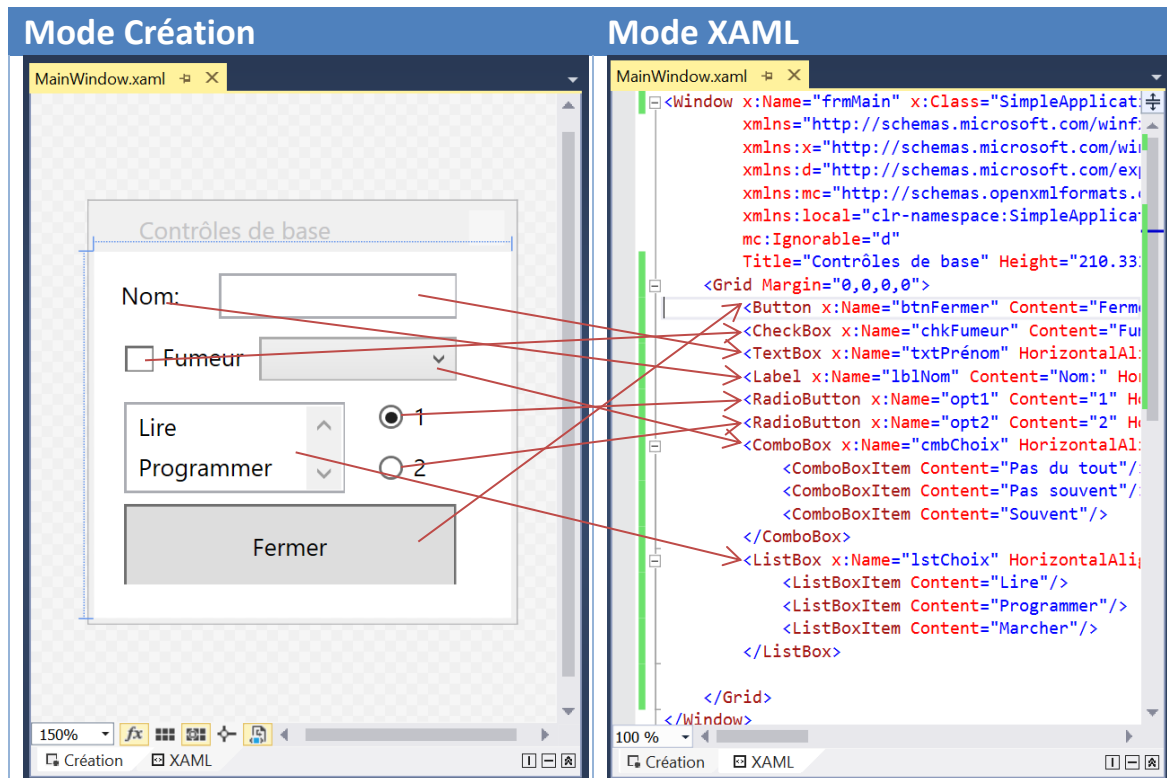
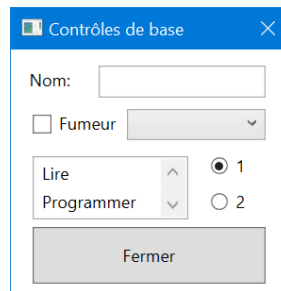
Qu'est-ce que le WPF?

WPF est l'acronyme anglais « Windows Presentation Foundation ». C'est un « système moderne d'affichage graphique » (MacDonald, 2012) destiné au système d'exploitation Windows. Caractéristiques :

- Est apparu avec la version Vista de Windows (.Net framework 3.0).
- Son but principal : **Offrir des interfaces modernes et fluides.**
- Utilise DirectX : ceci permet d'utiliser directement le processeur de la carte graphique. Ce qui libère le CPU (Il peut se concentrer sur les autres algorithmes). Il est conçu pour détecter les cartes graphiques et les utiliser à leur plein potentiel. Exemple : l'accélération et 3D.
- Facilite le travail collaboratif en distinguant l'interface (les designers !) et la logique de programmation (les programmeurs !).
- Permet les graphiques vectoriels, les animations et s'adapte à la résolution de l'écran.

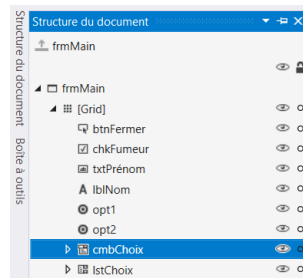
Que veut dire XAML?

L'acronyme XAML est « **Extensible Application Markup Language** ». Ce langage est principalement utilisé pour permettre le design des [interfaces utilisateurs en WPF](#). Il conserve les informations relatives aux contrôles qui sont déposés dans la fenêtre (*window*). Exemple :



De plus en WPF, le XAML permet de coder des animations que nous verrons un peu plus tard dans la session (si le temps le permet). Comme le HTML, XAML peut se coder sans éditeur spécial. Toutefois, il devient ardu de connaître chaque balise et ses attributs par cœur. C'est pourquoi Visual Studio

offre un mode conceptuel pour aider. Pour voir la structure d'un document : **Affichage/Autres fenêtres/Structure de document**. Exemple :



Quelques propriétés et événements de base (Contrôles WPF)

Voici quelques propriétés et événements de base associés aux contrôles visuels à connaître :

Contrôle	Propriétés	Événements
Commun	Content	IsEnabledChanged
	Nom	Click (si présent) ou MouseLeftButtonDown
	Visibility	
	IsEnabled	
	Texte	
	Background	
	Foreground	
	Title	
TextBox	IsReadOnly	TextChanged
	MaxLines	
	MinLines	
	MaxLength	
Checkbox	IsChecked	Checked Unchecked
RadioButton	IsChecked	Checked Unchecked
	IsThreeState	
Button	Content (Short key = _ devant la lettre)	Click
Combobox et ListBox	Items (type ComboBoxItem)	SelectionChanged
	SelectedIndex	
	SelectedItem	
	SelectedValue	

Quelques événements en WPF associés au formulaire

En WPF, il existe une multitude d'événements que l'on peut s'y rattacher pour exécuter du code. Voici les événements les plus communs associés à la gestion d'un formulaire :

Événement	Description
Loaded	Est déclenché après que la fenêtre soit entièrement chargée en mémoire. C'est un bon endroit pour initialiser d'autres objets. <code>IsLoaded</code> devient à <i>true</i> .
Unloaded	Est déclenché lorsqu'un contrôle est relâché. Peut arriver lorsqu'une fenêtre (<i>Close</i>) est fermée et retirée.
Activated	Est déclenché lorsque la fenêtre obtient le focus (soit en faisant ALT-TAB ou en cliquant sur la fenêtre avec la souris).
Deactivated	Est déclenché lorsque l'utilisateur quitte la fenêtre active.
Closing	Est déclenché lorsque l'utilisateur demande de fermer la fenêtre (cliquer sur le X, <code>Windows.Close()</code> ou <code>Application.Shutdown()</code>). Cet événement vous permet de <u>confirmer la fermeture ou non de la fenêtre</u> . Vous pouvez annuler en définissant la propriété <code>CancelEventArgs.Cancel</code> à <i>true</i> .
Closed	Est déclenché lorsque la fenêtre est fermée. Les contrôles sont toujours disponibles et accessibles. C'est l'endroit privilégié pour effectuer libérer adéquatement vos objets. Exemple : Enregistrer des données dans une table ou un fichier.

Normes pour l'attribution de nom pour les composantes visuelles

Chaque objet visuel doit posséder un nom unique (via la propriété ***name***) en respectant les normes départementales suivantes :

Nom français (liste partielle)	Nom anglais	Préfixe
Bouton	Button	btn
Bouton de la barre de boutons	ToolBarButton	btn
Barre de boutons	ToolBar	bar
Cases à cocher	CheckBox	chk
Boîte de liste avec case à cocher	CheckedListBox	lst
Boîte Combo ou Liste déroulante	ComboBox	cmb
Boîtes de groupes ou les cadres	GroupBox	grp
Étiquette	Label	lbl
Boîte de liste	ListBox	lst
Images	Image	img
Icônes	Icon	icn
Cadre déroulant dépourvu d'étiquette	Panel	pnl
Boutons radio ou Cases d'option	RadioButton	btn ou opt
Boîte de saisie	TextBox	txt
Fiche ou Feuille	Form ou window	frm
Menu	MainMenu	mnu
Item du menu	MenuItem	itm
Boîte de dialogue de sauvegarde	SaveFileDialog	sav
Boîte de dialogue d'ouverture	OpenFileDialog	Opn
Boîte de saisie de dates	DateTimePicker ou MonthCalendar	date

Selon l'objet choisi, il faut ajouter le préfixe suivi du nom à donner.
Exemples :

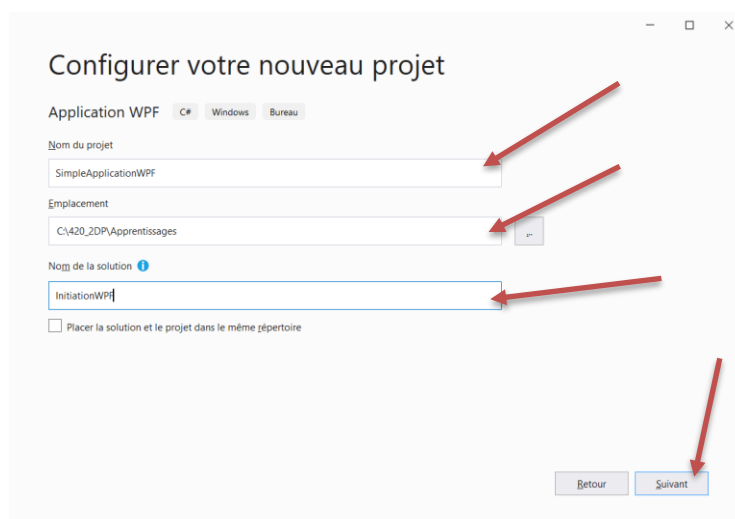
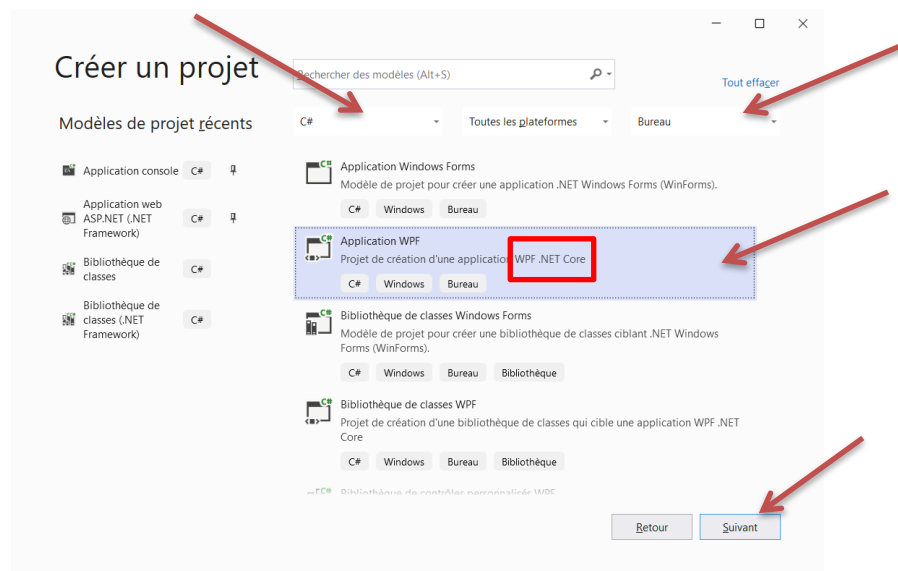
1. Un bouton OK → **btnOK**
2. Un formulaire pour les patients → **frmPatients**
3. Un menu Fichiers → **mnuFichiers**
4. Un sous-menu Ouvrir du menu Fichiers → **itmOuvrirFichier**
5. Une case à cocher Fumeur → **chkFumeur**

Nos premiers programmes de type WPF

Exercice 1. : Créer une solution intitulée **InitiationWPF** avec un projet de type WPF nommée **SimpleApplicationWPF** en suivant la procédure ici-bas. Cet exercice a pour but de vous initier aux composantes de base du WPF. Suivre les instructions de l'enseignant. Utiliser le dernier *framework .NET Core*. Modéliser selon MVVM.

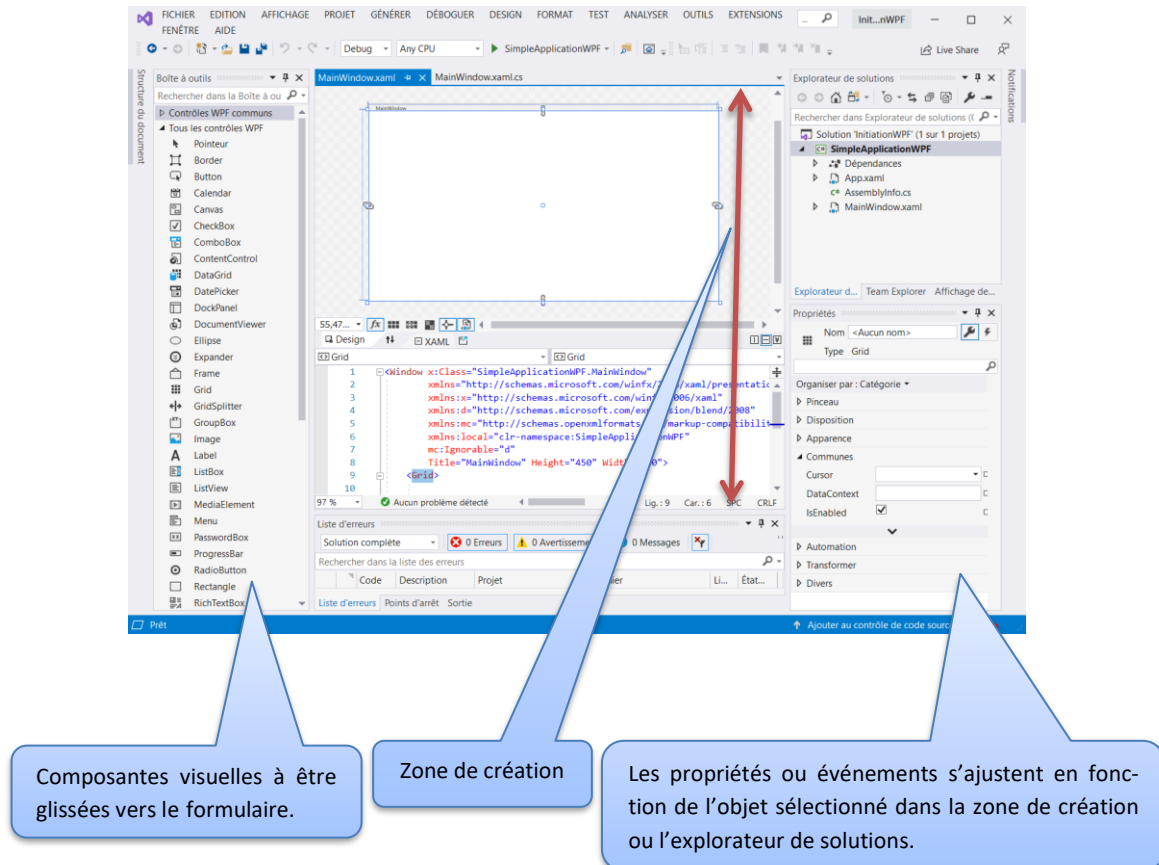
Comment créer un projet

Démarrer Visual Studio. Dans le menu **Fichier**, choisir **Nouveau...**



Interface associée

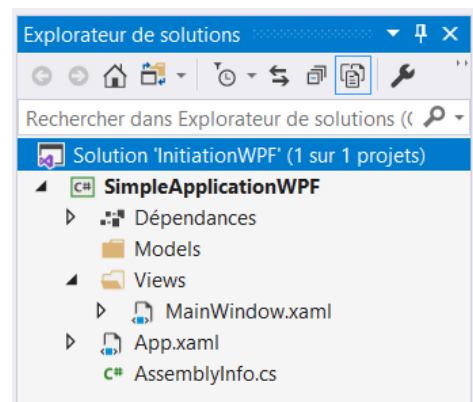
Lorsqu'un projet de type WPF est créé, nous avons accès à une gamme de composantes visuelles se trouvant dans **Boîte à outils**. Voici un exemple d'écran et ces différentes sections :



Structurer un projet WPF selon MVVM

Voici la procédure à suivre pour structurer votre nouveau projet sous MVVM :

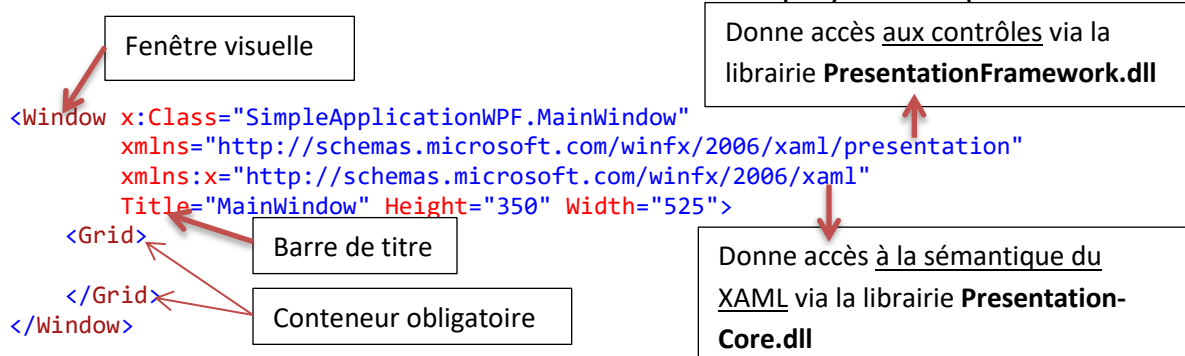
1. Ajouter le dossier **Models**,
2. Ajouter le dossier **Views**,
3. Déplacer le fichier **MainWindow.xaml** dans **Views**,
4. Ouvrir le fichier **App.xaml** et modifier la ligne suivante :
`StartupUri="MainWindow.xaml"` pour
`StartupUri="Views/MainWindow.xaml"`
5. Démarrer l'application pour vous assurer qu'il fonctionne.



En WPF, nous parlons de **fenêtres (windows)**. Voici les fichiers de base composant un projet WPF :

MainWindow.xaml et MainWindow.xaml.cs

Ces deux fichiers gèrent la fenêtre principale de l'application. Le fichier MainWindow.xaml contient à la base le code XAML ([namespace](#)) suivant et sert à conserver les informations sur les contrôles qui y sont déposés:



Le fichier MainWindow.xaml.cs va contenir le code C# associé aux événements de la fenêtre. Exemple :

```
namespace SimpleApplicationWPF
{
    /// <summary>
    /// Logique d'interaction pour MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void btnFermer_Click(object sender, RoutedEventArgs e)
        {
        }
    }
}
```

App.xaml et App.xaml.cs

Ces fichiers servent à contenir les ressources (ex. styles, animations, etc.) et les configurations de démarrage disponibles pour l'ensemble de l'application.

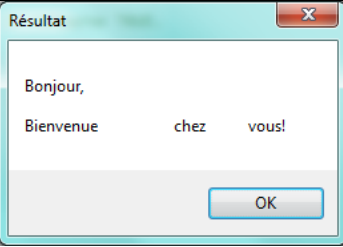
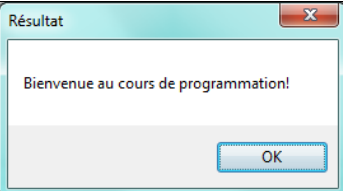
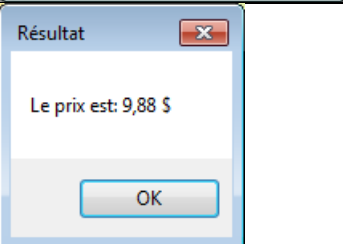
Exercice 2. : Dans le projet en cours, déposer les contrôles selon le modèle suivant :

Suivre les instructions de l'enseignant.

L'objet MessageBox

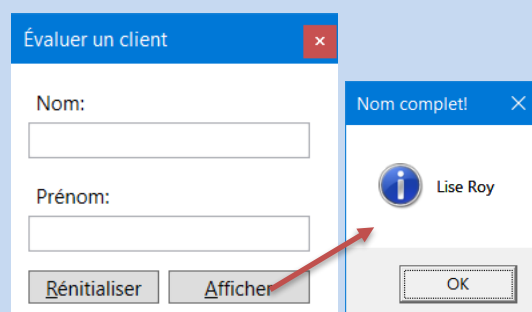
Voici maintenant un outil permettant d'afficher le résultat suite à un traitement. [MessageBox](#) est une classe qui permet d'afficher des messages afin d'informer l'utilisateur via une boîte de dialogue. Grâce à la méthode **Show(...)**, vous pouvez avoir accès à différents aspects de cette boîte de dialogue. Voici les possibilités de base :

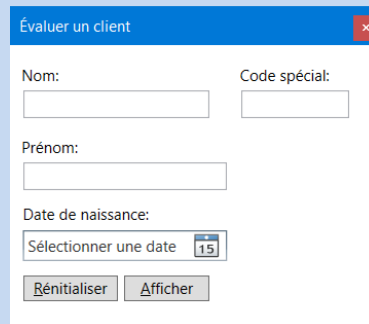
Exemple	Code C#	Visuel
Afficher « Bonjour! ».	<code>MessageBox.Show("Bonjour!");</code> Le premier paramètre correspond au message à afficher.	
Un titre et un message.	<code>MessageBox.Show("Votre nom est Marc Tremblay", "Information personnelle");</code> Ajouter un deuxième paramètre afin de spécifier le titre dans la barre de titre.	

Un message avec des changements de ligne et des tabulations.	<pre>string s1 = "Bon- jour,\n\nBienvenue\t\tchez\tvous!"; MessageBox.Show(s1, "Résultat");</pre>	
Un message avec concaténation.	<pre>string s1 = "Bienvenue au " + "cours de " + "programmation!"; MessageBox.Show(s1, "Résultat");</pre>	
Un message avec un nombre décimal à deux chiffres.	<pre>float fPrix = 9.8767f; MessageBox.Show("Le prix est: " + fPrix.ToString("0.00 \$"), "Résultat");</pre>	

Il existe d'autres possibilités offertes par cette classe que nous allons découvrir plus tard.

Exercice 3. Créer un nouveau projet WPF (.NET Core) nommé **Évaluer-Client** et ajouter les composantes suivantes sur le formulaire précédemment créé. Donner des noms significatifs et selon les normes aux objets de type TextBox et Button. Concernant le bouton Réinitialiser, ajouter le code qui permet de vider les deux TextBox lors d'un clic. Ajouter un raccourci sur la lettre R du bouton Réinitialiser. Exemple :



Exercice 4. Ajouter les composantes suivantes :

Ajouter un événement **TextChanged** aux objets **txtNom** et **txtPrénom**. Ajouter un événement **SelectedDateChanged** à l'objet **DatePicker**. Programmer l'algorithme suivant pour générer le code spécial **NNNPAAMMJJ** lorsque ces événements sont déclenchés.

- **NNN** : Trois premiers caractères du nom de famille (maj)
- **P** : Premier caractère du prénom (maj)
- **AA** : Deux derniers chiffres de l'année
- **MM** : Deux derniers chiffres du mois
- **JJ** : Deux derniers chiffres du jour

Les conteneurs

À la base, une fenêtre WPF ne peut contenir qu'un seul élément. Il faut utiliser des conteneurs afin de disposer les contrôles dans la fenêtre. Ce principe est tiré de la philosophie du Web. Avant de vous présenter ces conteneurs, voici quelques-unes des [meilleures pratiques](#) (Mosers, 2011) à tenir compte dans le design d'interface utilisateur en WPF :

1. Éviter les positions fixes. Utiliser les propriétés d'alignement et les marges.

2. Éviter les dimensions fixes. Définir *Width* et *Height* à Auto le plus possible.
3. Ne pas définir les dimensions et positions dans le code C#. Ceci rend difficile l'ajustement des conteneurs automatique.

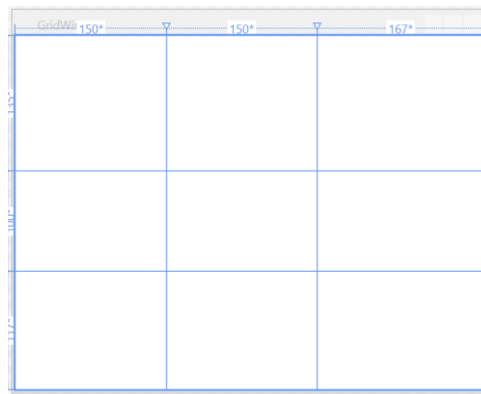
Voici un lien traitant de ce sujet :

1. <http://www.wpftutorial.net/LayoutProperties.html>

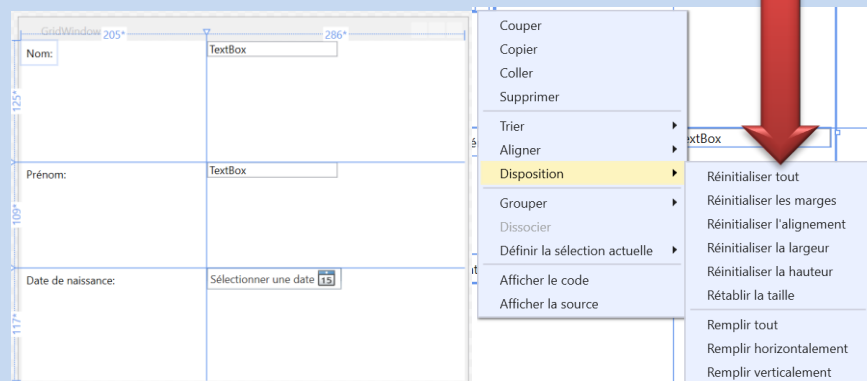
Voici les types de conteneurs possibles à utiliser.

Grid

Ce conteneur est le plus couramment utilisé. Il permet de disposer les contrôles en utilisant des lignes et des colonnes (sous forme d'un tableau).

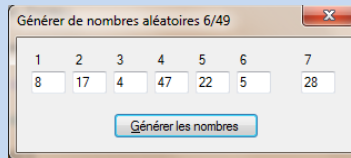


Exercice 5. : Dans le projet en cours, ajouter une fenêtre nommée **GridWindow.xaml**. Reproduire ceci:



Suivre les instructions de l'enseignant.

Exercice 6. (FACULTATIF) Concevoir un formulaire qui permet de générer aléatoirement des combinaisons pour la Loto 6/49. Il suffit de générer 7 nombres entre 1 et 49. Le 7^e correspond au nombre complémentaire. Exemple :



Gestion des images¹

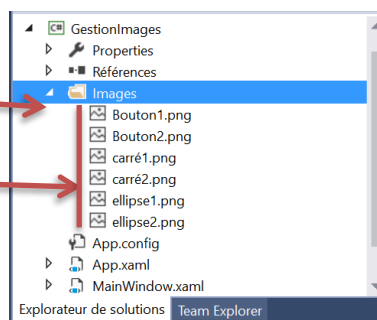
Voici une section qui résume l'intégration et l'utilisation des images dans un projet WPF.

Exercice 7. : Dans la solution en cours, créer le projet de type WPF nommé « **GestionImages** ». Structurer selon MVVM. Ce projet va servir aux explications de la prochaine section.

Ajouter des images à un projet WPF

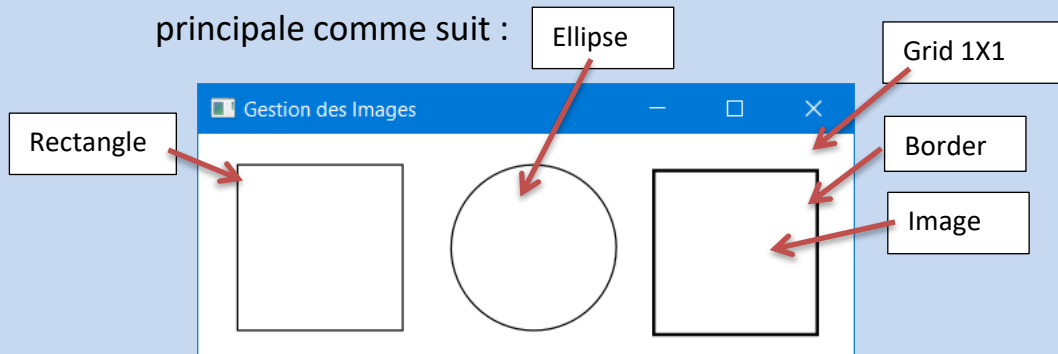
Voici la meilleure façon (ou pratique) d'intégrer des images dans un projet de type WPF. **Note** : **NE PAS UTILISER DE FICHER Resources.resx (dans Properties) pour la gestion des images et des sons. WPF ne le supporte pas.** Voici la procédure pour ajouter :

1. Créer un dossier nommé **Images**.
2. Ajouter les images dans ce dossier.
3. Pour chaque image définir **Resource** (Action de génération) dans **Propriétés**.



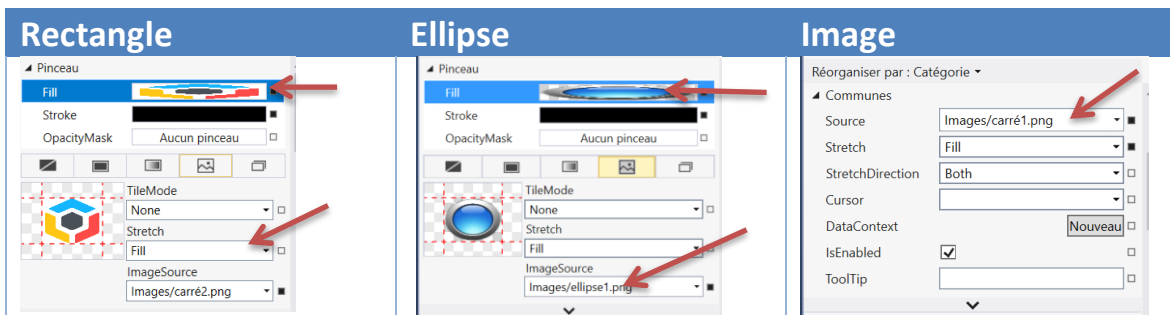
¹ Tiré du site : [https://msdn.microsoft.com/fr-fr/library/ms748873\(v=vs.110\).aspx](https://msdn.microsoft.com/fr-fr/library/ms748873(v=vs.110).aspx)

Exercice 8. : Extraire les images du fichier .ZIP fournit avec cette séquence et les déposer dans le dossier Images. Modifier la fenêtre principale comme suit :

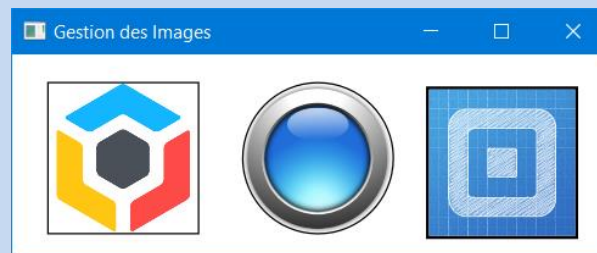


Les propriétés appropriées

Voici les propriétés à utiliser pour associer une image aux objets *Image*, *Button* et *Ellipse*.



Exercice 9. : Associer les images aux contrôles se trouvant dans la fenêtre comme suit :



Changer l'image d'un objet *Image* par programmation

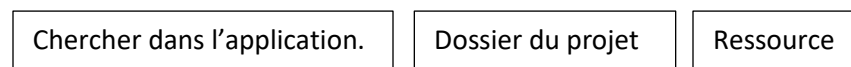
Pour changer d'image concernant cet objet, il faut utiliser la combinaison de deux autres objets.

Classe Uri (Uniform Resource Identifier)

Un [URI](#) est une représentation compacte d'une ressource disponible pour votre application sur un intranet ou sur Internet. En utilisant son constructeur, il crée une instance d'Uri à partir d'une chaîne de l'URI. Il analyse l'URI, le met dans un format réglementaire et procède aux encodages d'échappement requis. Exemple :

```
Uri maValeur = new Uri("http://www.contoso.com/");
```

Pour indiquer quelle image utiliser à l'objet BitmapSource dans notre projet, il faut utiliser la syntaxe particulière suivante, soit un [URI à entête pack](#):



```
Uri maValeur = new Uri("pack://application:,,,/Images/Carré1.png");
```

Classe BitmapImage

Afin de pouvoir affecter l'image désirée à la propriété *Image.Source*, il faut convertir la ressource utilisée dans le bon format. [BitmapImage](#) existe à l'origine pour prendre en charge la syntaxe Extensible Application Markup Language (XAML) et introduit des propriétés supplémentaires pour le chargement de bitmap. Exemple :

```
img_Image.Source = new BitmapImage(new Uri("pack://application:,,,/Images/Carré2.png"));
```

Changer l'image d'un objet Button, Ellipse ou Label par programmation

Pour assigner une image à un objet de type *Button*, il faut utiliser la propriété *Background* alors que pour l'ellipse c'est *Fill*. Ces deux propriétés sont de type *Brush*.

Classe Brush

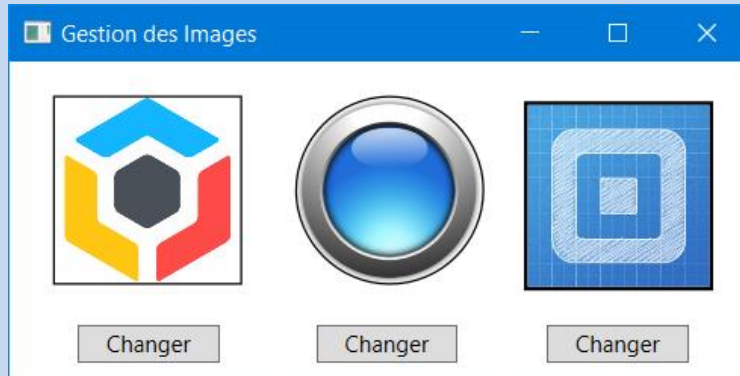
La classe [Brush](#) sert à définir les objets utilisés pour peindre des objets graphiques. Il existe plusieurs objets pour représenter les « brushes ». Concernant les images, il faut utiliser [ImageBrush](#). Exemple :

```
// Button
btnCliquer.Background =
new ImageBrush(new BitmapImage(new Uri("pack://application:,,,/Images/Bouton2.png")));

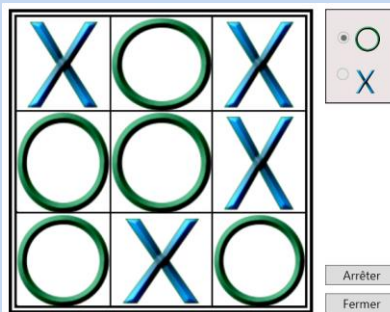
// Ellipse
```

```
imgEllipse.Fill =  
new ImageBrush(new BitmapImage(new Uri("pack://application:,,,/Images/Ellipse2.png")));
```

Exercice 10. : Ajouter trois boutons **Changer** sous les trois images. Ajouter le code qui permet d'alterner entre les images lorsque l'utilisateur clique sur un des boutons.



Exercice 11. : Dans la solution actuelle, créer le projet WPF nommé **TicTacToe**. Voici le formulaire à créer. Vous devez extraire les images du fichier .ZIP. Suivre les instructions du professeur.



Bibliographie

MacDonald, M. (2012). *Pro WPF 4.5 in C#*. New York: Apress.

Mosers, C. (2011, 01 01). *Introduction to WPF Layout*. Retrieved 01 10, 2016, from WPF Tutorial .Net: <http://www.wpftutorial.net/LayoutProperties.html>