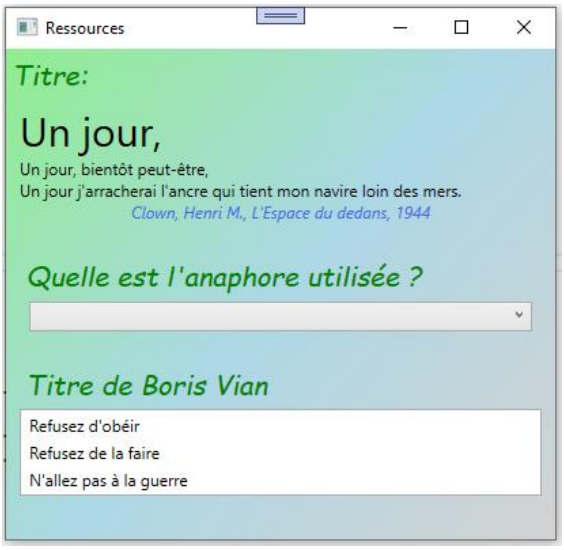
Styles et Panels

1. Les ressources en XAML:

- Le concept des ressources est un concept très pratique qui permet de sauvegarder un objet ou des données pour être réutilisés.
- Il existe trois méthodes pour définir une ressource :
    - Localement pour une fenêtre entière : La définition de la ressource se fera dans la balise <Window.Resource> avant la balise du panel principal de cette fenêtre.
    - Localement pour un seul contrôle.
    - Globalement pour l'application entière.
- Par exemples, les modèles, les styles et les pinceaux utilisent le concept des ressources.
- Une ressource est référencée depuis d'autres parties de l'application au moyen d'une clé utilisant l'attribut « x:Key ».

Exemple :



- Créez un nouveau projet « Ex_Ressource ».
- Dans une fenêtre 450 x 440, créez un « StackPanel » vertical pour contenir des « TextBlock ». L'interface graphique à créer contiendra l'anaphore avec l'expression « Un jour ». (Une anaphore est une figure de style par laquelle on répète un même mot ou un même groupe de mots en tête de phrases, de vers, de paragraphes qui se suivent (laculturegenerale.com)).

NB : Dans XAML, la ligne « `xmlns:sys="clr-namespace:System;assembly=mscorlib"` » représente la déclaration d'un espace de noms dans la balise racine « `Window` » de la fenêtre. Elle permet de faire référence aux primitives du langage CLR en mappant l'assembly « mscorlib » et l'espace de nom (namespace) « System ».

1.1. Définition d'une ressource locale à une fenêtre

- On peut définir une ressource locale à une fenêtre en l'ajoutant dans la balise `<Window.Resources>` dans la balise `<Window>`. Cette ressource ne pourra être référencée que dans la fenêtre où elle a été définie.
- Définissez la ressource « `keyUnJour` » de type chaîne de caractère pour contenir l'expression « Un jour » à référencer à partir des TextBlocks.
- Enlevez l'expression « Un jour » des TextBlock.
- Dans les « TextBlocks », référencez cette ressource statique.

1.2. Définition d'une ressource locale à un contrôle :

- On veut définir une chaîne de caractères comme ressource locale au contrôle nommé « StackPanelPrincipal » qui ne peut pas être référencée d'ailleurs.

  Puisque c'est une ressource locale à ce contrôle, nous allons donc utiliser une sous balise de ressource d'un « StackPanel ».
- Dans « StackPanelPrincipal », ajoutez un label qui référence cette chaîne de caractères :

  NB : Si on déplace le label à l'extérieur du contrôle « StackPanelPrincipal », le message d'erreur « `Impossible de résoudre la ressource "TitreListe"` » sera affiché.
- Il est possible de stocker un tableau de chaîne de caractères comme ressource à référencer. Créez cette ressource dans la balise `<Window.Resource>`.
- Dans « StackPanelPrincipal », insérez un « ComboBox » dont les items sont les éléments de l'Array « LesItems ».

1.3. Définition d'une ressource globale à la portée de l'application

- Une ressource globale est une ressource qui peut être référencée de n'importe où dans la même application (projet de Visual Studio). Elle doit être définit dans le fichier « App.xaml ».
- Par exemple, on veut définir une couleur dégradée pour l'arrière-plan pouvant être appliquée dans toutes les fenêtres de l'application.
- Définissez le dégradé de couleur suivant dans la balise `<Application.Resources>` du fichier « App.xaml » : « 0 LightGreen, 0.5 LigntBlue et 1 LightGray ».
- Dans la balise <Window> de « MainWindow.xaml », ajoutez l'attribut Background pour utiliser le dégradé de couleur définit dans « App.xaml ».

  NB : Vous pouvez ajouter cet attribut pour toutes les fenêtres de l'application (à venir).

1.4. Accès aux ressources à partir du code-behind

- Nous avons vu dans les exemples précédents comme utiliser des ressources dans le code XAML. Il est également possible d'utiliser des ressources à partir du code-behind.
- Ajoutez le label contenant le texte « Titre de Boris Vian » et le ListBox nommé « lbLignes » (sans items) dans « MainWindow.xaml ».
- Dans le fichier « App.xaml », ajoutez la ressource globale contenant la chaîne de caractères « Refusez d'obéir » et dont la clé est « strApp1 ».
- Dans le constructeur de la classe partielle « MainWindow », insérez « strApp1 » comme texte d'un item du ListeBox « lbLignes ».

  NB : Pour trouver la ressource « strApp1 », il est possible d'utiliser la méthode FindResource() qui cherche les ressources accessibles à partir de « StackPanelPrincipal » en les prenant en paramètre. Si la ressource est trouvée, la méthode FindResource la retourne en tant qu'objet.
- Dans « App.xaml », ajoutez les ressources globales « strWin2 » et « strWin3 » contenant les chaines de caractères « Refusez de la faire » et « N'allez pas à la guerre » respectivement.
- Dans le constructeur de « MainWindow », insérez ces 2 ressources comme items de « LbLignes ».

2. Les styles

- Les ressources sont aussi utilisées dans la définition des styles.
- Un style permet d'utiliser les mêmes attributs à plusieurs contrôles différents.
- Un style, comme dans Word, permet de faciliter la gestion des attributs de plusieurs contrôles en même temps.
- Dans le même projet, nous allons maintenant définir puis utiliser des styles.
- Nous avons déjà trois labels qui utilisent tous la même mise en forme : couleur du texte « vert », en italique, taille « 20 » et police « Comic Sans MS ».

  Ça sera donc intéressant de définir un style et de l'appliquer pour tous les labels pour alléger les balises <Label>. Tous les labels seront harmonisés et faciles à gérer et en cas de modification, il suffit de modifier juste le style.
- Dans `<Window.Resources>`, ajoutez la balise suivante pour qu'elle sera appliquée à tous les contrôles « Label » de la fenêtre « MainWindow ».

  Couleur de fond « Green », en « Italic », taille « 20 » et police de caractères « Comic Sans MS ».

  NB : L'attribut « TargetType » reçoit comme valeur le nom du contrôle qui aura le style, dans cet exemple le contrôle « Label ». Chaque sous-balise « Setter » a deux attributs : l'attribut du contrôle, exemple « Foreground » et la valeur à lui affecter, exemple « Green ».

- Enlevez ces quatre attributs déjà définis dans les balises des contrôles « Label ». Remarquez que les labels ont toujours la même mise en forme, qui cette fois est appliquée par le style définit.
- Pour que le style soit visible et appliqué automatiquement à tous les « Labels » de l'application, déplacez-le dans le fichier « App.xaml ».
- Quand un style possède un nom (Key), il sera possible de l'appliquer seulement à quelques contrôles de même type. Ainsi, le style n'est appliqué que lorsqu'il est appelé.
- Dans `<Window.Resources>`, créez un style ciblant les « TextBlock » et dont la clé est « monStyle ». Ce style doit utiliser les propriétés suivantes : alignement horizontal : centré, style de texte : Italique, et couleur de texte : RoyalBlue.
- Enlevez les propriétés (HorizontalAlignment, FontStyle et Foreground) dans le « textBlock » contenant le texte « Clown, Henri M., L'Espace du dedans, 1944 ». Évidemment, sur l'interface utilisateur la mise en forme de cette ligne a disparu.
- Faites appel à la ressource statique « monStyle » pour l'appliquer à ce « TextBlock ».

3. Menu

- Dans le « Grid », ajoutez un « stackPanel » permettant d'englober tous les contrôles de « MainWindow ».
- Nous allons ajouter un menu contenant les options « Fichier » et « Dessiner ». Ce menu sera ajouté dans un nouveau « StackPanel » en orientation « horizontale ».
    - Sous « Fichier », ajoutez les items « Nouveau », « Ouvrir », « Enregistrer » et « Sortir ». Ajoutez un séparateur entre « Sortir » et les autres items.
    - Sous « Dessiner », ajoutez l'item « InkCanvas ».
- Nous voulons ajouter l'icône d'un Canvas à gauche de l'item « InkCanvas ». Cherchez sur Google « Canvas Icon » et sauvegardez le fichier sur votre ordinateur. À partir de l'explorateur de solutions, ajoutez un nouveau dossier « images » dans le projet actuel puis y ajoutez l'icône téléchargée.

  NB : Si l'image de l'icône n'apparaît pas lors de l'exécution de l'application, bouton droite sur le nom du projet (dans l'explorateur de solutions) puis « Regénérer ».
- Ajoutez l'événement, qui sur clic sur l'item « Sortir », il ferme la fenêtre.
- Ajoutez un autre événement qui sur clic sur l'item « Nouveau », permet de lancer l'application « notepad ».

  NB : Il s'agit d'un nouveau processus qui sera lancé en utilisant la classe « Process » se trouvant dans « System.Diagnostics ». Il faudrait l'ajouter avec l'instruction :

```
using System.Diagnostics;
```

NB : Quand une erreur est détectée à cause du manque de l'utilisation d'un assembly, cliquez avec le bouton droit sur l'erreur, « Actions rapides et refactorisations… » pour afficher les ***propositions*** de correction (qui peuvent être bonnes ou erronées).

4. Application avec plusieurs fenêtres

- Sur clic sur un contrôle de « MainWindow », on veut ouvrir une autre interface utilisateur.
- Pour créer une nouvelle fenêtre dans le projet actuel, faites un clic droit sur le nom du projet dans l'explorateur de solution, puis choisissez « Ajouter / Nouvel élément/ Fenêtre WPF », appelez-la « Toile ».
    - Constatez l'ajout des fichiers « Toile.xaml » et « Toile.xaml.cs » dans le projet.
- Mettez les dimensions de la fenêtre « Toile » à 400 x 600. Le style de la fenêtre doit être « ToolWindow ».
- Ajoutez l'événement qui, sur clic sur l'item « InkCanvas » du menu « Dessiner » ouvre l'interface utilisateur « Toile ».

5. Le panel « InkCanvas »

« InkCanvas » est un élément assez intéressant et particulier. Il définit une zone qui reçoit et affiche des traits d'encre. Il permet de capturer l'écriture manuscrite (en utilisant la souris ou le stylet) sans qu'on ait à tout programmer.

En fait, ce n'est pas vraiment un Panel. Il est plutôt un « FrameworkElement », pas un type de Canvas.

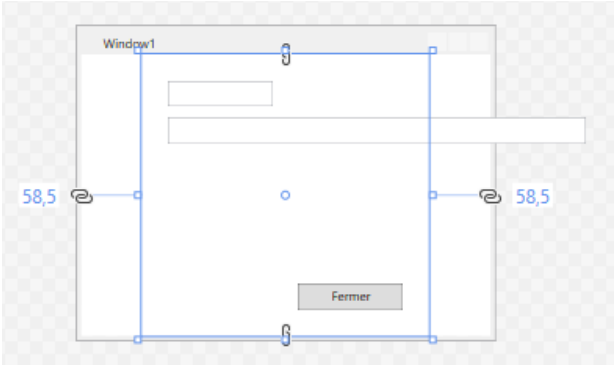
- Dans la fenêtre « Toile », ajoutez la balise « InkCanvas ». Vérifiez que vous pouvez désormais dessiner sur toute la grille.
- Le contrôle « InkCanvas » occupe toute la surface de la fenêtre. On peut le remarquer par l'absence de la couleur d'arrière-plan de « Grid ». Mettez l'arrière-plan de « InkCanvas » transparent.
- Nous voulons changer la taille du pinceau à 5x5 et sa couleur en bleu. Pour définir ces attributs, il faut utiliser « InkCanvas.DefaultDrawingAttributes ».

6. Le Panel « Canvas »

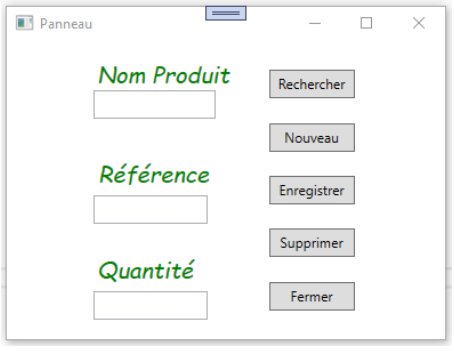
- Canvas est le panel le plus basic. Il supporte la notion classique de positionnement des éléments sur une interface utilisateur. En effet, un Canvas utilise les coordonnées X et Y pour positionner des éléments de manière absolue.
- Ajoutez dans le menu l'item « Panel » après l'item « Fichier ».

- Sur clic sur ce menu, une fenêtre nommée « Panneau » (à ajouter) sera créée.
- Dans le fichier « Panneau.xaml », remplacez la balise « Grid » par la balise « Canvas ».
- Référencez la ressource globale « WindowBackGroundBrush » permettant de changer la couleur d'arrière-plan de la fenêtre.
- Dans un Canvas, les éléments sont positionnés en utilisant des propriétés « left », « top », « right » et « bottom ».
    - La syntaxe utilise Canvas car c'est une propriété qui hérite de Grid.
- Un enfant du Canvas peut être dessiné à l'extérieur des limites du Canvas, et même par-dessus le contenu qui se trouve à l'extérieur.
- Pour bien illustrer, on va réduire la fenêtre et le Canvas sans changer les dimensions du TextBox.

  Remarquez le dépassement du TextBox



- Le positionnement des éléments est relatif aux bordures du Canvas dans la fenêtre « Window ». L'alignement horizontal par défaut est « center ». Testez les valeurs de l'attribut « HorizontalAlignment » : « left », « right ».
- On désire créer une interface utilisateur similaire à celle présentée ci-dessous.



- Avec le panel « Canvas », il suffirait de créer les contrôles avec « la boîte à outils » puis les positionner approximativement avec la souris.

  Remarquez que les propriétés Canvas.Left, Canvas.Top, Canvas.Right et Canvas.Bottom changent de valeurs.

NB: Remarquez que tous les contrôles de type “Label” ont pris la forme du style global défini dans le fichier « App.xaml ».

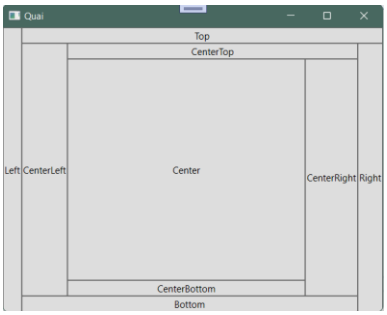
7. DockPanel

- DockPanel permet d’amarrer des contrôles dans les quatre directions du panel.
    - Divise la fenêtre en différentes zones : à gauche (Left), en haut (Top), à droite (Right) et en bas (Bottom).
    - L’ordre dans lequel les éléments sont placés peut changer la taille. C’est-à-dire, si on mentionne le contrôle qui sera en haut le premier, il prendra toute la largeur du DockPanel.
    - Le dernier élément ajouté au DockPanel remplira automatiquement l’espace central restant.
- Dans le menu « Panel », ajoutez le sous menu « DockPanel ».
- Ajoutez une nouvelle fenêtre WPF et nommez-la « Quai ».
- Ajoutez le code-behind de l’événement qui, sur clique, affiche la fenêtre « Quai »
- Écrivez le code suivant dans le fichier « Quai.xaml » :

```
    ...
    Title="Quai" Height="400" Width="500">
<DockPanel>
    <Button DockPanel.Dock="Left">Left</Button>
    <Button DockPanel.Dock="Top">Top</Button>
    <Button DockPanel.Dock="Right">Right</Button>
    <Button DockPanel.Dock="Bottom">Bottom</Button>
    <Button>Center</Button>
</DockPanel>
```

- Après la balise du bouton « Bottom », ajoutez les boutons suivants :

```
    <Button DockPanel.Dock="Left">CenterLeft</Button>
    <Button DockPanel.Dock="Top">CenterTop</Button>
    <Button DockPanel.Dock="Right">CenterRight</Button>
    <Button DockPanel.Dock="Bottom">CenterBottom</Button>
```



- Permutez l’ordre de quelques balises et constatez les modifications.