Services réseaux

Cours n°3 - TP n°2 : .net – Smart appli

Antony BRUGERE
anto.brugere@gmail.com

Frédéric CHASSAGNE
frederic.chassagne@capgemini.com

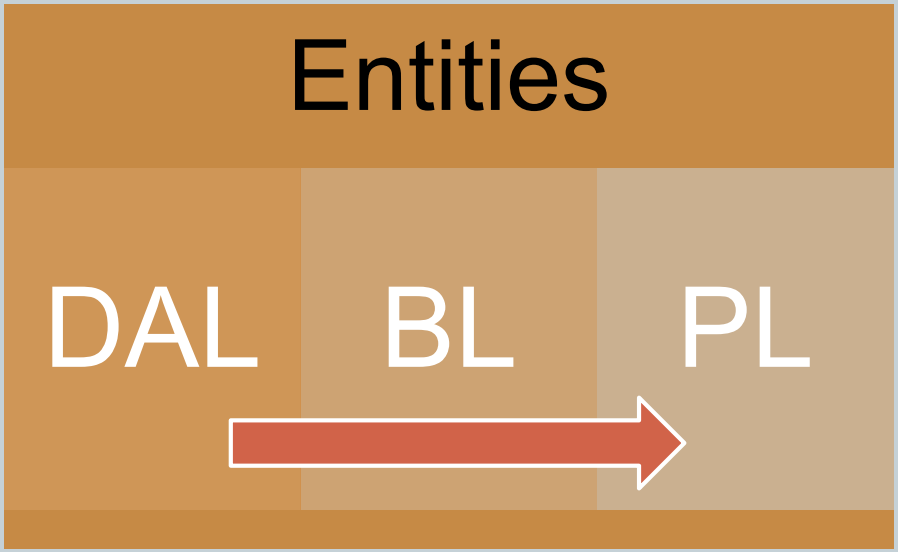
Plan du TP intégré

- Synthèse du TP précédent
- Présentation WPF
- Sérialisation
- WPF - Xaml
- Binding en WPF
- Gestion des erreurs

Architecture logicielle

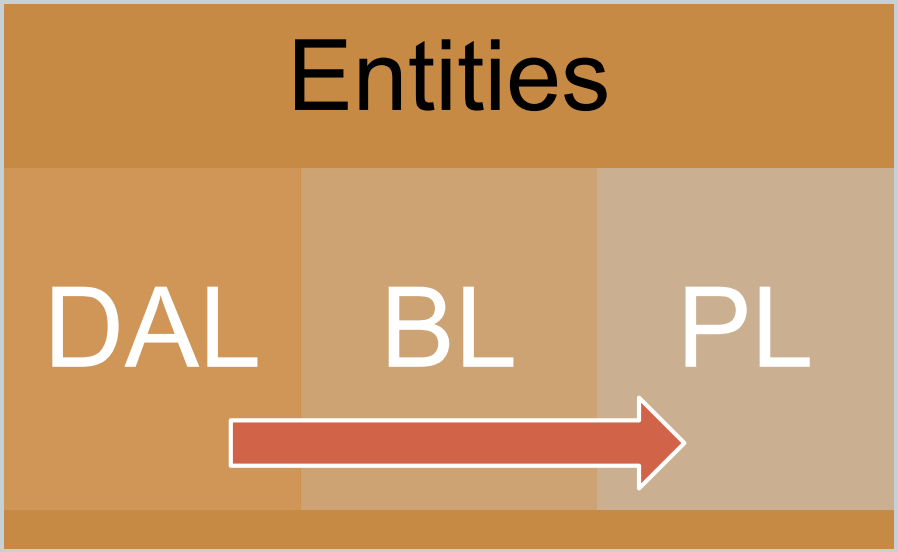
- Couches du modèle 4 tiers
  - Presentation Layer
    - Interface Homme Machine
  - Business Layer
    - Noyau métier
  - Data Access Layer
    - Accès aux données
  - Entities Layer
    - Objets basiques

    ie Plain Old Clr Objects



Architecture logicielle

- Couches du modèle 4 tiers / TP1
  - Presentation Layer
    - Projet console
  - Business Layer
    - Classe BusinessManager
  - Data Access Layer
    - Classe DalManager + Stub
  - Entities Layer
    - Evenement, Artiste



Objectifs du TP 2

- Mon Smart Agenda
  - Réutiliser la bibliothèque de classe du TP n°1
  - Créer une smart application WPF attrayante qui gère le stockage de ses dimensions.
    - Manipuler des fichiers xaml
    - Créer des windows
    - Utiliser la sérialisation

Smart Client

- 2 moyens pour des IHM « Smart Client »
  - Windows.Forms
  - WPF
- Avantages
  - Accès complet à l'ordinateur et au réseau
  - Aucun maintien de session
- Inconvénients
  - Peu interopérable
  - Installation nécessaire sur chaque poste (lourd)

WPF

- WPF = Windows Presentation Foundation
- Successeur de Windows.Forms
- Amélioration des possibilités
  - Intégration des flux Audio / Vidéo
  - Indépendance par rapport à la résolution (vectoriel)
  - Accès à la 3D
  - Customisation et enrichissement visuel facilités

WPF

- WPF = Mutualisation de différentes API

WPF

- En résumé, WPF = mix de
  - DirectX
  - Windows Forms
  - Adobe Flash
  - Html

- WPF ne permet pas de faire de nouvelles choses, WPF permet de le faire plus simplement

WPF

- 2 types d'applications possibles en WPF
  - « stand alone » ou « smart application », installées sur la machine cliente
  - « navigateur », téléchargées depuis internet et exécutées au travers d'un navigateur web (XBAP) (≠ Silverlight)

- Intérêts
  - Développement interface graphique enrichies
  - Séparation Interface / Implémentation

Etape 1 : Création du projet

- Créer un projet de type WPF Application
- Référencer les dll BusinessLayer

Interaction utilisateur

- But d'un programme = satisfaction client
- Satisfaction client = jolie interface

- But d'un programme = jolie interface

Interaction utilisateur

- Points importants d'une jolie interface
  - Regroupements logique
  - Alignement des contrôles
  - Pas de formulaire « placard à tout faire »
  - Mise en avant des informations importantes
  - Interaction avec l'environnement
    - Sauvegarde des dimensions / positions de l'ihm
    - Gestion des erreurs « Friendly »
  - Privilégier le « Windows » look
  - Avoir du « bon sens »

Interaction utilisateur

- Interaction utilisateur
  - Interaction avec l'application
  - Interaction avec l'environnement

  - → Simplifier la vie de l'utilisateur

Sérialisation

- Sérialisation = Conversion d'un objet sous une forme transportable
- Désérialisation = Conversion d'un flux de données en objet
- 3 grands types de sérialisation
  - Binaire
  - SOAP
  - XML

Sérialisation

- Avantages sérialisation binaire
  - Complète
  - Orientée métier (format non lisible)
- Avantages sérialisation SOAP
  - Complète
  - Orientée données (format lisible)
- Avantages sérialisation XML
  - Restrictive
  - Orientée Configuration (format lisible)

Sérialisation

- Comparaisons

	XML	SOAP	Binaire
"Human readable"	oui	oui	non
Sérialisation de types non standards	non	oui	oui
Sérialisation des éléments privés	non	oui	oui
Sérialisation des champs	non	oui	oui
Sérialisation des propriétés	oui	non	non

Sérialisation

- Principe
  - Marquer la classe comme [Serializable] (convention en XML)
  - Gérer les accesseurs
    - Implémenter une méthode de Load()
    - Implémenter une méthode de Save()

Sérialisation

- Exemple

```
[SerializableAttribute]
  public class Homme
  {
    public string Nom;
    public string Prenom;

    public Homme(string prenom, string nom)
    {
      Prenom = prenom;
      Nom = nom;
    }
  }
```

```
[SerializableAttribute]
  public class Hommes : System.Collections.
    CollectionBase
  {
    public void Add(Homme HommeA)
    {
      this.List.Add(HommeA);
    }

    public Homme this[int Index]
    {
      get
      { return (Homme)this.List[Index]; }
    }
  }
```

Sérialisation

- Sérialisation binaire
  - using System.Runtime.Serialization.Formatters.Binary;
- Accesseurs

Sauvegarde

```
FileStream mFile = new FileStream(@"c:
    \easyBin.net", FileMode.Create);

BinaryFormatter mS = new BinaryFormatter();
mS.Serialize(mFile, _lesHommes);
mFile.Close();
```

Chargement

```
FileStream mFile = new FileStream(@"c:
    \easyBin.net", FileMode.Open);
    BinaryFormatter mS = new
    BinaryFormatter();
    _lesHommes = (Hommes)mS.Deserialize
    (mFile);
    mFile.Close();
```

Sérialisation

- Sérialisation SOAP
  - using System.Runtime.Serialization.Formatters.Soap;
- Accesseurs

Sauvegarde

```
FileStream mFile = new FileStream(@"c:
    \easySoap.net", FileMode.Create);

SoapFormatter mS = new SoapFormatter ();
mS.Serialize(mFile, _lesHommes);
mFile.Close();
```

Chargement

```
FileStream mFile = new FileStream(@"c:
    \easySoap.net", FileMode.Open);
    SoapFormatter mS = new
SoapFormatter ();
    _lesHommes = (Hommes)mS.Deserialize
(mFile);
    mFile.Close();
```

Sérialisation

- Sérialisation XML
  - using System.Xml.Serialization;
- Accesseurs

Sauvegarde

```
StreamWriter stream = new StreamWriter(@"c:
    \easyXML.net");

XmlSerializer serializer = new XmlSerializer
    (typeof(Hommes));
serializer.Serialize(stream, _lesHommes);
stream.Close();
```

Chargement

```
XmlSerializer deserializer = new XmlSerializer
    (typeof(Hommes));
StreamReader stream = new StreamReader(@"c:
    \easyXML.net");

_lesHommes = (Hommes)deserializer.
    Deserialize(stream);
stream.Close();
```

Sérialisation

- Sérialisation XML
  - Marquer la classe comme [Serializable] (convention)
  - Exclure les attributs non souhaités avec [XmlIgnoreAttribute]
- Restrictions
  - Nécessite un constructeur par défaut.
  - Sérialise seulement les propriétés/champs publiques non « readonly »

Etape 2 : Classe sérialisation

- Objectif :
  - Sauvegarder le positionnement et la taille des fenêtres par utilisateur.

- Dans quelle couche ?
- A quel moment réaliser
  - La lecture
  - La sauvegarde
  - La mise à jour

WPF - XAML

- XAML = eXtensible Application Markup Language
  - Langage basé sur xml
  - Décrit les interfaces graphiques en WPF, Silverlight
  - Utilisé par des outils de design tel que Blend

Partie designer

- La logique de l'application est dans le code source C# ou VB.net

Partie developpeur

WPF - XAML

- Xaml = Description hiérarchique et intuitive
- Exemple 1

```
<Button xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”
Content=”OK” Click=”button_Click”>
   <Button.Background>
         <SolidColorBrush>
               <SolidColorBrush.Color>
                     <Color A=”255” R=”255” G=”255” B=”255”/>
               </SolidColorBrush.Color>
         </SolidColorBrush>
   </Button.Background>
</Button>
```

WPF - XAML

- Exemple 2

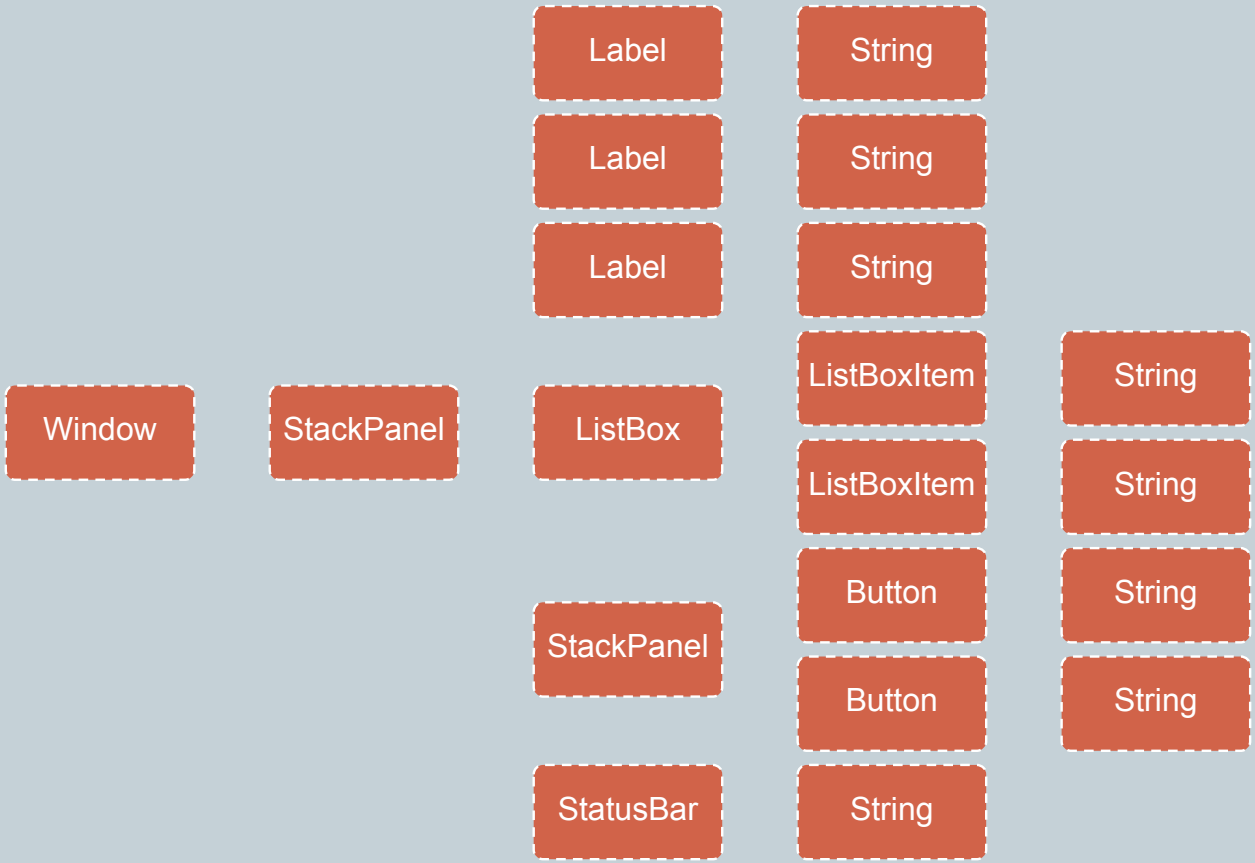
```
<Window xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”
Title=”About WPF” SizeToContent=”WidthAndHeight” Background=”OrangeRed”>
    <StackPanel>
        <Label FontWeight=”Bold” FontSize=”20” Foreground=”White”>
            WPF Sample
        </Label>
        <Label>Nov 2011</Label>
        <Label>Sample List Box</Label>
        <ListBox>
            <ListBoxItem>Item 1</ListBoxItem>
            <ListBoxItem>Item 2</ListBoxItem>
        </ListBox>
        <StackPanel Orientation=”Horizontal” HorizontalAlignment=”Center”>
            <Button MinWidth=”75” Margin=”10”>Help</Button>
            <Button MinWidth=”75” Margin=”10”>OK</Button>
        </StackPanel>
            <StatusBar>Sample's understanding 100%.</StatusBar>
    </StackPanel>
</Window>
```

WPF - XAML

- Résultat de l'exemple 2

WPF - XAML

- Xaml = représentation en arbre logique



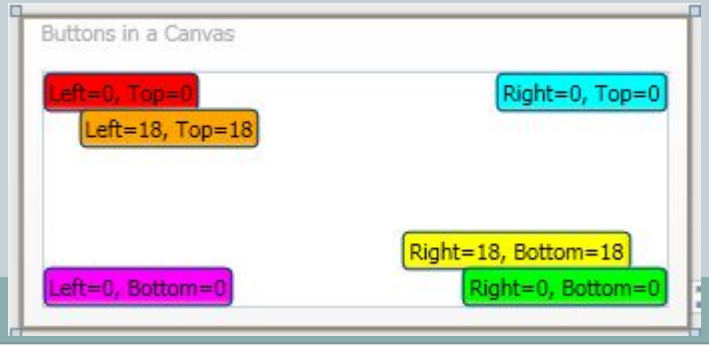
WPF - XAML

- Imbrication hiérarchique de containers et de contrôles
- Principaux containers disponibles
  - Canvas
  - StackPanel
  - WrapPanel
  - DockPanel
  - Grid

WPF - XAML

- Canvas
  - Très (Trop ?) Basique
  - Chaque contrôle définit Left, Top, Right, Bottom

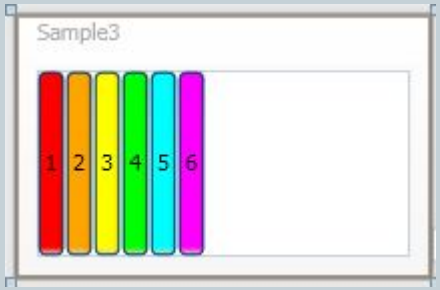
```
<Window xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”
Title=”Buttons in a Canvas”>
    <Canvas>
            <Button Background=”Red”>Left=0, Top=0</Button>
            <Button Canvas.Left=”18” Canvas.Top=”18” Background=”Orange”>Left=18, Top=18</Button>
            <Button Canvas.Right=”18” Canvas.Bottom=”18” Background=”Yellow”>Right=18, Bottom=18</Button>
            <Button Canvas.Right=”0” Canvas.Bottom=”0” Background=”Lime”>Right=0, Bottom=0</Button>
            <Button Canvas.Right=”0” Canvas.Top=”0” Background=”Aqua”>Right=0, Top=0</Button>
            <Button Canvas.Left=”0” Canvas.Bottom=”0” Background=”Magenta”>Left=0, Bottom=0</Button>
    </Canvas>
</Window>
```



WPF - XAML

- StackPanel
  - Container communément utilisé, simple et pratique
  - Définition de l'orientation sur le StackPanel

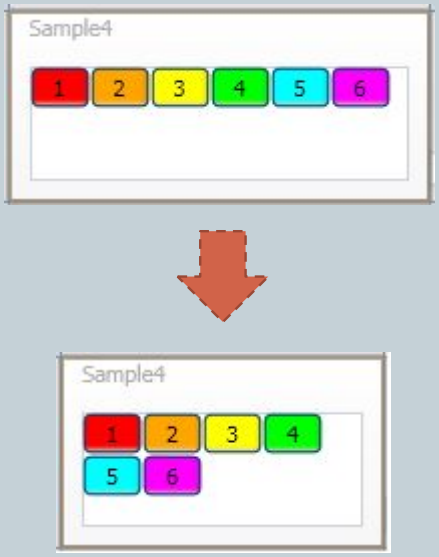
```
<Window x:Class="MonAgendaSmart.Sample3"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="Sample3" Height="132" Width="208">
   <StackPanel Orientation="Horizontal" >
      <Button Background="Red">1</Button>
      <Button Background="Orange">2</Button>
      <Button Background="Yellow">3</Button>
      <Button Background="Lime">4</Button>
      <Button Background="Aqua">5</Button>
      <Button Background="Magenta">6</Button>
   </StackPanel>
</Window>
```



WPF - XAML

- WrapPanel
  - Similaire au StackPanel
  - Réarrange les contrôles en fonction de la place disponible

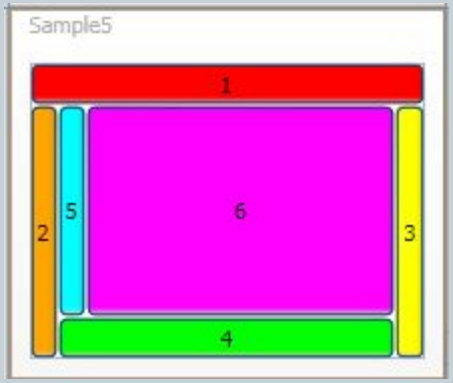
```
<Window x:Class="MonAgendaSmart.Sample4"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="Sample4" Height="96" Width="162">
   <WrapPanel Orientation="Horizontal" >
      <Button Width="30" Background="Red">1</Button>
      <Button Width="30" Background="Orange">2</Button>
      <Button Width="30" Background="Yellow">3</Button>
      <Button Width="30" Background="Lime">4</Button>
      <Button Width="30" Background="Aqua">5</Button>
      <Button Width="30" Background="Magenta">6</Button>
   </WrapPanel>
</Window>
```



WPF - XAML

- DockPanel
  - Plus évolué que le StackPanel
  - Permet d'arrangement 5 cinq directions : Top, Left, Right, Bottom et Center

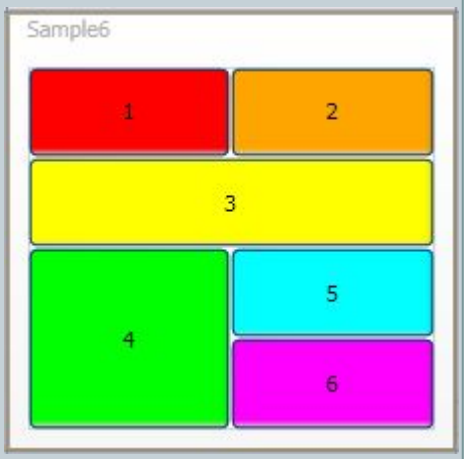
```
<Window x:Class="MonAgendaSmart.Sample5"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="Sample5" Height="300" Width="300">
   <Grid>
      <DockPanel>
         <Button DockPanel.Dock="Top" Background="Red">1</Button>
         <Button DockPanel.Dock="Left" Background="Orange">2</Button>
         <Button DockPanel.Dock="Right"  Background="Yellow">3</Button>
         <Button DockPanel.Dock="Bottom" Background="Lime">4</Button>
         <Button Background="Aqua">5</Button>
         <Button Background="Magenta">6</Button>
      </DockPanel>
   </Grid>
</Window>
```



WPF-XAML

- Grid
  - Container le + customisable, proposé par défaut dans VS2010
  - Permet de définir des Lignes et colonnes

```
<Window x:Class="MonAgendaSmart.Sample6" Title="Sample6" Height="220" Width="225">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition/><RowDefinition/><RowDefinition/><RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition/><ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Button Grid.Column="0" Grid.Row="0" Background="Red">1</Button>
    <Button Grid.Column="1" Grid.Row="0" Background="Orange">2</Button>
    <Button Grid.ColumnSpan="2" Grid.Row="1" Background="Yellow">3</Button>
    <Button Grid.Column="0" Grid.Row="2" Grid.RowSpan="2" Background="Lime">4</Button>
    <Button Grid.Column="1" Grid.Row="2" Background="Aqua">5</Button>
    <Button Grid.Column="1" Grid.Row="3" Background="Magenta">6</Button>
  </Grid>
</Window>
```



Etape 3 et 4 : Windows

- Main windows à étoffer
- Créer la fenêtre de connexion

Gestion des erreurs

- Une étape importante à ne pas oublier dans vos futurs développements : une gestion d'erreurs efficace.

- Une bonne gestion d'erreurs permet :
  - D'éviter le « crash » de votre application (= fermeture soudaine).
  - D'afficher des messages simples et non techniques à l' utilisateur (évite souvent une panique inutile).
  - De gagner beaucoup de temps lors des phases de test.

Gestion des erreurs en WPF

- En WPF toutes les exceptions peuvent être récupérées grâce à l'événement « DispatcherUnhandledException » dans le fichier « App.cs »

- Une bonne gestion pourrait être, en cas d'exception, de :
  - Afficher un message « friendly » à l'utilisateur.
  - Loguer l'erreur dans un fichier.

Exemple gestion d'erreurs

```
public partial class App : Application
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);
        DispatcherUnhandledException += App_DispatcherUnhandledException;
    }

    void App_DispatcherUnhandledException(object sender, System.Windows.Threading.DispatcherUnhandledExceptionEventArgs e)
    {
        try
        {
            //Récupération de toutes les exceptions :
            MessageBox.Show(Assembly.GetExecutingAssembly().GetName().Name , "Une erreur est survenue mais pas de panique tout est sous contrôle. ",
                            MessageBoxButton.OK, MessageBoxImage.Error);

            //Si vous ne mettez pas le handled a true l'application "crash" et se ferme.
            e.Handled = true;

            //On log l'exception dans un fichier
            ClogManager.WriteLog(e.ToString());
        }
        catch (Exception)
        {
            //Seul cas possible ou un catch peut être vide : une gestion d'erreur ne DOIS PAS planter.
        }
    }
}
```

Etape 5 et 6 : Exception et Windows « enfants »

- Implémenter une gestion d'erreurs
- Créer les nouvelles windows
  - Containers
  - Contrôles
  - Affichage