

DataGrid, Data binding et UserControl

Contenu

CONTENU

1. DATAGRID
2. NOTION DE « TEMPLATE » (MODÈLE) EN WPF
3. UTILISATION D'UN CONTROLTEMPLATE
4. DATA BINDING VERS DES OBJETS DE DONNÉES
 - 4.1. UTILISATION DE LA CLASSE « OBSERVABLECOLLECTION »
 - 4.2. IMPLÉMENTATION DE L'INTERFACE « INOTIFYPROPERTYCHANGED »
5. UTILISATION D'UN « LISTVIEW » POUR L'AFFICHAGE DES DONNÉES
6. UTILISATION D'UN DATATEMPLATE
7. CRÉATION DE LA FENÊTRE « INPUTDIALOG »
8. UTILISATION DE « INPUTDIALOG » COMME FENÊTRE MODALE
9. UTILISATION DE « USERCONTROL »
 - 9.1. CHARGEMENT D'UN USERCONTROL SUR TOUTE LA SURFACE D'UNE FENÊTRE
 - 9.2. UN DEUXIÈME USERCONTROL
 - 9.3. CHARGEMENT DE PLUSIEURS USERCONTROL DANS UNE MÊME FENÊTRE

1. DataGrid

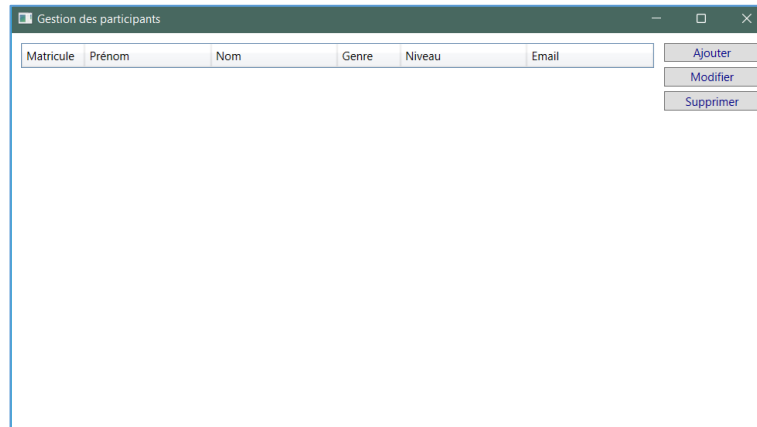
Un DataGrid (grille de données) est un contrôle qui affiche les données dans une grille personnalisable. Il fournit un moyen flexible d'afficher une collection de données dans des lignes et des colonnes. Un DataGrid peut être rempli manuellement ou avec du code-behind.

On veut créer l'interface utilisateur ci-dessous pour la gestion des données des participants. Les informations d'un participant sont : Matricule, Prénom, Nom, Genre, Niveau et Email. Les données d'un participant seront représentées sur une ligne du DataGrid.

À droite de l'interface utilisateur se trouvent trois boutons :

- Le bouton « Ajouter » : permettra d'ajouter un nouvel étudiant dans le DataGrid.
- Le bouton « Modifier » : permettra de modifier les données d'un étudiant.

- Le bouton « Supprimer » : permettra de supprimer un étudiant du DataGrid.



2. Notion de « Template » (modèle) en WPF

Un Template décrit l'aspect général et visuel d'un contrôle. Chaque contrôle possède un modèle par défaut qui lui est associé pour donner son apparence. Dans WPF, il est possible de personnaliser le comportement visuel et l'apparence d'un contrôle.

Il existe deux types de modèles :

- Les modèles de contrôle (**ControlTemplate**) : définit une nouvelle apparence visuelle d'un contrôle différente de celle par défaut.

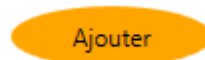
Exemple : Par défaut, un « Button » a la forme d'un rectangle cliquable. On peut utiliser un modèle de contrôle pour personnaliser le bouton.

- Les modèles de données (**DataTemplate**) : formate et définit la présentation d'une collection de données sur un élément de l'interface utilisateur. Un modèle de données est le plus souvent utilisé sur un contrôle d'élément lié aux données comme le DataGrid, ListView, ComboBox, ListBox, etc.

3. Utilisation d'un ControlTemplate

Nous avons déjà vu les styles qui permettent de personnaliser l'apparence d'un contrôle. La différence avec le modèle de contrôle c'est que le style fait des modifications seulement sur les propriétés déjà existantes alors que le « ControlTemplate » permet de définir de nouveaux aspects visuels à un contrôle.

On veut personnaliser l'apparence visuelle du bouton « Ajouter » pour la forme d'une ellipse de couleur « Orange ».



- Commençons par définir une ressource au niveau de l'élément « Window » en y définissant le contrôle « **ControlTemplate** » qui cible le type de contrôle "**Button**" grâce à la propriété « **TargetType** ».

Dans un panel « Grid », nous définissons les éléments qui composent le modèle : l'élément « **Ellipse** » pour la nouvelle forme du bouton et l'élément « **ContentPresenter** » pour y mettre le « Content » du bouton.

- Ensuite, ajoutez la propriété « **Template** » au bouton « Ajouter » pour le relier avec « **StaticResource** » au « **ControlTemplate** » ayant comme clé « ButtonTemplate ».

On veut aussi personnaliser l'apparence visuel du bouton « Modifier » pour l'afficher sous forme de rayons sur les côtés.

- Définissez un nouveau « ControlTemplate » dont la clé est « RadiusButtonTemplate » visant les boutons et utilisant les contrôles « Border » et « ContentPresenter ».
- Faites appel au Template dans la balise du bouton « Modifier ».

Maintenant, on veut personnaliser l'apparence visuelle et le comportement du bouton « Supprimer ». Nous allons définir un ControlTemplate formé d'un Grid qui contient un Rectangle et un TextBlock. Le Rectangle est utilisé pour dessiner le fond du bouton et le TextBlock est utilisé pour afficher le contenu du bouton.

Ensuite, nous allons ajouter un déclencheur « **ControlTemplate.Triggers** » pour changer la couleur de remplissage du Rectangle lorsque la souris survole le bouton.

- Définissez le « ControlTemplate »
- Faites appel au Template dans la balise du bouton « Supprimer ».

4. Data binding vers des objets de données

La liaison de données (Data Binding) est le processus qui établit une connexion entre l'interface utilisateur de l'application et les données qu'elle affiche.

- **La source de la liaison:** l'objet réel contenant les données. Il peut s'agir d'une chaîne, d'un entier ou même d'un objet de données personnalisé tel qu'une classe ou une liste, etc.
- **La cible de la liaison:** le contrôle qui représente les données de la source sur l'interface utilisateur. Exemples : DataGrid, ListView, ListView, ComboBox, ListBox, etc.

WPF dispose d'une large variété de contrôles pour afficher une liste de données. Ces contrôles se présentent sous plusieurs formes et varient selon leur complexité et les tâches qu'ils peuvent accomplir.

4.1. Utilisation de la classe « *ObservableCollection* »

- Ajoutez une nouvelle classe « Participant ».

```
public class Participant
{
    public int Matricule { get; set; }
    public string Prenom { get; set; }
    public string Nom { get; set; }
    public char Genre { get; set; }
    public string Niveau { get; set; }
    public string Email { get; set; }
}
```

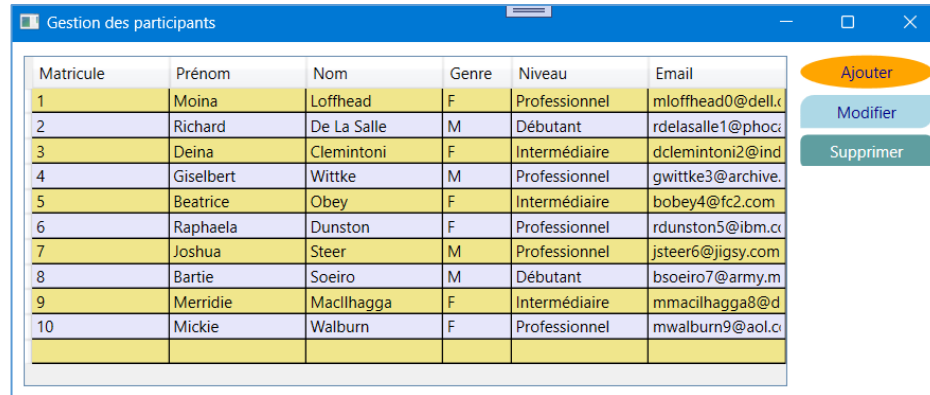
- Dans le fichier « XAML » de la fenêtre principale, ajoutez les liaisons nécessaires pour les trois propriétés « Matricule », « Prenom » et « Nom ».

NB: Le binding dans les instructions ci-dessus est de type « Unidirectionnel » car on récupère des valeurs de la source de données et on les affiche dans le DataGrid.

- Dans la classe partielle de la fenêtre principale, créez une liste de participants public.
- Remplissez la liste « participants » dans le constructeur de la classe « MainWindow » avec les informations suivantes :

Matricule	Prénom	Nom	Genre	Niveau	Email
1	Moina	Loffhead	F	Professionnel	mloffhead0@dell.com
2	Richard	De La Salle	M	Débutant	rdelasalle1@phoca.cz
3	Deina	Clemintoni	F	Intermédiaire	dclemintoni2@independent.co.uk
4	Giselbert	Wittke	M	Professionnel	gwittke3@archive.org
5	Beatrice	Obey	F	Intermédiaire	bobey4@fc2.com
6	Raphaëla	Dunston	F	Professionnel	rdunston5@ibm.com
7	Joshua	Steer	M	Professionnel	jsteer6@jigsy.com
8	Bartie	Soeiro	M	Débutant	bsoeiro7@army.mil
9	Merridie	MacIlhagga	F	Intermédiaire	mmacilhagga8@dell.com
10	Mickie	Walburn	F	Professionnel	mwalburn9@aol.com

- **La source** des items du DataGrid « dgParticipant » est la liste « participants ». Ajoutez l'instruction nécessaire.



Matricule	Prénom	Nom	Genre	Niveau	Email
1	Moina	Loffhead	F	Professionnel	mloffhead0@dell.com
2	Richard	De La Salle	M	Débutant	rdelasalle1@phoc.com
3	Deina	Clemintoni	F	Intermédiaire	dclemintoni2@ind.com
4	Giselbert	Wittke	M	Professionnel	gwittke3@archive.com
5	Beatrice	Obey	F	Intermédiaire	bobey4@fc2.com
6	Raphaela	Dunston	F	Professionnel	rdunston5@ibm.com
7	Joshua	Steer	M	Professionnel	jsteer6@jigsy.com
8	Bartie	Soeiro	M	Débutant	bsoeiro7@army.mil
9	Merridie	MacIlhagga	F	Intermédiaire	mmacilhagga8@domain.com
10	Mickie	Walburn	F	Professionnel	mwalburn9@aol.com

- Définissez la méthode d'ajout d'un participant à la liste de participant avec des valeurs prédéfinies.
 - ➔ En testant le bouton, le nouveau participant n'apparaît pas dans l'interface utilisateur. Nous allons comme-même continuer, puis on verra ce qu'il faut faire.
- Définissez une méthode, qui après avoir modifier les informations d'un participant directement dans le DataGrid, affiche les données du participant dans une boîte de messages (MessageBox).
 - ➔ L'affichage ci-dessus avec MessageBox permet d'afficher les valeurs des propriétés de l'objet « Participant » après avoir appliqué les modifications apportées à une ligne sélectionnée du DataGrid.
- Définissez une méthode qui permet de supprimer de la liste de participants l'enregistrement (la ligne) sélectionnée dans le DataGrid.
 - ➔ Aucun changement n'est observé sur l'interface utilisateur quand on clique sur le bouton supprimer.

Pour que les deux boutons « Ajouter » et « Supprimer » fonctionnent convenablement, nous devons juste remplacer « `List<Participant>` » par « `ObservableCollection<Participant>` ». « ObservableCollection » s'utilise de façon similaire à une « List » moyennant quelques différences.

```
using System.Collections.ObjectModel;
```

```
public ObservableCollection<Participant> participants = new ObservableCollection<Participant>();
```

Conclusion : La classe de collection « ObservableCollection » est utilisée dans les cas où on veut que les changements de la source d'une liste se reflètent dans un binding de destination. ➔ Binding de type « Unidirectionnel » : le changement des données dans la source se reflète dans le contrôle.

4.2. Implémentation de l'interface « *INotifyPropertyChanged* »

- Pour que nos objets de type « Participant » soient capables d'avertir l'interface utilisateur des changements de leurs propriétés, notre classe « Participant » doit implémenter l'interface « INotifyPropertyChanged ».
- Le « Data binding » vers les classes du projet en utilisant l'interface « INotifyPropertyChanged » permet de refléter, immédiatement, les changements dans l'interface utilisateur dans le DataGrid.
➔ On dit qu'on utilise du « binding Bidirectionnel ».

5. Utilisation d'un « ListView » pour l'affichage des données

ListView est un contrôle d'interface utilisateur dans WPF qui permet d'afficher des données sous forme de liste. Il permet aux utilisateurs de visualiser, trier, filtrer et sélectionner des données de manière interactive.

- Ajoutez une nouvelle fenêtre WPF « ListViewUI ».
- Ajoutez un nouveau bouton nommé « btnListView » au-dessous des 3 premiers boutons.
- Sur clic sur le bouton « btnListView », la fenêtre « ListViewUI » est ouverte.
- Dans « ListViewUI.xaml » ajoutez une « ListView », que vous nommez « myListView », permettant d'afficher les données de l'ObservableCollection « participants ».
- Dans le constructeur de « ListViewUI », définissez la collection observable « participants » comme la source des items de myListView.

NB : Il faut que l'ObservableCollection « participants » soit statique.

6. Utilisation d'un DataTemplate

- Il est fréquent de vouloir afficher plus d'informations concernant un objet. Le DataGrid donne la possibilité d'afficher des détails d'une ligne juste au-dessous de cette ligne.
- Dans notre exemple, on veut afficher « Email » sur une ligne de détails. Mettez en commentaire la balise correspondante dans le DataGrid.
- Dans l'élément « DataGrid » et après la fin du sous-élément « `DataGrid.Columns` », ajoutez le sous-élément « `DataGrid.RowDetailsTemplate` » avec un « `DataTemplate` » pour afficher l'Email.

Gestion des participants

Matricule	Prénom	Nom	Genre	Niveau	Email
1	Moina	Loffhead	F	Professionnel	mloffhead0@dell.coi
2	Richard	De La Salle	M	Débutant	rdelasalle1@phoca.c
3	Deina	Clemintoni	F	Intermédiaire	dclemintoni2@indep
4	Giselbert	Wittke	M	Professionnel	gwittke3@archive.or
5	Beatrice	Obey	F	Intermédiaire	bobey4@fc2.com
6	Raphaëla	Dunston	F	Professionnel	rdunston5@ibm.con
Email: rdunston5@ibm.com					
7	Joshua	Steer	M	Professionnel	jsteer6@jigsy.com
8	Bartie	Soeiro	M	Débutant	bsoeiro7@army.mil
9	Merridie	MacIlhagga	F	Intermédiaire	mmacilhagga8@dell
10	Mickie	Walburn	F	Professionnel	mwalburn9@aol.con

Ajouter

Modifier

Supprimer

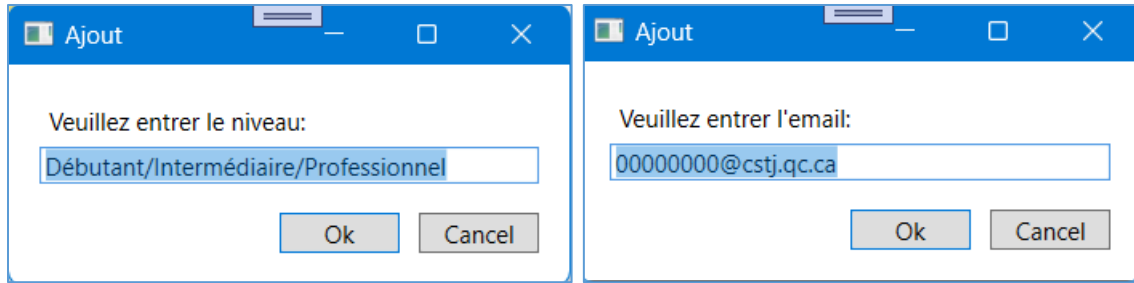
ListViewUI

7. Création de la fenêtre « InputDialog »

- On veut créer une GUI permettant à l'utilisateur de communiquer des informations à l'application pour l'ajout d'un nouveau participant. Nous allons la nommer « InputDialog ».
- Cette fenêtre devrait être **modale**, c'est-à-dire que tant qu'elle est affichée, la fenêtre principale ne sera pas accessible. Ceci est possible au moment de l'ouverture de cette fenêtre via la méthode « **ShowDialog** » (voir section suivante).
- Créez une nouvelle fenêtre WPF et nommez-la « InputDialog ».
- Un texte est affiché dans un label pour indiquer à l'utilisateur l'information demandée et une zone de texte lui permet de saisir une valeur. La fenêtre contient aussi deux boutons : un pour valider et l'autre pour annuler. Ajoutez le code XAML et le code métier nécessaires.

The figure displays four sequential screenshots of a WPF dialog box titled 'Ajout'. Each screenshot shows a different input field being focused, with 'Ok' and 'Cancel' buttons at the bottom right.

- Top Left:** The label 'Veuillez entrer le matricule:' is above a text box containing '00000000'.
- Top Right:** The label 'Veuillez entrer le prénom:' is above a text box containing 'John'.
- Bottom Left:** The label 'Veuillez entrer le nom:' is above a text box containing 'Doe'.
- Bottom Right:** The label 'Veuillez entrer le genre:' is above a text box containing 'M/F'.



8. Utilisation de « InputDialog » comme fenêtre modale

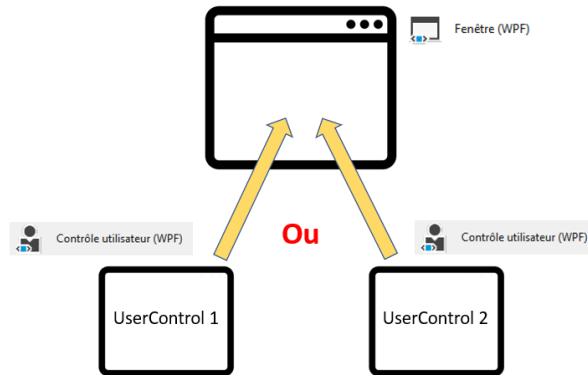
- On veut améliorer le fonctionnement du bouton « Ajouter » en permettant à l'utilisateur de saisir lui-même les données du nouvel participant à ajouter. On peut utiliser des contrôles de type « InputDialog ».
- Comme mentionné plus haut, on veut bloquer l'utilisation de la fenêtre « MainWindow » tant que la fenêtre « InputDialog » est affichée. Il suffirait d'ouvrir la fenêtre « InputDialog » en utilisant la méthode « ShowDialog ».
 - o Il faut créer un objet de type « InputDialog » en utilisant un constructeur paramétré dont le premier argument est le titre de la boîte de dialogue, le deuxième argument est le texte à y afficher et le troisième argument est le texte par défaut à y afficher.
 - o La structure « if » est nécessaire car c'est elle qui permet de vérifier que la boîte de dialogue a été affichée.

9. Utilisation de « UserControl »

- Notre utilisation actuelle d'une application à plusieurs fenêtres fait en sorte qu'on affiche une nouvelle fenêtre quand on a une autre chose à montrer, par exemple : l'appel d'un formulaire d'inscription, l'affichage d'une liste d'informations, etc.
- On peut utiliser un « UserControl » pour changer le contenu d'une fenêtre (sans devoir ouvrir une nouvelle fenêtre).
 - o UserControl est lui-même un contrôle, donc il peut être ajouté à une fenêtre.
 - o On peut lui ajouter les mêmes éléments qu'à une fenêtre.
- Ajoutez un nouveau UserControl, dans le même projet actuel :
 - o Ajouter / Contrôle utilisateur (WPF).
 - o Nommez-le « UCGestionPart ».
- On remarque que le UserControl créé est très similaire à un élément de type « Window ». Deux fichiers sont ajoutés dans le projet : « UCGestionPart.xaml » et « UCGestionPart.xaml.cs ». Toutefois, l'interface du UserControl n'a pas de barre en haut, pas de boutons pour fermer ou minimiser et on n'a que le contenu de la fenêtre.

9.1. Chargement d'un UserControl sur toute la surface d'une fenêtre

- Lorsqu'on utilise des UserControls, on doit avoir une fenêtre WPF dans laquelle on charge à chaque fois un des UserControls.
- Pour afficher un autre UserControl, il faut tout d'abord décharger le premier UserControl qui est actuellement chargé dans la fenêtre WPF.



- Dans notre exemple, la fenêtre « MainWindow » contient tous les contrôles et le code-behind. Dans la solution actuelle, la fenêtre « MainWindow » devrait contenir seulement un Panel vide dans lequel on va charger les UserControls.
- Déplacez dans « UCGestionPart.xaml » tous les contrôles qui se trouvaient dans « MainWindow.xaml ». (Il faut juste remplacer `<Window.Resources>` par `<UserControl.Resources>`).

Modifiez les dimensions de ce UserControl comme suit :

```
d:DesignHeight="300" d:DesignWidth="800"
```

- Déplacez dans « UCGestionPart.xaml.cs » le code-behind qui se trouvait dans « MainWindow.xaml.cs ».

NB : Il faut garder le constructeur dans la classe « MainWindow » avec son contenu minimal `InitializeComponent();` permettant d'initialiser les éléments dans la fenêtre.

- Dans notre exemple, Chaque UserControl chargée dans la fenêtre principale « MainWindow » doit occuper toute sa surface. Définissez une ligne et une colonne dans la grille « gPrincipal ».
- Au démarrage de l'application, « UCGestionPart » est chargé dans le Grid de « MainWindow ». Déclarez dans la classe « MainWindow » la propriété « ContenuEcran » de type « UserControl » et l'objet « GestPart » de type « UCGestionPart ».
- Dans le constructeur de « `MainWindow` », ajoutez le code permettant d'y charger « UCGestionPart » dans la case de la grille.

- Dans le constructeur de “ListViewUI”, remplacez “MainWindow” par « UCGestionPart »

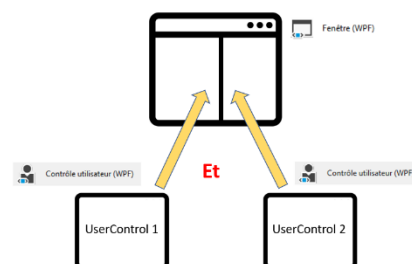
9.2. *Un deuxième UserControl*

- Créez un nouvel UserControl « UCAjoutPart ».
- En cliquant sur le bouton « Ajouter » dans le UserControl « UCGestionPart », on veut charger dans la fenêtre principale « MainWindow » un deuxième UserControl nommé « UCAjoutPart ».

- Écrivez le code XAML nécessaire pour insérer les contrôles permettant la saisie des informations d'un nouvel participant.
- Lorsque l'utilisateur clique sur le bouton « Valider », toutes les informations du nouveau participant seront sauvegardées dans la collection « participants ». Pour que cette dernière soit utilisable de « UCAjoutPart » sans avoir à instancier un nouvel objet de type « UCGestionPart », rendez-la de type « static ».
- Instanciez un objet de type « UCAjoutPart » dans le UserControl « UCGestionPart ».
- Toujours dans le code-behind de la classe « UCGestionPart », la méthode « BtnAjouterPart_Click » devrait contenir le code permettant de remplacer le contenu actuel de la fenêtre principale par le contenu de « UCAjoutPart » lorsque l'utilisateur clique sur le bouton « Ajouter ».
- Après avoir saisi les données du nouveau participant dans le UserControl « UCAjoutPart », on peut cliquer sur le bouton « Valider » qui permet d'ajouter les données dans la collection de participants, ensuite on remplace le UserControl actuel par « UCGestionPart » :

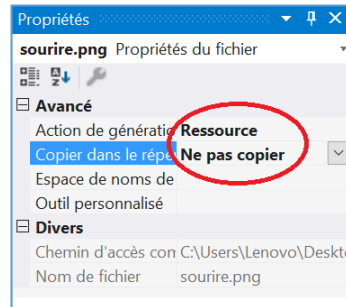
9.3. *Chargement de plusieurs UserControl dans une même fenêtre*

- Un UserControl peut occuper toute la surface d'une fenêtre ou juste une partie de la fenêtre. Quand la surface d'une fenêtre est divisée, il est possible de charger plusieurs UserControls, chacun dans une zone de la fenêtre.



- Pour tester le chargement de deux ou plusieurs UserControls au même temps dans une fenêtre, ajoutez une deuxième colonne dans « MainWindow ». La première colonne aura une largeur égale à 9/10 de la largeur totale.

- Créez un nouvel UserControl et nommez-le « UCIconPart » de hauteur 300 et de largeur 450.
- Cherchez sur Internet une icône d'un participant avec les mots clés « participant icon » et nommez-la « iconePart.png ».
- À partir de l'explorateur de solutions, ajoutez dans le projet un dossier « images » puis mettez le fichier de l'icône dans ce dossier. Dans l'explorateur de solutions, faites un clic droit sur l'image et allez dans ses propriétés : Action de génération : Ressource, copier dans le répertoire : Ne pas copier.



Si l'image n'apparaît toujours pas, allez dans le dossier du projet et supprimez les deux dossiers « bin » et « obj » puis réexécutez le programme.

- Ajoutez l'image dans « UCIconPart ».
- Dans le code-behind de « MainWindow », créez la propriété « ContenuColonne2 » de type « UserControl » et l'objet « UCIcone » de type « UCIconPart ».
- Dans le constructeur de « MainWindow », ajoutez le code permettant d'afficher le UserControl « ContenuColonne2 » dans la deuxième colonne.

