Programmation des Interfaces Interactives Avancées

Travaux pratique numéro 3 - Correction

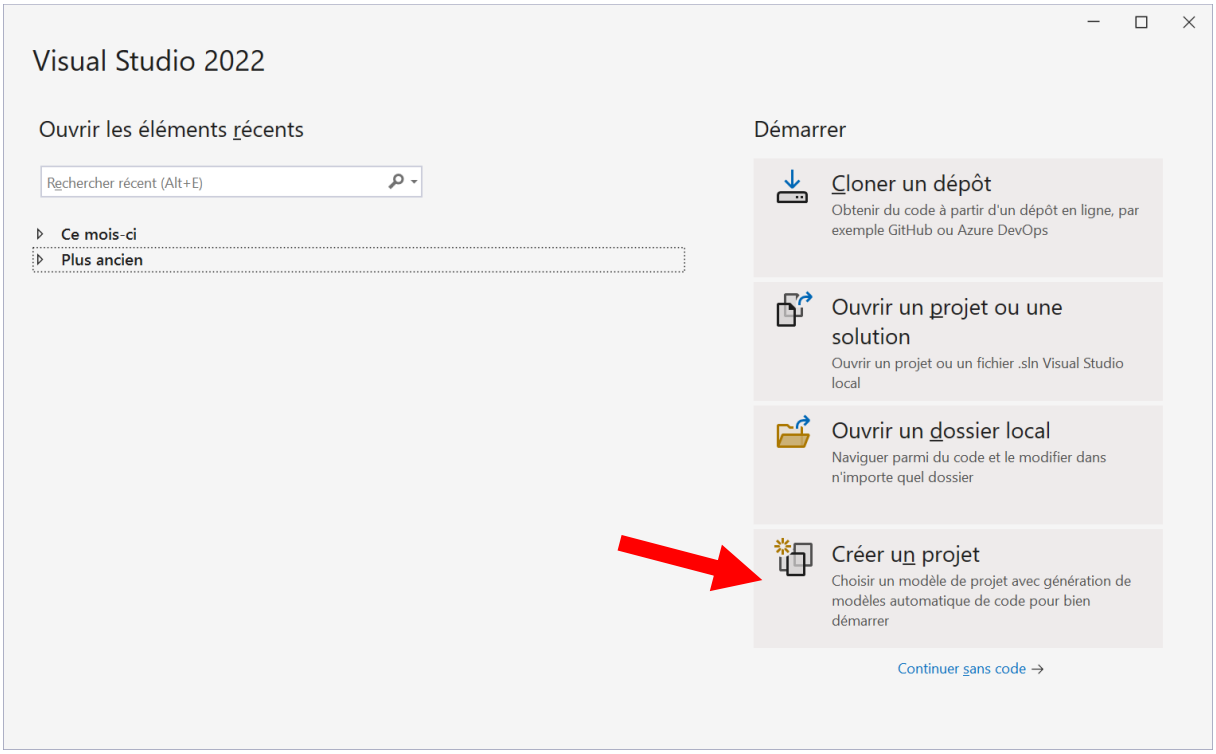
Dans ce TP et les suivants, il est fortement conseillé de consulter la documentation officielle de Visual C# pour répondre aux questions. Le site web de documentation de Visual Studio 2022 pour Visual C# est https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2022. Vous pouvez aussi consulter le navigateur des API : https://learn.microsoft.com/en-us/dotnet/api/?view=windowsdesktop-8.0 et plus spécifiquement celles du namespace System.Windows : https://learn.microsoft.com/en-us/dotnet/api/?view=windowsdesktop-8.0
Nous travaillerons avec la plateforme .NET 8.0 (Long-Term Support).

Exercice 1

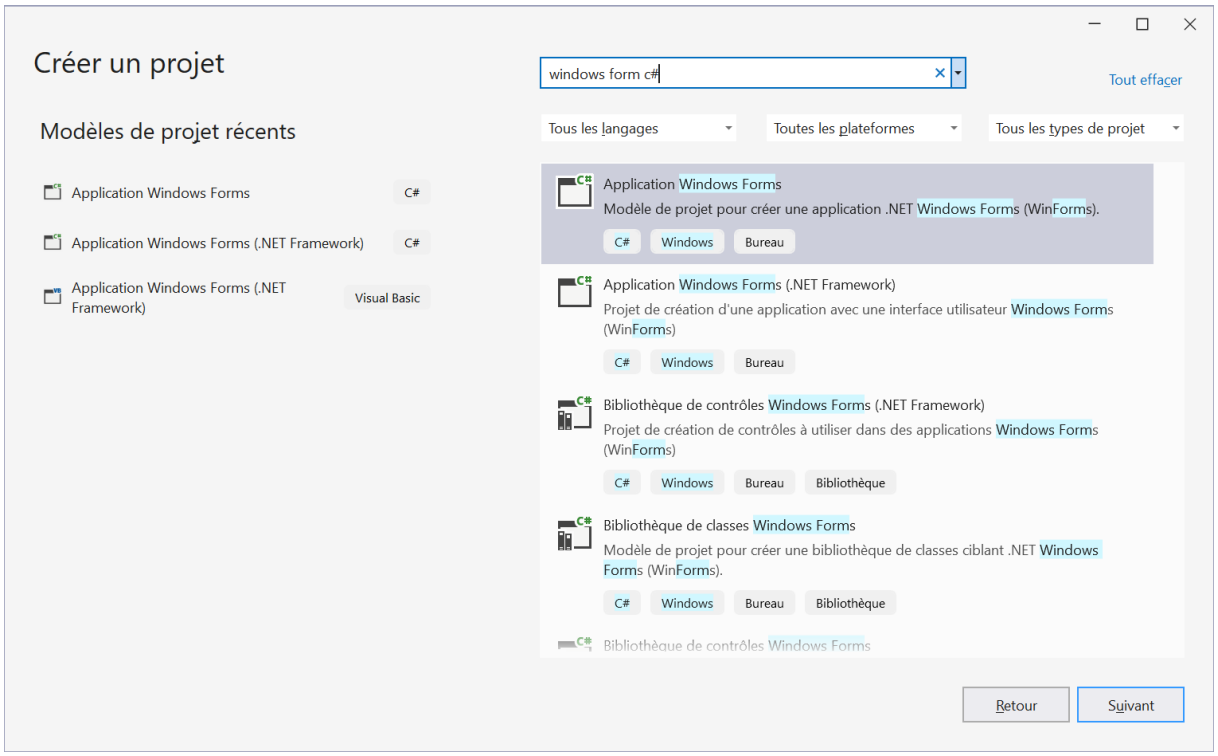
Dans cet exercice, vous serez guidé pour créer une application graphique très simple qui affiche une fenêtre avec le mot « Bonjour ».

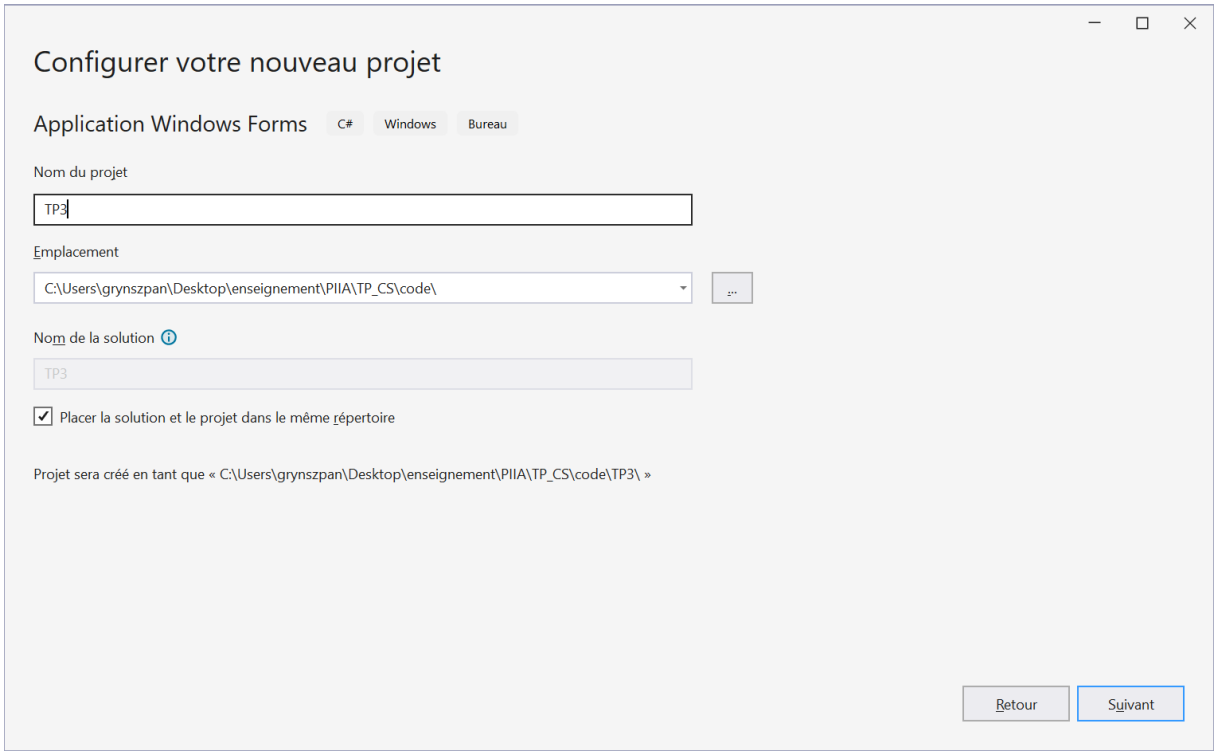
Partie A : Création d'un projet Visual C# sur Visual Studio 2022.

1) Ouvrir Visual Studio 2022 et créer un projet `TP3` en utilisant le modèle « Application Windows Forms » pour C#.

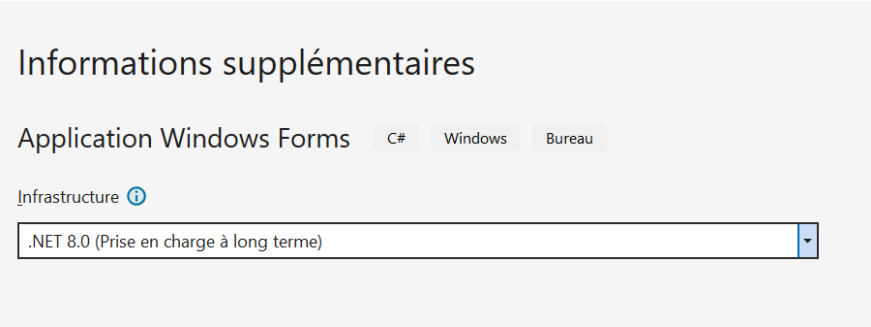


Pour trouver le modèle « Application Windows Forms », taper « windows from c# » dans l'encart de recherche.

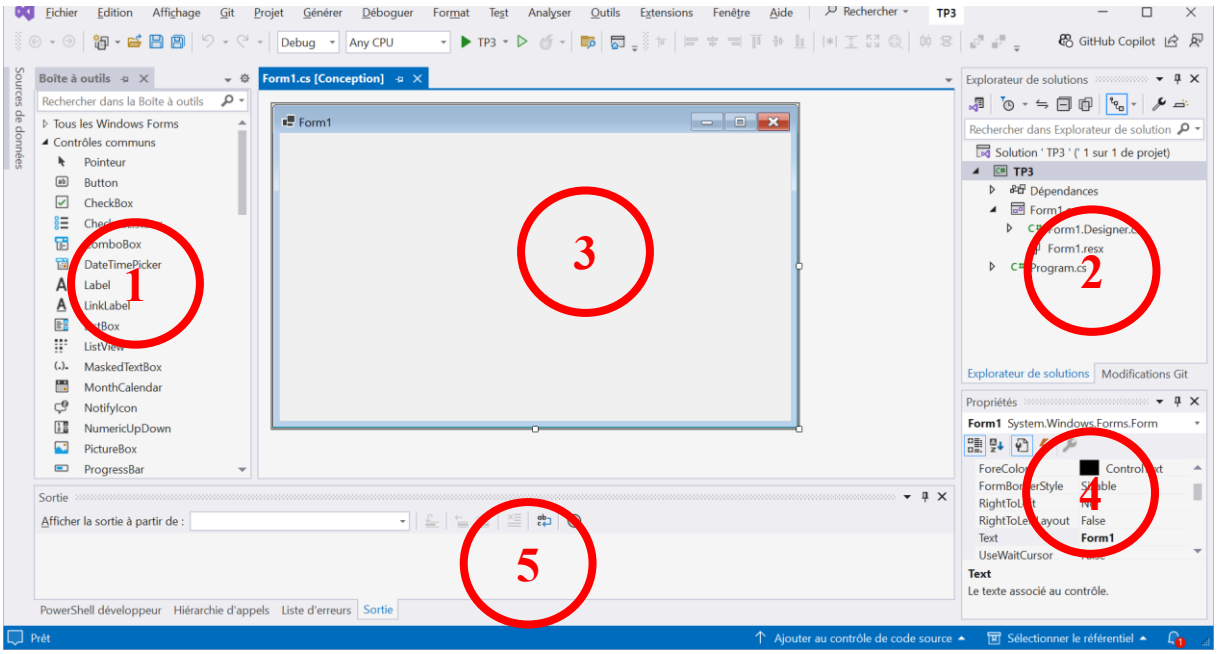




Dans la configuration, sélectionner la plateforme .NET 8.0 (Long-Term Support)



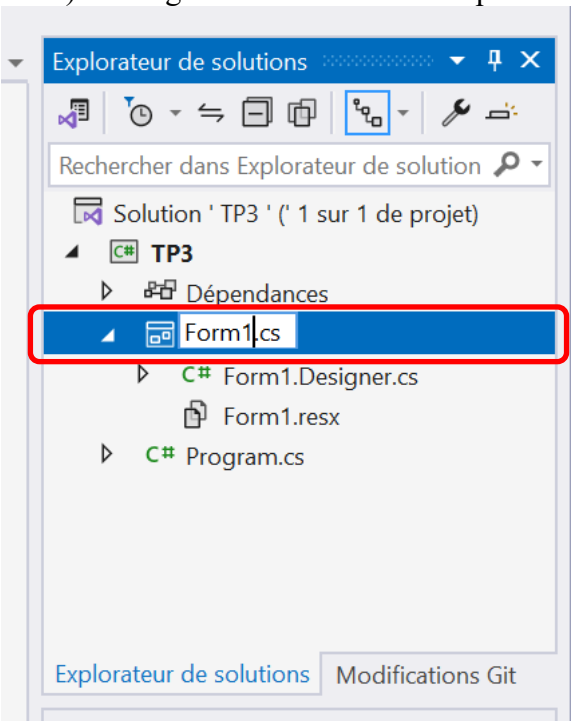
L'interface de Visual Studio :



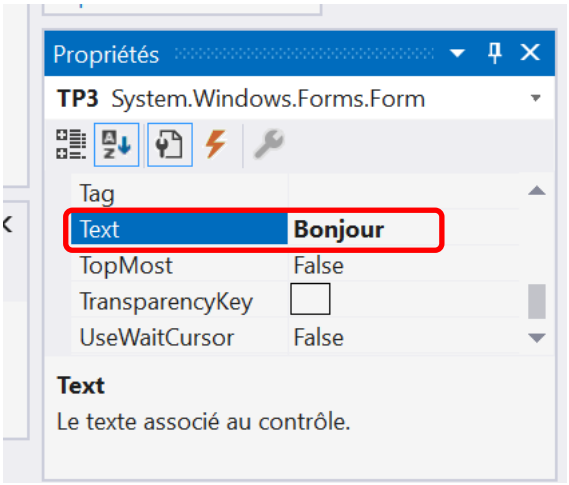
1. **La boite à outils de widgets /contrôles** : Cette boîte contient les contrôles que l'on peut faire glisser-déposer sur la fenêtre dans la vue Designer.
2. **L'arborescence du projet :** On peut voir l'ensemble des fichiers de l'application. Dans une application .Net, l'ensemble des fichiers se trouve encapsulé dans une « Solution ». Une « Solution » peut contenir plusieurs « Projets ». Chaque « Projet » d'une « Solution » peut être un exécutable (exe) ou une bibliothèque (dll). Par exemple, si on développe un jeu, on aurait une « Solution » pour ce jeu, puis on pourrait avoir à l'intérieur un « Projet » pour la bibliothèque de ressources graphiques, un « Projet » pour l'exécutable du mode solo, un « Projet » pour l'exécutable du mode multijoueur etc. C'est la façon d'organiser le code dans Visual Studio.
3. **La vue Designer** : Cette vue permet d'éditer interactivement les contrôles de l'interface que on est en train de concevoir. C'est aussi ici que s'ouvre les fichiers de code source.

4. **La vue propriété** : On peut configurer l'interface et ses contrôles dans cette vue. Lorsqu'un élément de l'interface est sélectionné dans la vue Designer, cet espace se met à jour avec l'ensemble des propriétés et des événements de l'élément sélectionné afin de pouvoir les éditer. On peut par exemple changer le texte d'un bouton ou assigner une procédure événementielle depuis cet endroit.
5. **La vue sortie** : C'est dans cette vue que le compilateur affiche les erreurs de compilation ou que le programme affiche les sorties de la console.

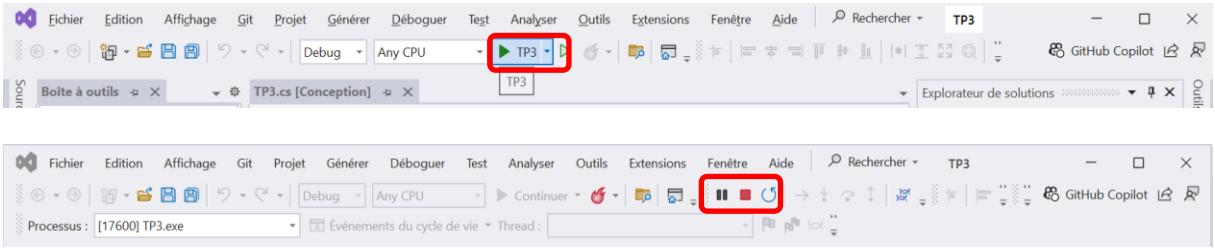
2) Changer le nom de la fenêtre par défaut Form1 → TP3



3) Changer le titre de la fenêtre TP3 dans l'onglet « Propriétés » après avoir cliqué sur la vue Designer → Mettre « Bonjour » pour la propriété Text.



4) Cliquer sur « `TP3` » pour lancer l'application. Vous ne devriez pas avoir d'erreur. Pour arrêter l'exécution, cliquer sur le carré rouge sur la barre d'outils de la console au-dessu de l'encart de code.



Partie B. Questions :

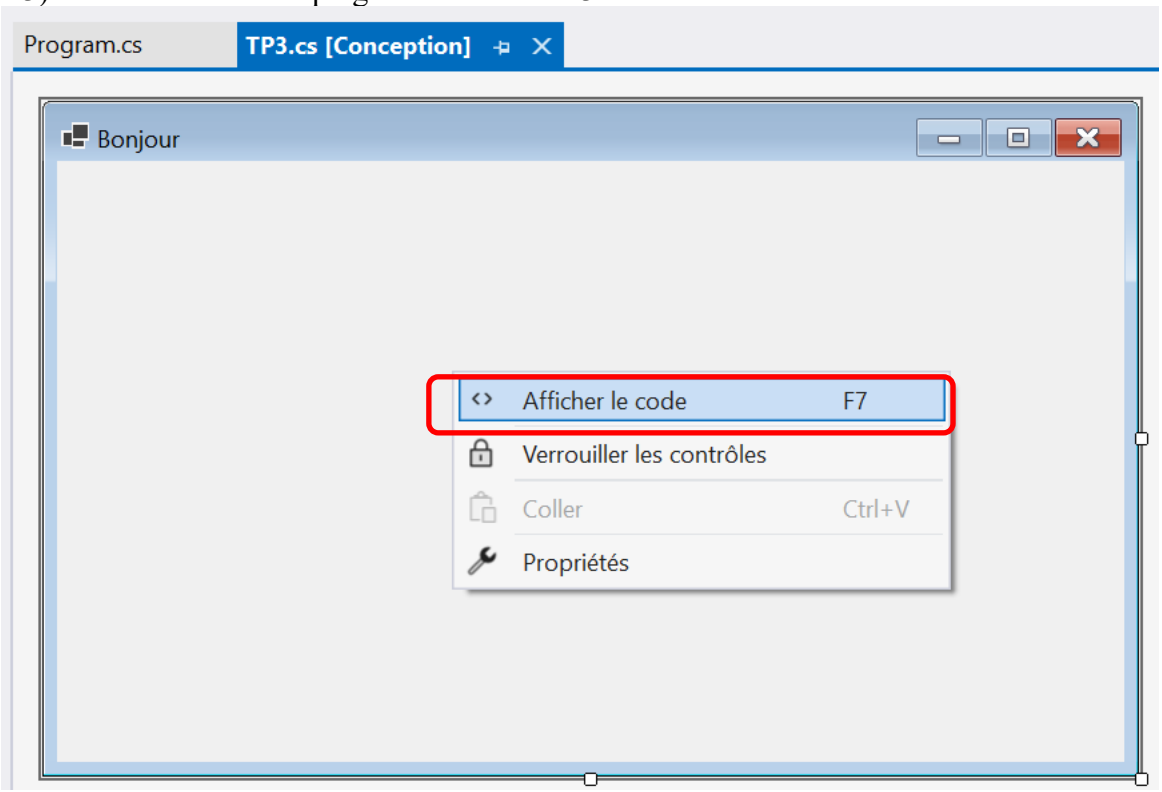
1) Aller voir le source code de la classe `Program.cs`. Quel est le type de `Application` ? A quoi sert-elle ?

Correction : Elle est de type System.Windows, Forms.Application. Elle fournit des méthodes statiques pour lancer et arrêter des fenêtres graphiques.

2) Quel est le type de `Application.Run()` ? A quoi sert-elle ? Quand vous exécutez ce code, que voyez-vous ?

Correction : `Application.Run()` est une méthode qui permet de lancer une fenêtre. Quand elle s'exécute, on voit la fenêtre apparaître.

3) Aller voir le code qui gère la fenêtre TP3



4) Ajouter le code ci-dessous dans la classe "public partial class TP3 : Form"

```
namespace TP3
{
    public partial class TP3 : Form
    {
        public TP3()
        {
            InitializeComponent();
            AddLabel();
        }

        private void AddLabel()
        {
            // Create a new Label
            Label myLabel = new Label();

            // Set Label properties
            myLabel.Text = "Bonjour, c'est un label !";
            myLabel.Font = new System.Drawing.Font("Arial", 10,
System.Drawing.FontStyle.Bold);
            myLabel.AutoSize = true;
            myLabel.Location = new System.Drawing.Point(this.Width/3, this.Height/2);

            // Add Label to the Form
            this.Controls.Add(myLabel);
        }

    }
}
```

5) Quel est le type de `Form` ? A quoi sert-il ?

Correction : Il est de type `System.Windows.Forms.Form`. Il permet de définir la vue d'une fenêtre.

6) Expliquer l'objectif du mot clé `partial` dans la définition de la classe "`public partial class TP3 : Form`"

Correction : Ce mot clé permet de répartir la définition d'une classe sur plusieurs fichiers.

7) A partir votre code de la classe "`public partial class TP3 : Form`", aller voir le contenu de la méthode `InitializeComponent()` sans la modifier.

Correction : Click droit puis « Atteindre la définition »

Quelles sont les propriétés qui sont définies par défaut dans cette méthode ?

Correction : AutoScaleDimensions, AutoScaleMode, ClientSize, Name, Text

Aller consulter d'autres fonctionnalités de cette méthode sur la documentation de Visual C# .NET 8.0.

Correction : voir https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.form?view=windowsdesktop-8.0

8) Afficher le texte "Ceci est un texte supplémentaire au-dessus de l'étiquette" au-dessus de l'étiquette `myLabel` chaque fois que nous plaçons le curseur au-dessus de celle-ci. Aide : Il faut utiliser un objet de type ToolTip.

Correction : Ajouter le code ci-dessous dans la méthode AddLabel() :

```
ToolTip myToolTip = new ToolTip();
myToolTip.SetToolTip(myLabel, "Ceci est un texte
supplémentaire au-dessus de l'étiquette");
```

Exercice 2

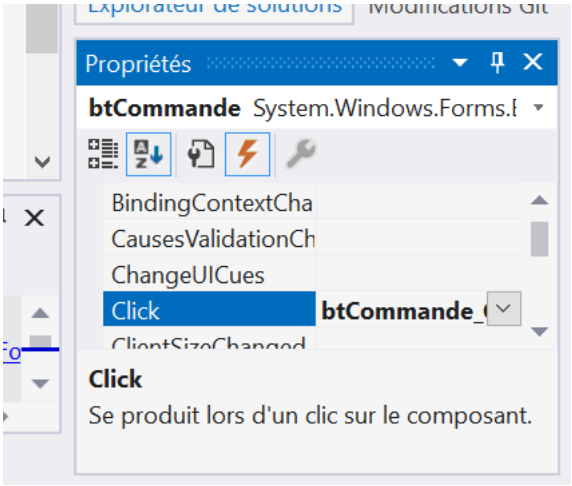
Dans cet exercice, vous allez créer une application simple avec les éléments graphiques pour déclencher des événements et collecter des inputs de l'utilisateur.

1) Faire glisser-déposer un bouton (« `Button` ») de la « **Boîte à outils** » vers la fenêtre dans la vue **[Conception]**. Dans la vue **Propriété**, on peut changer sa propriété « `Name` » afin de remplacer « button1 » par « btnCommande ». Profitez-en pour remplacer sa propriété « `Text` » de « button1 » à « Cliquez moi ! »

2) Depuis la **boîte à outils**, faire glisser un « `Label` » à droite de notre bouton. Changer sa propriété « Name » pour respecter les conventions. On peut l'appeler « lblAffiche » par exemple. Effacer le texte par défaut de sa propriété « `Text` ». Si jamais sa propriété « `AutoSize` » était à « True » alors ce « `Label` » s'est surement redimensionné en fonction de la taille du texte qu'il doit afficher. Comme on vient d'effacer le texte, le contrôle peut être devenu trop petit pour être sélectionné à la souris depuis la vue **Conception**.

Dans la vue **Conception**, faire un double-clic sur le bouton. Visual Studio va ouvrir automatiquement le fichier « TP3.cs » et va ajouter le code d'une procédure événementielle minimale qui devrait s'appeler ici « `btnCommande_Click` ».

Si on regarde dans les propriétés événementielles du bouton, on devrait voir que Visual Studio a également déjà associé l'événement « Click » du bouton à notre procédure événementielle qui vient d'être créée.

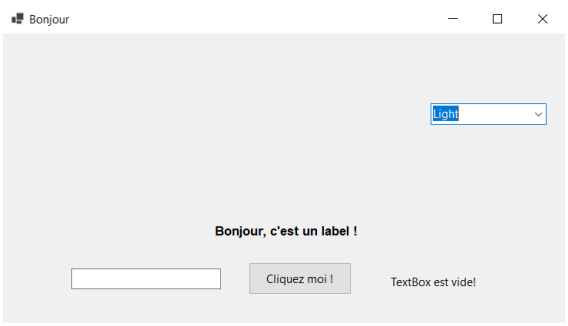
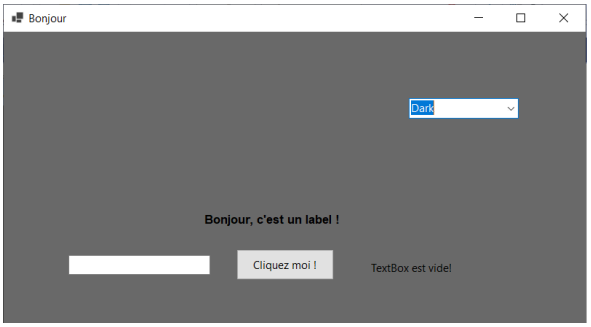


En réalité, tous les changements de l'interface que nous faisons dans la vue **Conception** et dans la vue **Propriétés** sont répliqués par Visual Studio dans le fichier « TP3.Designer.cs ».

Retourner maintenant dans le code de la procédure événementielle et faire en sorte que lorsque l'on clique sur le bouton, le label change sa propriété « Text » pour afficher le message : « Vous avez cliqué. » par la commande :

```
lblAffiche.Text = " Vous avez cliqué." ;
```

Sauver et relancer l'application, si on clique sur le bouton, la procédure événementielle est appelée ce qui a pour effet de changer le texte du label.

3) Découvrir d'autres outils de la boîte. Par exemple, en utilisant un `TextBox` pour récupérer le contenu de sa propriété « Text » et l'afficher sur le label quand l'utilisateur clique sur le bouton. Si le TextBox est vide, afficher un message pour l'indiquer. Ajouter un `ComboBox` pour proposer des choix à l'utilisateur pour la couleur du fond de la fenêtre.

Aide : Pour ComboBox, regarder la propriété Items pour définir une liste de choix possibles.