1. Un formulaire

Propriété

(Name)	frmPrincipal
AutoScroll	True – False : Affichage des barres de défilement
icon	… image dans le coin supérieur gauche
text	Titre de la fenêtre
FormBorderStyle	FixedSingle, Sizable (l'utilisateur peut la modifier) none (pas de bordure mais attention le menu système disparait aussi)
ControlBox	True – False détermine si la fenêtre possède le menu système (fermer agrandir …)
MaximizeBox	True – False détermine si la fenêtre possède le bouton agrandir
MinimizeBox	True – False détermine si la fenêtre possède le bouton réduire
Size	Taille de la fenêtre : 800;600
ShowInTaskbar	True – False : Affichage dans la barre de tâche (permet de fermer le programme en cas de blocage par exemple)
WindowsState	Dimension initiale : Normal - Minimized - Maximized
StartPosition	Définit la position initiale de la fenêtre Manuel : Taille et l'emplacement sont déterminés par l'application. CenterScreen : La taille est fixée par l'application. L'emplacement est le centre de l'écran. WindowsDefaultLocation – La taille est déterminée par l'application. L'emplacement est déterminé par le système d'exploitation. WindowsDefaultBounds - La taille et l'emplacement sont déterminés par le système d'exploitation. CenterParent - ne fonctionne que pour les formulaires possédant un parent. Positionne le formulaire au centre de son parent. Si on utilise Manuel à défaut d'indication de position, le formulaire sera placé dans le coin supérieur gauche La position de départ doit être renseignée dans le constructeur (nécessiste using System.Drawing;) this.Location = new Point((Screen.PrimaryScreen.WorkingArea.Width - this.Width) / 2, (Screen.PrimaryScreen.WorkingArea.Height - this.Height) / 2);

Méthode

Show()	Afficher le formulaire
Hide()	Masque le formulaire qui pourra alors être ouvert de nouveau par Show
Close()	Ferme le formulaire qui ne pourra plus être ouvert par Show

Événement

Load	Au chargement du formulaire
FormClosing	Avant la fermeture du formulaire : cela permet d'interdire la fermeture en définissant la propriété Cancel de l'événement CancelEventArgs transmis à votre gestionnaire d'événement sur true private void FrmBase_FormClosing(object sender, FormClosingEventArgs e) { e.Cancel = true; }

Rappel : pour quitter une application : Application.Exit(0);

1.1. Ouverture et fermeture des instances de formulaires

Lorsqu'une application utilise plusieurs formulaires, il faut garder à l'esprit que la fermeture du premier formulaire entraine la fermeture de l'application. Par conséquent lorsqu'on veur changer de formulaire le premier formulaire ne doit pas être fermé (.close) mais masqué (.hide) afin de rester en mémoire.

Il est parfois nécessaire lorsque les formulaires ont une organisation arborescente de revenir automatiquement sur le formulaire de niveau supérieur (le père) lorsqu'on ferme le formulaire courant.
Pour cela on définit une propriété automatique FormulaireParent au niveau du formulaire afin de mémoriser la référence du formulaire parent :

```
public partial class Form2 : Form     {
    public Form FormulaireParent { get; set; }
    …
}
```

Lorsqu'on ouvre le formulaire fen2 (instance de la classe Form2) à partir du formulaire fen1

```
this.Hide();
Form2 fen2 = new Form2();
fen2.FormulaireParent = this;
fen2.Show();
```

Lorsqu'on ferme le formulaire fen2 (événement FormClosed) on ouvre (rend visible) l'instance mémorisée dans la propriété FormuliareParent :

```
this.FormulaireParent.Show();
```

Bien évidemment avec un seul niveau de profondeur la propriété FormulaireParent n'est pas à placer au niveau de chaque formulaire enfant puisqu'on doit toujours revenir sur le formulaire principal (le premier ouvert). Dans ce cas on stocke l'adresse du premier formulaire dans une constante accessible dans toute l'application.

Attention : La fermeture d'un formulaire peut se réaliser à l'aide de la méthode Close() mais aussi en cliquant sur la croix en haut à droite de la fenêtre si la propriété ControlBox du formulaire possède la valeur 'true'. Dans tous les cas c'est l'événement FormClosing qui se produit et c'est donc a ce niveau qu'il faut programmer la réouverture du formulaire.

1.2. Application de l'héritage sur les formulaires

Tous les formulaires dérive de la classe Form. Il est possible d'utiliser de concevoir un formulaire de base et de faire dériver tous les formulaires utilisés par l'application de ce formulaire. Cela permet de définir un comportement commun pour chaque formulaire (le menu général, l'entête, le pied, la taille, le mode d'ouverture, la mise en forme etc.)

```
public partial class FrmBase : Form
```

Il faut ajouter dans l'événement load : `if (DesignMode) return;`
Le formulaire de base doit pouvoir être instanciée pour permettre à Visual Studio d'afficher en mode design les formulaires.

2. Propriété commune des contrôles

Propriété

Name	Nom de contrôle
Dock	Permet de définir les bordures du contrôle liées avec celles du formulaire
Font	Définition de la police de caractère (hérité du conteneur)
Size	Taille : (largeur;hauteur)
Enabled	True – False : le composant est accessible ou pas
ContextMenuStrip	Menu contextuel associé au composant

3. Les composants standards

Une étiquette (label : lblNom)

Text	Contenu affiché
autoSize	false (pour un label le label sinon il est réduit au maximum s'il ne contient pas de valeur) true pour un panel afin qu'il s'adapte à son contenu
BorderStyle	3D, fixedSingle, none
Enable	False

Un champ de type texte (textBox : txtNom)

Propriété

Text	Valeur affichée
Multiligne	Sur plusieurs lignes
AcceptsReturn	Accepte la touche Entrée dans le champ
AcceptsTab	Accepte la tocuhe Tab dans le champ
BorderStyle	3D

Événement

`KeyUp`	Permet par exemple de déclencher un traitement après l'appui sur la touche Entrée : `if (e.KeyCode == Keys.Return)`

Un bouton (Button : btnNom)

Text	texte affiché
TextAlign	MiddleRight
BackGroundImage	
ImageAlign	MiddleLeft
FlatStyle	Standard – Flat (permet de définir la bordure avec FlatAppearance)
FlatAppaerance	paramétre : BorderColor, BorderStyle,

Une case à cocher : (checkbox : caseNom) ou bouton radio (radiobutton): rdbNom

Checked	retourne true si la case est cochée

Un texte enrichi (RichTextBox : rtbNom)

Text	texte associé
ContextMenuStrip	menu contextuel associé
AppendText(string)	Ajoute du texte dans le composant
Clear()	Efface tout le contenu
Lines	Collection des lignes du composant (string)

Un champ de type DateTime associé à un calendrier (DateTimePicker : dtpNom)

CustomFormat	Définit le format d'affichage, par défaut le format est MM/DD/YYYY
MinDate	Obtient ou définit la date et l'heure minimales pouvant être sélectionnées dans le contrôle.
MaxDate	Obtient ou définit les date et heure maximales pouvant être sélectionnées dans le contrôle.
Text	Obtient ou définit le texte associé à ce contrôle.

```
dtpUneDate.Format = DateTimePickerFormat.Custom;
dtpUneDate.CustomFormat = "yyyy MMMM dd";
dtpUneDate.MinDate =
```

Une boite de dialogue pour l'ouverture d'un fichier (OpenFileDialog : bteNom)

Propriété

Name	bteCharger
Filter	(Fichier C#|*.cs)
MultiSelect	False
FileName	Nom par défaut proposé
InitialDirectory	Répertoire de recherche initial
Title	Titre de la boîte

Événement

FileOk	Après le clic sur le bouton Ok de la boîte

```
private void optionOuvrir_Click(object sender, EventArgs e)      {
   bteCharger.Filter = "Fichier txt|*.txt|fichier cs|*.cs";
   bteCharger.InitialDirectory = Application.StartupPath;
   bteCharger.Multiselect = false;
   bteCharger.ShowDialog();
}
private void bteCharger_FileOk(object sender, CancelEventArgs e)      {
   chargerFichier(bteCharger.FileName);
}
```

Une boîte de dialogue pour enregistrer dans un fichier (SaveFileDialog : bteNom)

Propriété

Name	bteEnregistrer
Filter	(Fichier C#|*.cs)
DefaultExt	Cs
FileName	Vide initialement
Title	Enregistrer le fichier généré

Événement

FileOk	Oui (à traiter après la mise de place de tous les composants)

4. Une boîte de dialogue standard : la classe MessageBox

Une boîte de dialogue affiche un message et permet éventuellement à l'utilisateur de choisir une option en cliquant sur une des boutons affichés. Par défaut, une boîte de message simple dispose d'un bouton OK et affiche un message.
Il est possible d'ajouter un titre, une image et de proposer d'autres boutons telles que Oui, Non et Annuler.

La classe MessageBox est défini dans l'espace de noms System.Windows.
La méthode statique Show permet d'afficher la boîte
Cette méthode retourne une énumération MessageBoxResult qui a la valeur Aucun, OK, Annuler, Oui et Non. Cela permet de savoir sur quel bouton l'utilisateur a cliqué.

Pour un message simple sans récupération de la valeur retournée :

```
MessageBox .Show ( "Action réalisée avec succès");
```

Si on veut ajouter un titre pour la fenêtre

```
MessageBox .Show ( "Action réalisée avec succès", "Titre de la fenêtre");
```

Choix des boutons affichés

L'énumération MessageBoxButtons est chargée de montrer différents boutons de la boîte de dialogue.

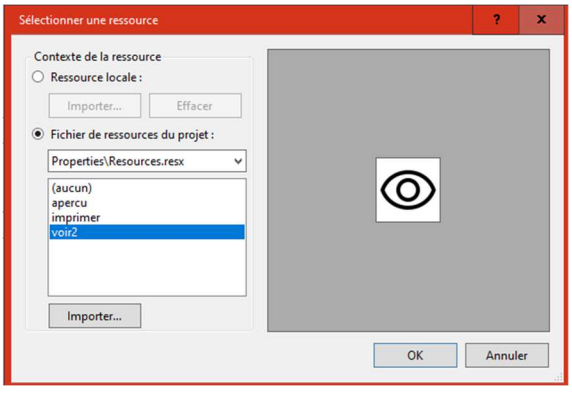
```
if (MessageBox.Show("Confirmez-vous la suppression de ce rendez -vous ?", "Confirmation",
MessageBoxButtons.YesNo) == DialogResult.Yes)        { …  }
```

Il est possible de placer une icône liée au type de message.
L'énumération MessageBoxImage représente une icône. Elle possède les valeurs suivantes :

None, Hand (main), Question, Exclamation, Asterisk, Stop, Error, Warning, Information

```
string message = "Attention les données vont être perdues \n Confirmez-vous la suppression";
string titre = "Confirmation";
MessageBoxButtons bouton = MessageBoxButtons.YesNo;
MessageBoxIcon image = MessageBoxIcon.Warning;
if (MessageBox.Show(message, titre, bouton, image) == DialogResult.Yes) { …  }
```

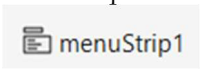
5. Intégration d'une image



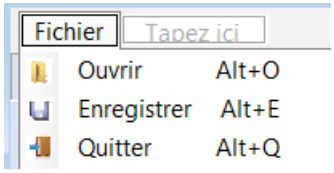
Une image peut être utilisé en arrière-plan (backgroundImage) avec une disposition donnée (BackgroundimageLayout) ou directement dans la propriété image avec un alignement défini dans la propriété ImageAlign
L'image une fois sélectionnée est intégrée au projet, on la retrouve dans la branche ressource du projet

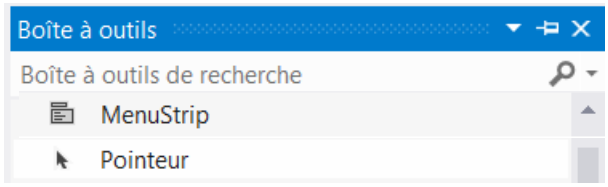
6. Un menu : Le composant Menu Strip

Sélectionner le composant : une barre est automatiquement ajouté au formulaire

Le composant non visuel apparaît en bas : 

Placer l'ensemble des contrôles



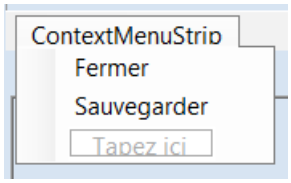


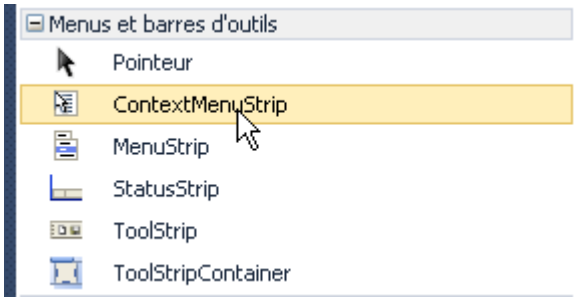
MenuStrip	
Name	menuConversion

Menu Fichier	
text	Fichier
option Ouvrir du menu Fichier	
Name	optionOuvrir
Image	… (On profite pour importer toutes les images)
ShortcutKeys	Alt+O Alt+E ALT+F
Text	reprend automatiquement le nom saisi lors de la conception
ToolTipText	Ouvrir le fichier cs à traiter - Enregistrer le fichier sous un autre nom - fermer l'application

7. Un menu Contextuel

Sélectionner le composant ContextMenuStrip dans la Catégorie Menus et barres d'outils de la boite à outils.





Name	menuCtx

Pour associer un menu contextuel à un composant de l'interface graphique, il faut renseigner la propriété ContextMenuStrip de ce composant