Le composant ComboBox

1. Présentation

Le composant représente une zone de texte lié à une zone de liste.
La zone de liste peut contenir des éléments de tout type (entier, chaîne, date, objet).
Dans un ComboBox la zone de liste est déroulante contrairement à une ListBox.

2. Les principaux membres

Propriétés et méthodes	Type	Rôle
ResetText	void	Réinitialise la valeur par défaut de la propriété Text
SelectedIndex	int	Obtient ou définit l'index spécifiant l'élément sélectionné.
SelectedItem	object	Obtient ou définit l'élément actuellement sélectionné
SelectedValue	object	Obtient ou définit la valeur de la propriété ValueMember
Text	string	Obtient ou définit le texte associé à ce contrôle
DataSource	object	Obtient ou définit la source de données (le conteneur, une collection avec des propriétés le plus souvent)
DisplayMember	string	Obtient ou définit la propriété de l'élément qui sera affichée par la liste
ValueMember	string	Obtient ou définit la propriété de l'élément qui sera utilisée pour alimenter la propriété SelectedValue du composant
Items		Retourne la collection des éléments composant la zone de liste
Items.Count	int	Retourne le nombre d'éléments dans la liste
Items.Add(element)	void	Ajoute un élément dans la zone de liste
Items.AddRange(object[])	void	Ajouter plusieurs éléments dans la liste
Items.Contains(element)	bool	Retourne true l'élément se trouve dans la zone de liste
Items.Remove(element)	void	Supprime l'élément spécifié
Items.RemoveAt(index)	void	Supprime l'élément avec le numéro d'index spécifié
Items.Clear()	void	Supprime tous les éléments de la liste
Sort()	void	Tri les éléments de la liste. Si l'élément est un objet il doit définir une méthode de comparaison (interface IComparable)

3. Alimentation de la zone de liste

Alimentation ligne par ligne

```
cbxA.Items.Clear();
cbxA.Items.Add(1);
cbxA.Items.Add(2);
…
cbxA.ResetText();  // si on ne veut pas sélectionner un élément par défaut
cbxA.SelectedIndex = 0 // si on veut sélectionner le premier élément
```

Alimentation à l'aide d'un tableau d'objets

```
lesLignes = new object[5];
for (int i = 0; i < 5; i++)
   lesLignes[i] = (object) i;
cbxA.Items.AddRange(lesLignes);
```

Alimentation à partir d'une collection

```
List<Element> lesElements = mesDonnees.getLesElements();
cbxA.DataSource = lesElements;
cbxA.DisplayMember="Libelle";  // nom d'une propriété de la classe Element
cbxA.ValueMember = "Id"; // nom d'une propriété de la classe Element
```

La classe Element doit posséder des propriétés pour pouvoir alimenter les propriétés DisplayMember et ValueMember du composant ComboBox.

```
class Element    {
      private string id;
      private string libelle;

      public Element(string id, string nom, ) {… }

      // propriété utile pour la zone de liste
      public string Id { get => id;  }
      public string Libelle { get => libelle;  }
   }
```

La propriété ValueMember perd de son intérêt ici cat il devient possible de récupérer directement l'objet sélectionné par la propriété selectedItem et ainsi accéder à l'ensemble de ses propriétés ou méthodes

```
Element monElement = (Element) cbxA.SelectedItem
```

En l'absence d'indication dans les propriétés DisplayMember et ValueMember c'est la méthode ToString qui sera utilisée pour afficher et retourner la valeur.

Si seule la méthode ValueMember est renseignée, la liste affiche aussi la valeur de cette propriétés.

Pour pouvoir utiliser la même collection dans la propriété dataSource de deux combobox différents, il faut créer un nouveau contexte sinon les deux listes sont synchronisées :

```
cbxA.DataSource = lesElements;
cbxB.BindingContext = new BindingContext();
cbxB.DataSource = lesElements ;
```

source : https://www.akadia.com/services/dotnet_unshare_datasource.html

Si les données qui doivent servir à alimenter la zone de liste proviennent d'un dictionnaire, il suffit de créer une collection à partir de la propriété Values du dictionnaire :

```
cbxA.DataSource = new List<Famille>(Globale.db.LesFamilles.Values);
```

4. Exemple d'opérations sur une zone de liste déroulante

Retourner l'index de l'élément sélectionné	cbxA.SelectedIndex;
Retourner l'élément sélectionné	Element unElt = (Element) cbxA.SelectedItem
Retourner la propriété Id de l'élément sélectionné	((Element) cbxA.SelectedItem).Id; cbxA.SelectedValue; *// si ValueMember = "Id";*
Retourner la propriété Libelle de l'élément sélectionné	((Element) cbxA.SelectedItem).Libelle ;
Supprimer l'élément sélectionné	cbxA.Items.Remove(cbxA.SelectedItem);
Supprimer l'élément sélectionné	cbxA.Items.RemoveAt(cbxA.SelectedIndex);
Sélectionner un élément	cbxA. SelectedIndex = n; cbxA. SelectedItem = unElement;

Ne pas sélectionner d'élément	cbxA.SelectedItem = null;
	cbxA.SelectedIndex = 0;
	cbxA.ResetText();

5. Contrôle sur une zone de liste déroulante.

Il est possible d'effacer la zone de texte de la zone de liste ou d'y placer une valeur qui ne se trouve pas dans la liste déroulante.

Si on ne doit pas accepter cela, il suffit de regarder la valeur de la propriété SelectedItem qui doit être supérieure à 0.

Le contrôle peut se faire lors de la validation des données du formulaire mais cela amène à devoir afficher le message d'erreur sur le formulaire (l'affichage dans une fenêtre modale n'est pas judicieux quand il y a plusieurs données à contrôler)

Il est possible d'intervenir dès la fin de la saisie par l'utilisateur en intervenant au niveau de l'événement Leave (lorsque l'utilisateur quitte le contrôle)

```
private void cbxPraticien_Leave(object sender, EventArgs e) {
   if (cbxPraticien.SelectedIndex == -1){
      MessageBox.Show("Vous devez sélectionner un praticien dans la liste");
      cbxPraticien.Focus();
   }
}
```

6. Conflit entre la propriété DataSource et l'événement SelectedIndexChanged

Initialiser la propriété DataSource déclenche l'événement SelectedIndexChanged ci ce dernier est activé hors à ce moment-là, la zone de liste est vide et la tentative de travailler sur l'élément sélectionné échoue. Il faut donc que la propriété DataSource soit alimentée avant que l'événement ne soit activé.

```
cbxPraticien.DataSource = "…";
this.cbxPraticien.SelectedIndexChanged += new
System.EventHandler(this.cbxPraticien_SelectedIndexChanged);
```

Si l'événement est déjà activé il faut le désactiver

```
this.cbxPraticien.SelectedIndexChanged -= this.cbxPraticien_SelectedIndexChanged;
```