LABEL OU TEXT BROWSER ?

Le `label`, comme le `textBrowser` permet l'affichage seulement. Aucune entrée de donnée n'est permise dans ces objets.

`Label` : Permet plusieurs lignes de texte mais ne gère pas automatique la possibilité de voir le texte qui déborde de la dimension du label.

`Text Browser` : Permet plusieurs lignes de texte. Si le texte déborde la capacité du `textBrowser`, une barre de défilement verticale apparait automatiquement.

Pour un meilleur contrôle de l'affichage et de la mise en forme de son contenu, il faut attribuer au `textBrowser` la police de caractère (font) **`Courier New`**

Méthodes de propriété souvent utilisées pour ces objets :

`.setText("Chaine")` pour alimenter l'objet

`.Text()` pour récupérer le texte contenu dans l'objet

LINE EDIT OU TEXT EDIT ?

Le `lineEdit`, comme le `textEdit`, permet l'affichage **et** l'entrée de donnée.

`Line Edit` : Est limité à une seule ligne de texte.

`Text Edit` : Permet plusieurs lignes de texte. Si le texte déborde la capacité du `textBrowser`, une barre de défilement verticale apparait automatiquement. Utile pour un attribut/champs de type *Commentaire* ou *Remarque*.

Méthodes de propriété souvent utilisées pour ces objets :

`.setText("Chaine")` pour alimenter un `lineEdit` ou un `textEdit`

`.Text()` pour récupérer le texte contenu dans un `lineEdit`

`.toPlainText()` pour récupérer le texte contenu dans un `textEdit`

LES DIMENSIONS ET LES COORDONNÉES D'UN OBJET

En cour d'exécution, il peut être utile de modifier les dimensions d'un objet, comme le formulaire lui-même, ou encore de modifier l'emplacement d'un objet à partir de ses coordonnées. Cette technique de déplacement d'objets dans une section cachée du formulaire est souvent utilisée quand il y a un seul formulaire dans l'application.

Redimensionner un formulaire

Il est bien sûr facile de dimensionner un formulaire à partir de Qt Designer. Mais, une fois en cour d'exécution, il peut être nécessaire de modifier ses dimensions pour une situation quelconque. La méthode de propriété à utiliser est **.setFixedSize(*Largeur, Hauteur*)**

Exemple :
```
self.setFixedSize(585, 375)
```

Un des avantages de cette méthode de propriété est qu'elle rend le formulaire non-dimensionnable par l'usager.

Redimensionner et repositionner un objet

Les dimensions et le positionnement d'un objet font partie de la même propriété, soit **geometry**. On pourra donc modifier la dimension et le positionnement d'un objet avec la même méthode de propriété.

Exemple :
```
self.gbxMenu.setGeometry(QtCore.QRect(125, 70, 331, 291))
```

Dans cet exemple, les coordonnées de l'objet sont `x=125; y=70`, et ses dimensions sont `Width=331; Height=291`.

Il est aussi possible de modifier l'un des deux groupes de données géométriques d'un objet :

.resize(*Lageur, Hauteur*) : Permet de modifier la dimension de l'objet.

Exemple :
```
self.gbxMenu.resize(331, 291)
```

.move(*x, y*) : Permet de modifier les coordonnées x et y de l'objet. Utile pour déplacer l'objet dans le formulaire selon les besoins.

Exemple :
```
self.gbxMenu.move(125, 70)
```

L'ORDRE DE TABULATION

L'ordre de tabulation est l'ordre de déplacement du curseur d'un objet à l'autre, quand on appuie sur la touche *TAB* du clavier. C'est pratique quand l'usager a à entrer des données dans plusieurs boites de texte. Ça facilite le déplacement d'un objet à l'autre.

Par défaut, l'ordre de tabulation est établi selon l'ordre de création des objets dans Qt Designer. Si, pour une raison quelconque, cet ordre doit être modifié, il suffit de suivre cette démarche :

Pour entrer en mode d'édition de l'ordre de tabulation, ouvrez le menu *Edit* et sélectionnez *Edit Tab Order*. Dans ce mode, chaque widget d'entrée du formulaire est affiché avec un numéro indiquant sa position dans l'ordre de tabulation. Ainsi, si l'utilisateur donne le focus d'entrée au premier widget d'entrée, puis appuie sur la touche de tabulation, le focus se déplacera sur le second widget d'entrée, et ainsi de suite.

L'ordre de tabulation est défini en cliquant sur chacun des nombres dans le bon ordre. Le premier nombre sur lequel vous cliquez deviendra rouge, indiquant la position actuellement modifiée dans la chaîne d'ordre de tabulation. Le widget associé au numéro deviendra le premier de la chaîne d'ordre de tabulation. Cliquer sur un autre widget en fera le deuxième dans l'ordre de tabulation, et ainsi de suite.

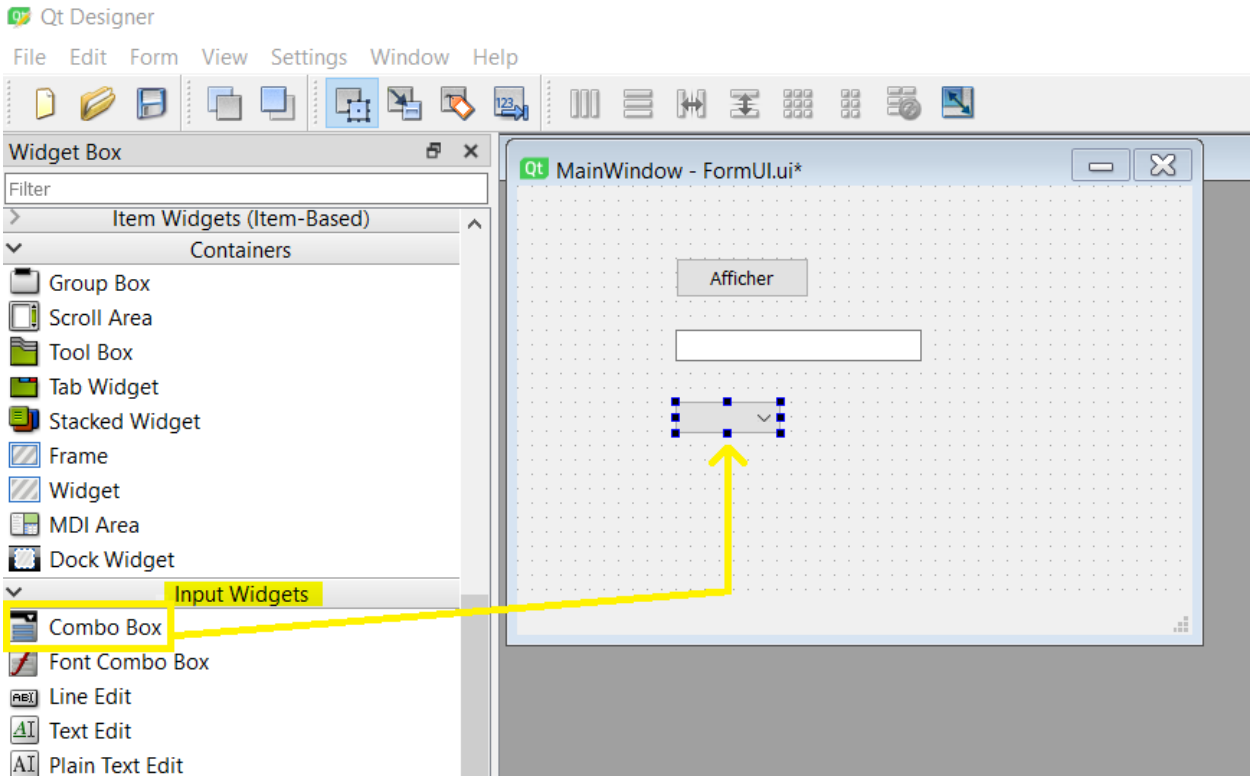
Répétez ce processus jusqu'à ce que vous soyez satisfait de l'ordre de tabulation dans le formulaire - vous n'avez pas besoin de cliquer sur chaque widget d'entrée si vous voyez que les widgets restants sont déjà dans le bon ordre. Les nombres, pour lesquels vous avez déjà défini l'ordre, passent au vert, tandis que ceux sur lesquels vous n'avez pas encore cliqué restent bleus.

Si vous faites une erreur, double-cliquez simplement en dehors de n'importe quel nombre ou choisissez *Redémarrer* dans le menu contextuel du formulaire pour recommencer. Si vous avez de nombreux widgets sur votre formulaire et que vous souhaitez modifier l'ordre de tabulation au milieu ou à la fin de la chaîne d'ordre de tabulation, vous pouvez le modifier à n'importe quelle position. Appuyez sur *Ctrl* et cliquez sur le numéro à partir duquel vous souhaitez commencer. Vous pouvez également choisir Commencer à partir d'ici dans le menu contextuel.

Faites *ESC* pour sortir de ce mode.

LE COMBOBOX

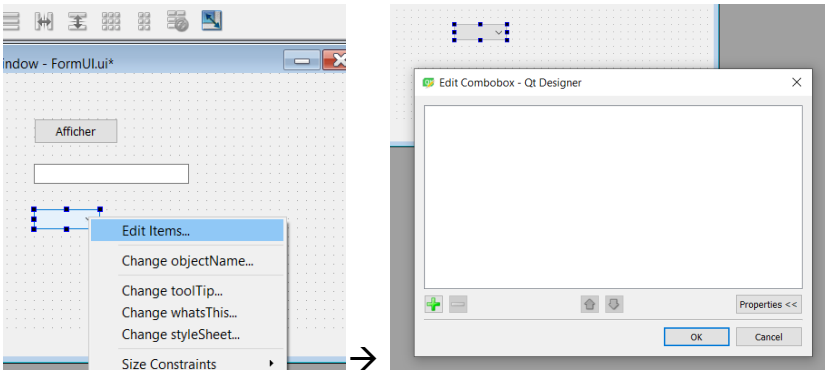
Le `combobox` est une liste déroulante qui facilite les entrées de données et permet de les uniformiser (écrits de la même manière). L'usager n'aura qu'à choisir une des options du `combobox` pour faire son entrée de donnée au lieu de l'écrire lui-même dans un `textbox`.



Pour remplir le combobox de ses options, il y a plusieurs façons. Voyons les deux principales manières d'alimenter une liste déroulante :

À partir de Qt Designer :

Clique-droit sur le comboBox et choisir Edit Items…, et ajouter chaque élément à insérer dans le combobox :



À partir du code Python

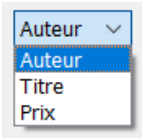
Les éléments insérés dans un `combobox` peuvent être traités à partir de leur valeur respective, ou à partir de leur index. Un peu comme une liste dans Python, chaque élément d'un `combobox` est associé à un index.

Les méthodes de propriété intéressantes à utiliser :

`.addItem()` : Permet d'ajouter des éléments au `combobox`.

Exemple :

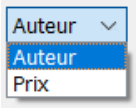
```
self.cboChoix.addItem("Auteur")
self.cboChoix.addItem("Titre")
self.cboChoix.addItem("Prix")
```



`.removeItem()` : Permet de retirer un éléments du `combobox`. L'index de l'élément à retirer est à spécifier.

Exemple :

```
self.cboChoix.removeItem(1)
```



`.currentText()` : Récupère la chaine de caractères qui correspond à l'option choisie dans le `combobox`.

Exemple :

```
ChoixUsager = self.cboChoix.currentText()
```

`.currentIndex()` : Récupère l'index de l'option choisie dans le `combobox`. Le premier élément du `combobox` est à l'index 0.

Exemple :

```
ChoixIndex = self.cboChoix.currentIndex()
```