

## R2.02 : développement d'application avec IHM Feuille TD n° 4

### Création d'une application à interface graphique simple à l'aide de QtDesigner

#### Objectifs :

- 1.- Qt – découverte des classes *QMainWindow*, *QAction*, *QMenu*
- 2.- Découverte du générateur d'interfaces *QtDesigner*

### Activité 1 : Le convertisseur de températures – v2

#### Sujet :

Il s'agit de ré-crée l'application convertisseur de températures vue précédemment en utilisant le générateur d'interfaces QtDesigner.

Les composants graphiques sont placés sur l'interface par glisser-déposer, et leurs propriétés sont réglées via des boîtes de dialogue.

Le(s) fichier(s) contenant l'interface (les interfaces) graphique(s) créé(s) sont des fichiers .XML et porte(nt) l'extension .ui

Lors de la compilation, l'outil **uic**, fourni par l'EDI QtCreator, transforme ce fichier en classe(s) C++. La compilation peut alors se faire.

#### 1. Étape 1 : Préparer votre répertoire de travail

Dans le dossier consacré à la ressource R2.02, créer un dossier td4.

Dans le dossier td4, créer un sous-dossier v2.

#### 2. Étape 2 : Création du projet

Dans votre répertoire td4/v2, **créez** avec l'assistant de QtCreator, créer un projet :

- Nom : convertisseurTemp
- Type du projet : « **Application Qt – Qt Widgets Application** »
- Nom de la classe de la fenêtre principale (et unique) de l'application : **ConvertisseurTemp**
- Cette fois-ci, la classe de base sera : **QMainWindow**
- Cette fois-ci, vous cochez l'option « **Generate Form** ».

#### Class Information

Specify basic information about

Class name:

Base class:

Header file:

Source file:

☒ Generate form

Form file:



## Details

Translation

Kits

Summary

Class name:

Base class:

Header file:

Source file:

☒ Generate form

Form file:

### 3. Étape 3 : Structure de l'application graphique

The screenshot shows the Qt Creator IDE interface. On the left, the 'Projets' (Projects) pane displays the project structure for 'convertisseurTemp'. It includes a 'convertisseurTemp.pro' file, a 'Headers' folder containing 'convertisseurtemp.h', a 'Sources' folder containing 'convertisseurtemp.cpp' and 'main.cpp', and a 'Forms' folder containing 'convertisseurtemp.ui'. The main editor area shows the content of 'convertisseurtemp.h'. The code defines the 'ConvertisseurTemp' class, which inherits from 'QMainWindow'. It includes the 'Q\_OBJECT' macro and defines a public constructor 'ConvertisseurTemp(QWidget \*parent = nullptr);'. A private member 'Ui::ConvertisseurTemp \*ui;' is declared. The code is enclosed in a preprocessor guard 'ifndef CONVERTISSEURTEMP\_H' and 'define CONVERTISSEURTEMP\_H'. The bottom status bar shows 'Sortie de l'application' and a 'Filter' button.

```
1  #ifndef CONVERTISSEURTEMP_H
2  #define CONVERTISSEURTEMP_H
3
4  #include <QMainWindow>
5
6  QT_BEGIN_NAMESPACE
7  namespace Ui { class ConvertisseurTemp; }
8  QT_END_NAMESPACE
9
10 class ConvertisseurTemp : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     ConvertisseurTemp(QWidget *parent = nullptr);
16     ~ConvertisseurTemp();
17
18 private:
19     Ui::ConvertisseurTemp *ui;
20 };
21 #endif // CONVERTISSEURTEMP_H
22
```

#### A remarquer :

- a) La classe **ConvertisseurTemp** contient un membre privé : **\*ui**

Il s'agit du pointeur vers l'objet de type **Ui::ConvertisseurTemp** = l'objet contenant les objets graphiques de la classe **ConvertisseurTemp**.

La relation entre la classe **ConvertisseurTemp** et la classe graphique est implémentée par Qt par une **composition** : l'application est **composée** d'une GUI, représentée par le membre **ui**.

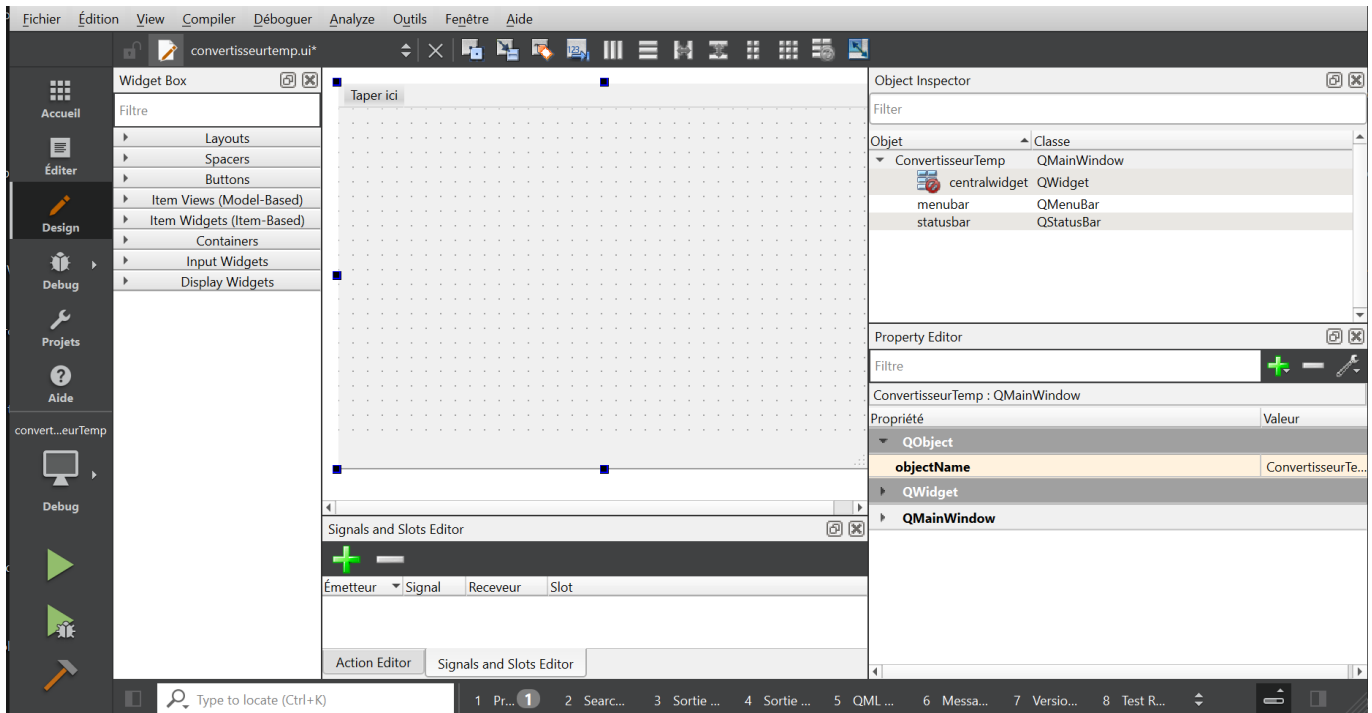
Ainsi, dans `convertisseurtemp.cpp` (le corps de la classe `ConvertisseurTemp`), les éléments graphiques seront accessibles via le pointeur **ui** :

**ui->labelIntitule...**

**ui->bConvertir...**

b) Un double-clic sur le formulaire `convertisseurtemp.ui` ouvre QtDesigner :

#### 4. Étape 4 : Découverte de l'interface de QtDesigner



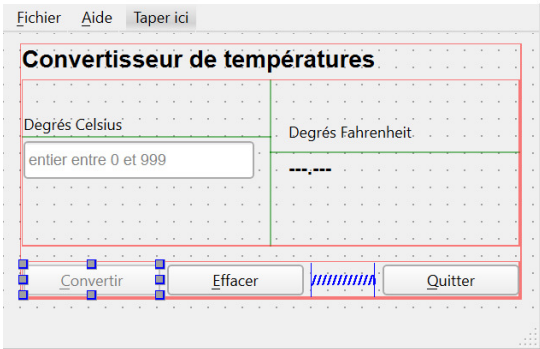
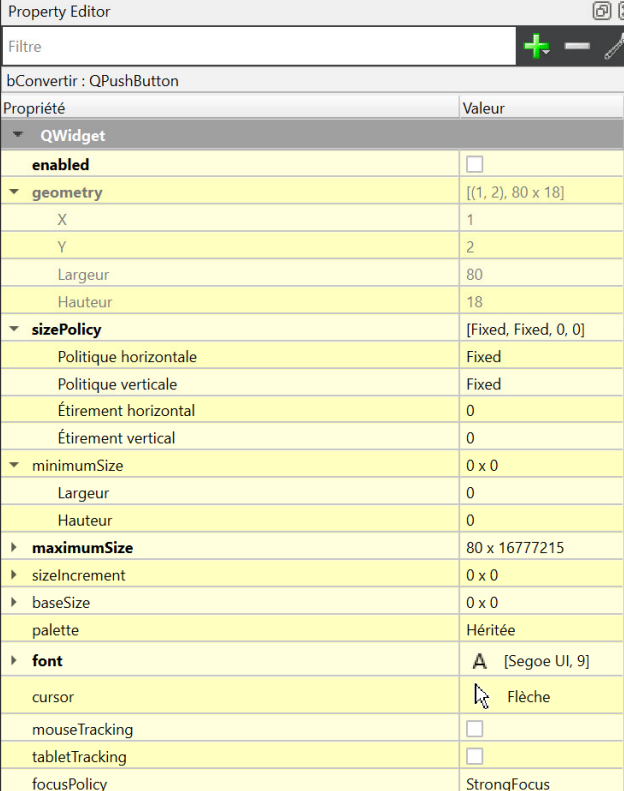
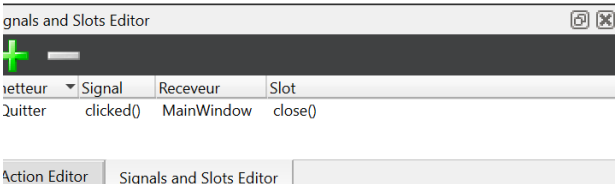
#### A remarquer :

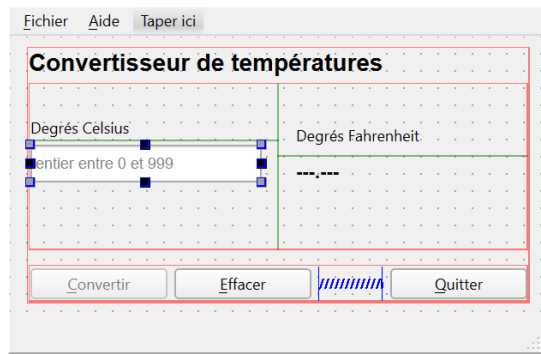
Le panneau central : c'est le dessin de la fenêtre <code>Ui::ConvertisseurTemp</code> , l'interface graphique pointée par le pointeur <b>ui</b> de la classe <code>ConvertisseurTemp</code> .	
L'outil <b>Object Inspector</b> permet de connaître la liste des objets graphiques contenus dans la classe <i>Lister les objets graphiques présents dans la fenêtre affichée.</i> <i>Est-ce normal ?</i>	
Pour chaque objet graphique, l'outil <b>Property Editor</b> permet de visualiser et paramétrer les propriétés de cet objet. <i>Les propriétés sont classées par 'groupes' : à quoi correspondent ces groupes ?</i>	
L'outil <b>WidgetBox</b> contient la liste des objets graphiques prédéfinis qu'il est possible de placer sur la fenêtre, par glisser-déposer. - Les objets graphiques sont classés par catégories. <i>Sans ouvrir les catégories, citer 1 input widget et 1 display widget.</i> Cette fenêtre est déplaçable (dockable). - Un filtre permet de retrouver les widgets par leur nom. Très pratique pour aller vite !	

Les onglets « <b>Action Editor</b> » et « <b>Signals and Slots Editor</b> » servent à définir directement sur l'interface des signals/slots et des objets Action. Nous utiliserons peu l'onglet « Signals and Slots Editor»	
La barre d'outils située au dessus du dessin de la fenêtre permet d'accéder rapidement : <ul style="list-style-type: none"> <li>- A l'édition de widgets : pour la composition de la fenêtre</li> <li>- A l'édition de Signals et Slots</li> <li>- A l'édition de buddies (widgets copains)</li> <li>- A la définition de l'ordre de navigation entre les objets graphiques</li> <li>- A la mise en place de layouts</li> </ul>	
Compiler et exécuter. Tester les fonctionnalités par défaut de la fenêtre principale.	

## 5. Étape 5 : Reconstituer l'interface du convertisseur de températures

- A l'aide du filtre du **WidgetBox**, retrouver rapidement les objets graphiques composant l'interface du convertisseur, et les déposer tous sur la fenêtre, sans les arranger
- Une fois tous les éléments graphiques présents, sélectionnez-les, un par un, et, à l'aide de l'éditeur de propriétés, les paramétrer de la même manière que dans la version manuelle : nom de la variable, contenu du libellé, police, style ....
- Peaufiner la taille (largeur) des boutons Convertir /Effacer, et de la zone de texte, leur sizePolicy, minimumSize et maximumSize.

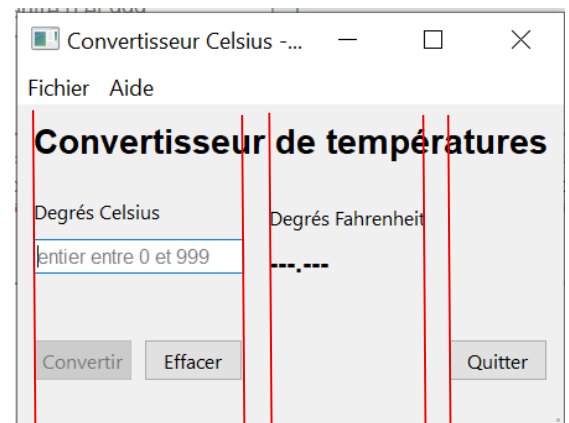


Propriété	Valeur
<b>QWidget</b>	
enabled	<input checked="" type="checkbox"/>
<b>geometry</b>	[(1, 36), 135 x 22]
X	1
Y	36
Largeur	135
Hauteur	22
<b>sizePolicy</b>	[Preferred, Fixed, 0, 0]
Politique horizontale	Preferred
Politique verticale	Fixed
Étirage horizontal	0
Étirage vertical	0
<b>minimumSize</b>	0 x 0
Largeur	0
Hauteur	0
<b>maximumSize</b>	170 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Héritée
<b>font</b>	A [Arial, 9]
cursor	I IBeam
mouseTracking	<input checked="" type="checkbox"/>
tabletTracking	<input type="checkbox"/>
focusPolicy	StrongFocus

Émetteur	Signal	Receveur	Slot
bQuitter	clicked()	MainWindow	close()

- d) Contrôler l'évolution du fichier xml.
- e) Il existe dans le projet un fichier ui\_convertisseurtemp.h. Il est généré par l'outil uic à partir du fichier xml. Trouvez-le et regardez analysez son contenu.
- f) Placer les objets graphiques dans les layouts

La définition de tailles maxi pour les boutons et zone de texte, ainsi que le réglage des espaces entre les éléments de la grille, les espaces au dessus et au dessous de la grille, permettent d'aligner / répartir les objets graphiques de manière un peu harmonieuse..



g)

### Attention :

le widget central de la fenêtre QMainWindow doit être attaché à un layout. Associer le layout principal (le layout horizontal) à ce widget central.

Objet	Classe
ConvertisseurTemp	QMainWindow
centralwidget	QWidget
horizontalSpacer_2	Spacer
horizontalSpacer_3	Spacer
verticalLayout	QVBoxLayout
gridLayout	QGridLayout
eValeurSource	QLineEdit
hSlider	QHBoxLayout

- h) Propriété **placeholder** de la classe QLineEdit : elle permet d’afficher un message d’aide à la saisie. Le message s’efface dès que l’utilisateur démarre la saisie, il réapparaît dès que la zone de texte est réinitialisée/
- i) Propriété **autodfault** de la classe QPushButton – cochée à true : lorsque le focus est sur un bouton et que l’utilisateur presse la touche Entrée, la méthode associées au clic sur le bouton est exécutée<sup>1</sup>.
- j) Bloquer la taille de la fenêtre dès la première Conversion :  
méthodes setFixedSize() et minimumSize() de la classe QMainWindow

---

<sup>1</sup> Quel principe d’ergonomie favorise-t-on en activant cette propriété ?

## 6. Étape 6 : Améliorer la saisie

### **Objectif :**

La version actuelle de l'application ne rend actif le bouton Convertir que si la zone de texte n'est pas vide.

La méthode convertir() vérifie si le texte saisi est convertible en un entier.

Si c'est le cas la méthode réalise la conversion, si ce n'est pas le cas, elle affiche un message d'erreur.

### **Il s'agit maintenant de contraindre la saisie des caractères souhaités.**

Du coup, l'algorithme de la méthode convertir() sera simplifié puisque la donnée fournie sera toujours correcte !

#### **a) Définir la combinaison des caractères autorisés**

Pour ce faire, on définit un objet (une expression régulière) : il exprime la/les combinaisons de caractères autorisés :

. au maximum, 1 caractère - **ou** 1 caractère +

. suivi d'une combinaison d'au plus 3 chiffres compris entre 0 et 9.

La syntaxe :

```
QRegExp carAutorises ("[-,+] {0,1} [0-9] {0,3}");
```

#### **b) Définir un objet validateur chargé de valider la combinaison précédente.**

La syntaxe :

```
QRegExpValidator *validateur = new QRegExpValidator(carAutorises,  
                                                    fenetreParent);
```

fenetreParent est le widget contenant le validateur : dans notre cas, c'est la fenêtre principale de l'application.

#### **c) Associer le validateur à la zone de texte.**

La syntaxe :

```
ui->eValeurSource->setValidator(validateur);
```

### **Liens**

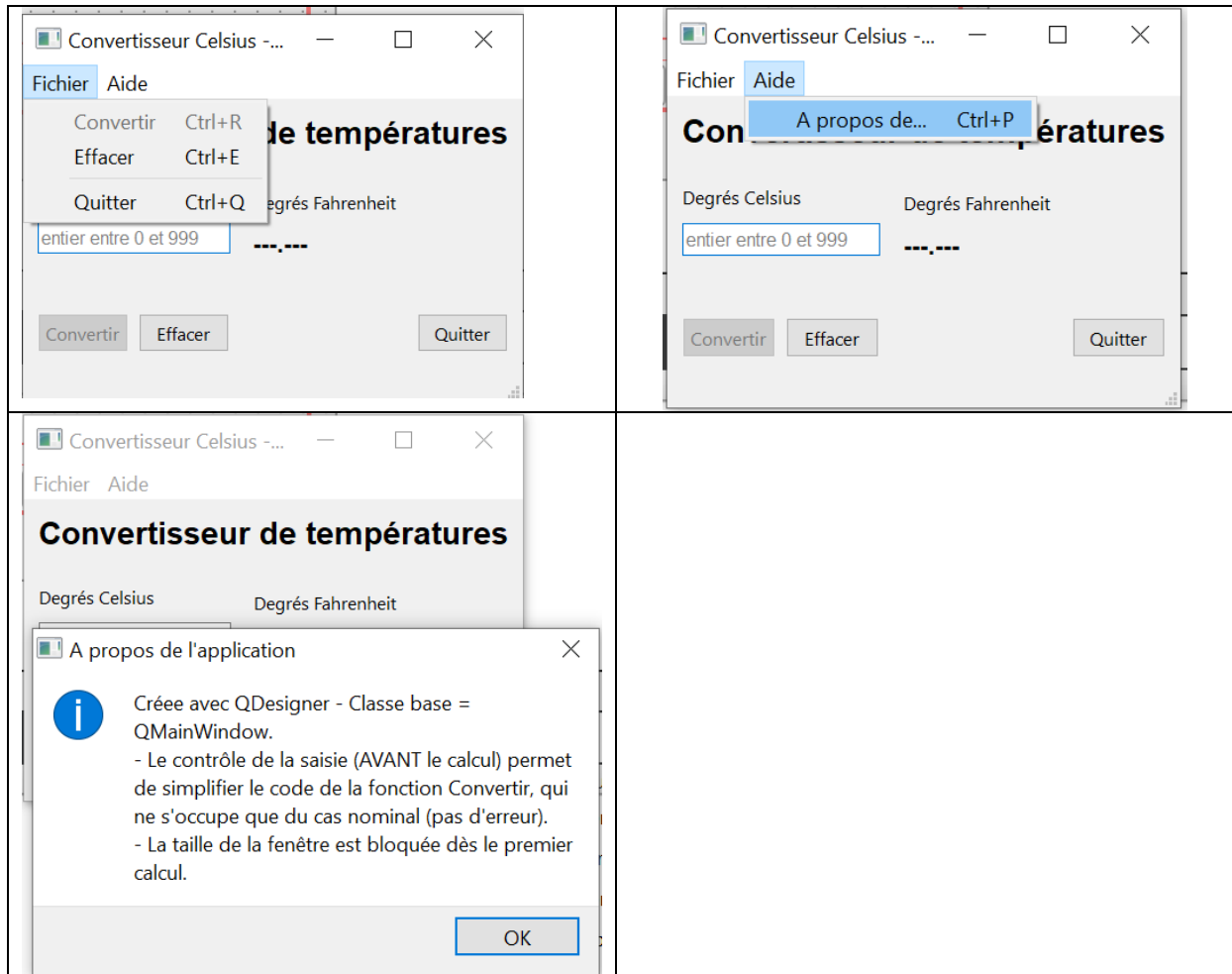
voir projet find.zip disponibles sur eLearn, dans la rubrique Ressources.

#### **d) Simplifier le code de la méthode convertir().**

## 7. Étape 7 : Ajouter les items de menu

### Objectif :

Dans la barre des menus, nous souhaitons ajouter les menus Fichier et Aide, avec les items de menu suivants :

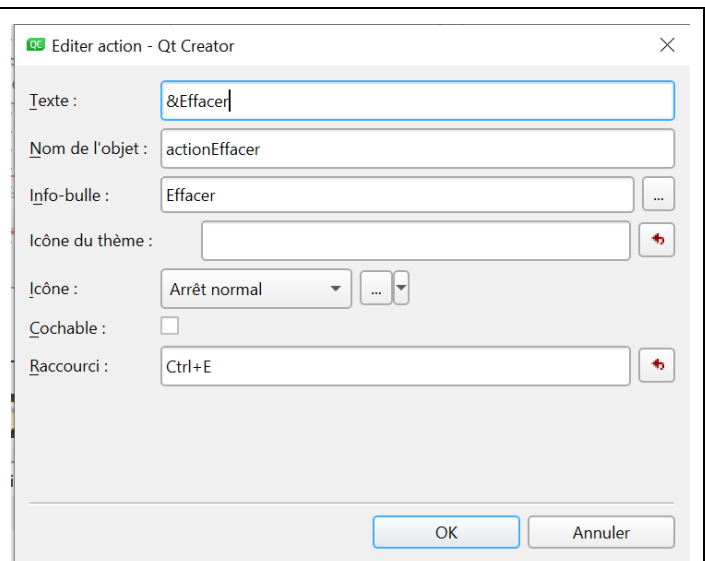


### a) Définir les objet QAction

A l'aide de l'Action Editor, définir les 4 objet QAction nécessaires, en remplissant le formulaire proposé, uniquement les champs :

- Texte : le libellé de l'item de menu.  
L'esperluette permet d'accéder à l'item via la touche Alt. Par exemple ici : AltF-E pour Effacer
- Le raccourci = Ctrl + E

La figure ci-dessous montre 'mes' objets QAction ;  
Dans votre cas, les cases 'Utilisé' ne sont pas encore cochées.

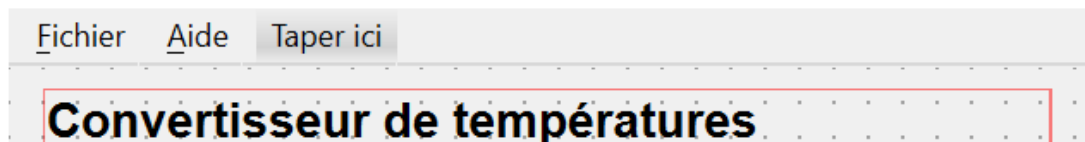




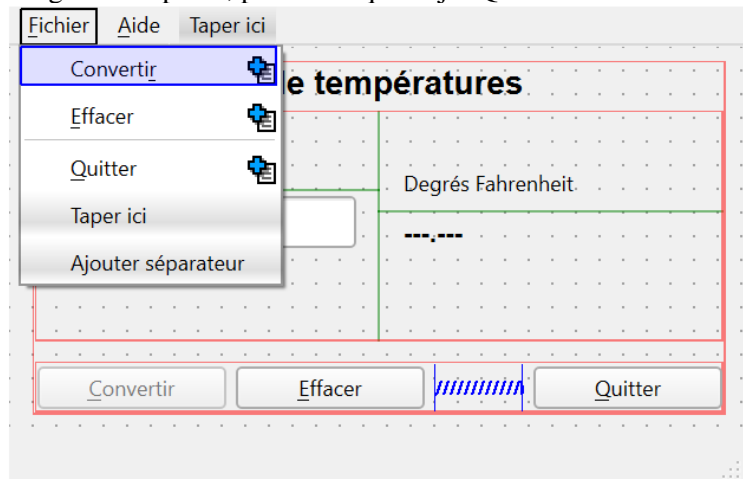
Action Editor						
Filtre						
Nom	Utilisé	Texte	Raccourci	Vérifiable	Info-bulle	
<input type="checkbox"/> actionConvertir	<input checked="" type="checkbox"/>	Converti&r	Ctrl+R	<input type="checkbox"/>	Convertir	
<input type="checkbox"/> actionEffacer	<input checked="" type="checkbox"/>	&Effacer	Ctrl+E	<input type="checkbox"/>	Effacer	
<input type="checkbox"/> actionQuitter	<input checked="" type="checkbox"/>	&Quitter	Ctrl+Q	<input type="checkbox"/>	Quitter	
<input type="checkbox"/> actionA_propos_de	<input checked="" type="checkbox"/>	A &propos de...	Ctrl+P	<input type="checkbox"/>	A propos de	

## b) Associer les objets QAction aux éléments de menu

- Dans la barre de menu, saisir les 2 éléments de menu : &Fichier et &Aide.



- Par glisser-déposer, placer chaque objet QAction dans le menu qui lui correspond.



## c) Dans le constructeur de la classe, connecter chaque objet QAction avec le slot adéquat.

Exemple

```
connect(ui->actionConvertir, SIGNAL(triggered()), this, SLOT(convertir()));
```

### Liens

<https://doc.qt.io/qt-5/qaction.html>

## d) Test du convertisseur.

Vérifier que la conversion est réalisable selon tous les moyens suivants :

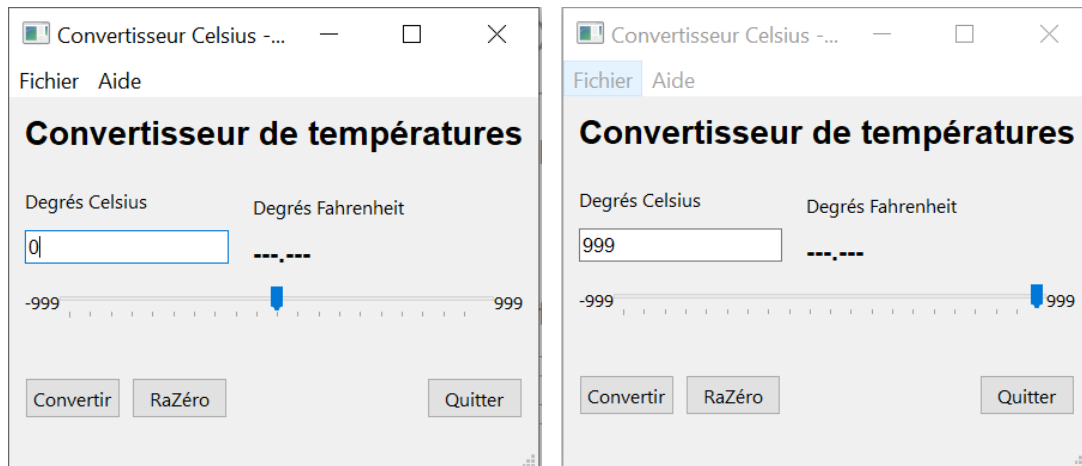
- Clic sur bouton Convertir
- Alt-C
- « Entrée » lorsque le focus est sur le bouton Convertir, par exemple suite à une navigation via les touches de Tabulation
- Ctrl+R
- Alt-F-R

## Activité 2 : Le convertisseur de températures – v3

### Sujet :

Il s'agit de modifier l'application convertisseur de températures comme suit :

- Ajout d'un curseur horizontal (horizontal slider) qui permet de choisir des températures Celsius dans la plage autorisée, sans avoir à saisir de valeur
- La valeur pointée par le curseur du slider et le contenu de la zone de texte sont synchronisés : le déplacement du curseur entraîne un changement de valeur dans la zone de texte et, inversement, une saisie dans la zone de texte entraîne un déplacement du curseur.
- L'état 'initial' du convertisseur est modifié : Le curseur du slider est positionné au milieu (0), la zone de texte affiche donc la valeur 0, et le bouton Convertir est donc toujours actif.



- Les degrés Fahrenheit ne sont affichés que lorsque l'utilisateur demande la conversion
- Le comportement et l'intitulé du bouton Effacer sont modifiés.

**Question :** Lister les raisons de cette modification et lister les comportements / propriétés à modifier.

### 1. Étape 1 : Préparer votre répertoire de travail

Dans le dossier td4, créer un répertoire v3.

### 2. Étape 2 : Récupérer le projet de la v2

Dans votre répertoire td4/v3, coller le contenu du répertoire td4/v2/convertisseurTemp.

Supprimer le fichier .pro.user

Ouvrir le projet.

### 3. Étape 3 : Mettre en œuvre l'objectif annoncé

#### **Liens :**

<https://doc.qt.io/qt-5/qslider.html>