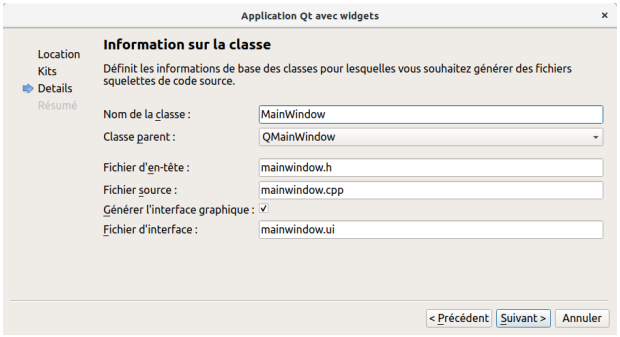
4 TP 4 : UI et QTDesigner

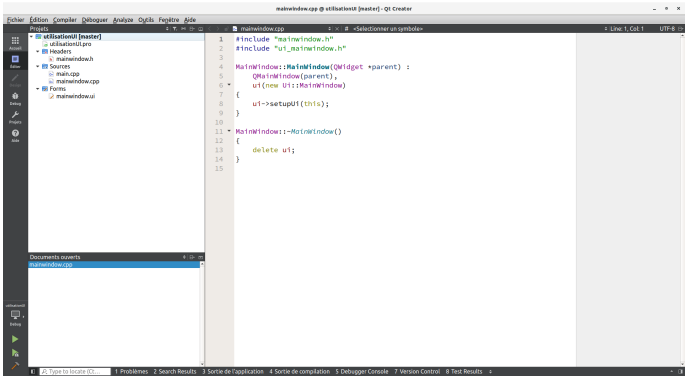
Vous allez découvrir dans ce TP : la création d'une UI (« *User Interface* ») que `QTDesigner` permet de manipuler (construction graphique, signaux et `slots`).

4.1 La version 0

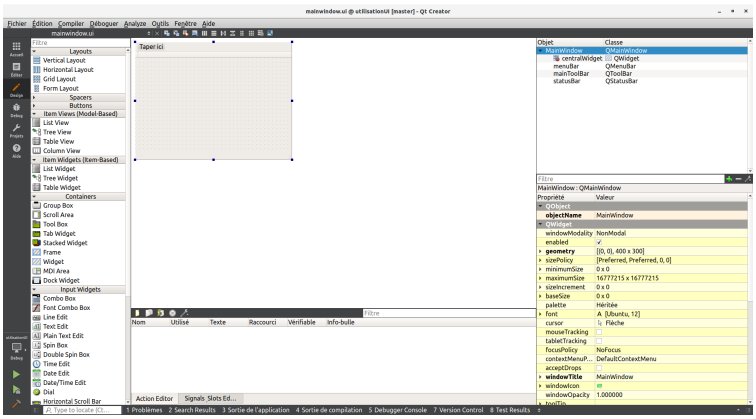
Vous devez créer une nouvelle application QT, avec la création d'une UI. Pour cela, lorsque vous utiliser « Nouveau Projet », il faut laisser cochée la case « Générer l'interface graphique ».



Vous avez maintenant un projet et son exécution correspondante :



Pour accéder à `QtDesigner`, cliquer sur « Design » dans les icônes à gauche du projet ou double-cliquer sur « mainwindow.ui » dans l'explorateur de projet.

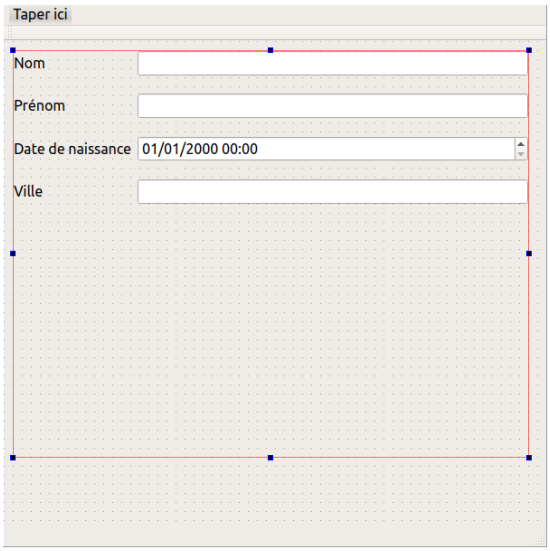


On retrouve donc une `QMainWindow` avec `MenuBar`, `mainToolbar`, `Central Widget`... Le « Designer » vous propose pour chaque objet sélectionné les propriétés correspondantes (partie en jaune en bas à droite) et les `Widgets` possibles (liste à gauche).

On va commencer par changer une propriété (à vous de la trouver) :
— modifier le `windowTitle` de la `QMainWindow`, pour afficher « Ma première UI »
Ok, fastoche... Maintenant on va remplir le `Central Widget` :

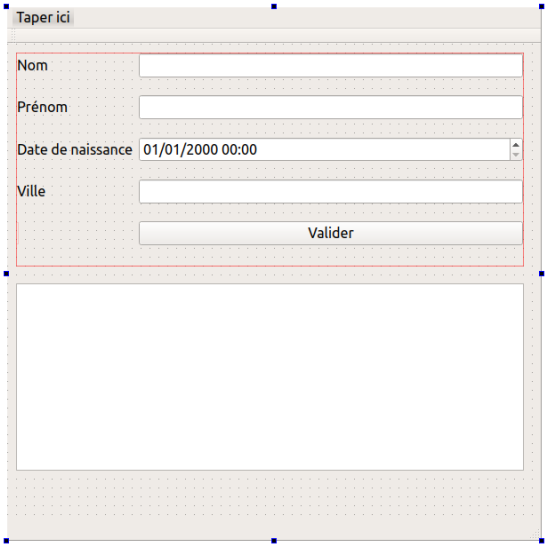
1. ajouter une `Form Layout`,
2. dans ce formulaire, ajouter une ligne (cf clic droit) pour chaque champ suivant :
   — nom (`QLineEdit`)
   — prénom (`QLineEdit`)
   — date de naissance (`QDateTimeEdit`)
   — ville (`QLineEdit`)

Votre interface devrait maintenant ressembler à cela :



Il nous faut maintenant mettre en place des interactions (signal/`slot`) et une structure de données pour notre formulaire :

1. rajouter un `PushButton` « Valider »,
2. rajouter, en dehors du `Form Layout`, un `textEdit` que l'on nommera `texteResultat` (cf l'« `objectName` »),



3. pour vérifier que le contenu d'un champ est accessible, que le bouton « Valider » fonctionne et que le `textEdit` est rempli, on va relier ces éléments par des signaux/`slots`.

(a) sélectionner le bouton « Valider », bouton droit « aller au `slot` ». Cela crée automatiquement une fonction du `MainWidget` nommée `void MainWindow::on_pushButton_clicked()`, avec la déclaration dans le « .h » (et le `connect`, qui est inclus dans « mainWindow.ui » donc non-modifiable en texte). On va vérifier que cela fonctionne, ajouter un :

```
qDebug() << bouton cliqué;
```

dans le `slot` et lancer le programme. Au passage on voit qu'il faut définir des noms adaptés pour vos widgest (en changeant la propriété « object name ») puisque ce nom est utilisé automatiquement par `QtDesigner`. Vous risquez sinon de ne plus comprendre votre programme si des noms génériques sont utilisés.

(b) remplissez une structure de données « `Personne` » (classe avec accesseurs [1]), qui stockera les contenus des champs nom, prénom, date de naissance, ville sous forme de `std::string`. L'appui sur « Valider » provoque l'enregistrement. N'oubliez pas de déclarer un variable de type « Personne » dans le « mainWindow.h », d'instancier cette variable dans le constructeur de `MainWindow` et de désallouer cette mémoire dans le destructeur (`delete`).

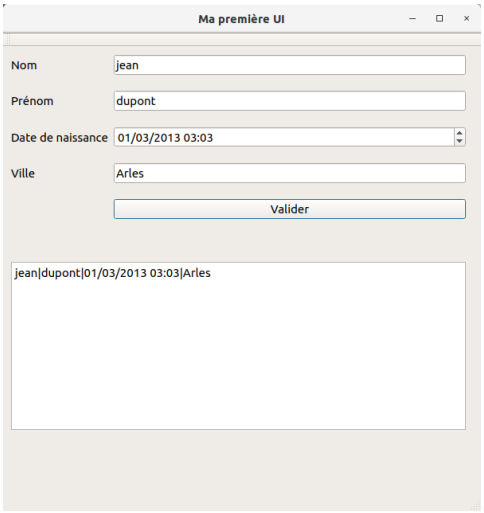
(c) pour accéder aux champs de l'UI :

```
ui->nomLineEdit->text();
```

(si votre `QLineEdit` est « nomLineEdit »). Attention, c'est un `QString` et votre classe `Personne` attend un `std::string`.

(d) créez une fonction `Personne::affiche()` qui permet d'afficher dans le flux `cout` (toujours pas d'interaction avec QT).

(e) affichez les données de la personne dans le `QTextEdit`.



4.2 Menus/Actions

Nous allons maintenant nous concentrer sur les menus de l'application.

4.2.1 Ajout d'un menu « About »

Premier exemple : ajout d'une action dans l'UI et raccordement avec le menu de la `QMainWindow`.

1. Astuce : dans le fichier « personne.h », un clic gauche sur une variable de la classe, bouton droit « Créer les fonctions membres, accesseurs ou mutateurs » - parfois « Create getter » ou « Create setter » - fera le travail pour vous

Nous allons créer une action, relier un morceau du menu avec cette action et associer un signal/`slot` à cette action. Le but est de ne pas appeler directement une fonction depuis une entrée de menu, ni de lever un signal avec une fonction pré-définie. Avec une action, on détache l'interface de l'action engendrée, le code est plus général [2].

Dans l'UI, la zone basse contient le « `Signal_slots Editor` » et l'« `Action Editor` » (si ce dernier n'est pas visible, bouton droit pour une menu contextuel). Ajoutez une entrée `actionA_Propos` dans l'`Action Editor`. Son texte est « A propos ».

Dans « MainWindow.cpp », dans le constructeur, déclarez et connecter l'entrée de menu :

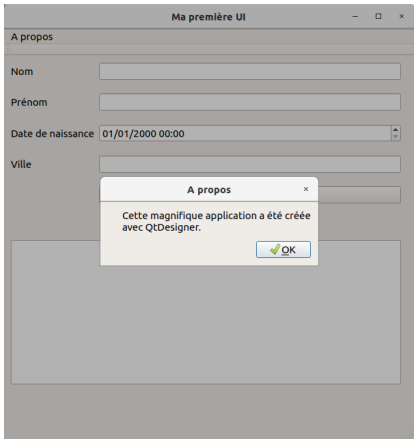
```
ui->menuBar->addAction(ui->actionA_Propos);
connect(ui->actionA_Propos, SIGNAL(triggered(bool)), this, SLOT(slot_aPropos()));
```

Évidemment, il vous faut déclarer/créer le slot `slot_aPropos()`...
Il contiendra une `QMessageBox` (attention à l'*include*) :

```
QMessageBox * messageAbout = new  QMessageBox();
messageAbout->about(this, "A propos",
  "Cette magnifique application a été créée\navec QtDesigner.");

delete messageAbout;
```

Cela devrait donner cela :



4.2.2 Menu avec l'UI

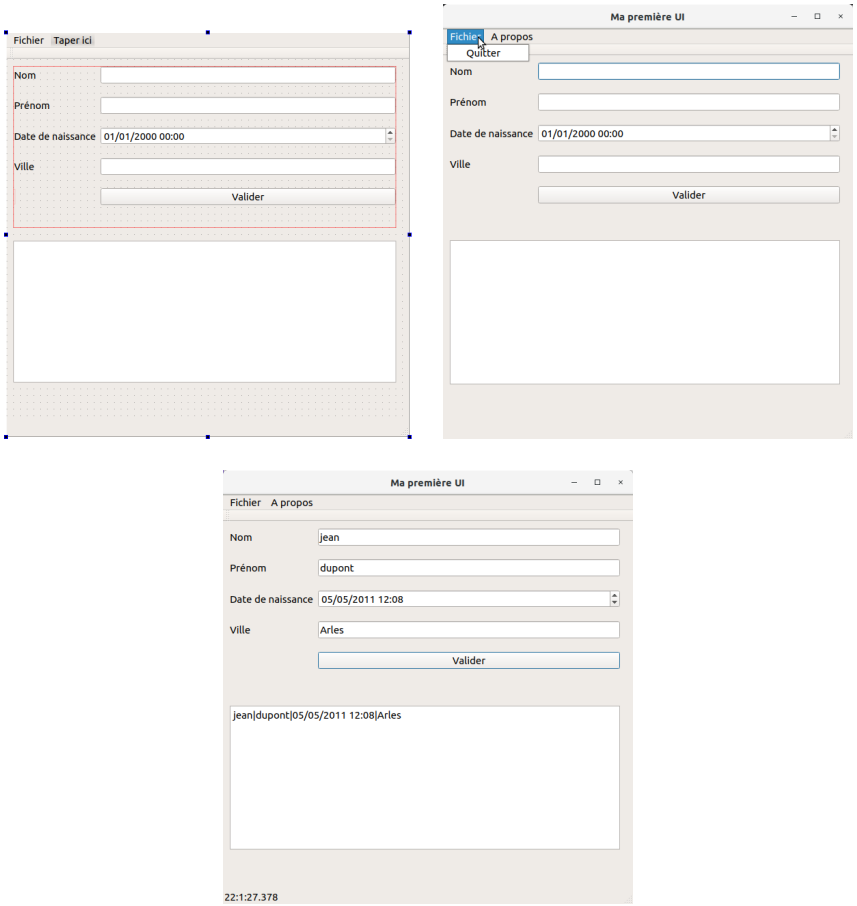
Maintenant qu'on sait le faire dynamiquement, on va ajouter un menu avec l'UI dans `QTDesigner`. En fait c'est trop facile : dans `QTDesigner`, la barre de menu est directement modifiable « Taper ici ».

Ajouter un menu « Fichier » et un sous-menu « Quitter ». Vous voyez également que l'action `actionQuitter` a été créée automatiquement.

On associe un `slot` avec l'action `actionQuitter` :

```
connect(ui->actionQuitter, SIGNAL(triggered(bool)), qApp, SLOT(quit()));
```

Et vous devriez pouvoir quitter l'application via ce nouveau menu !



4.3 Label heure

Last trick, créer un label et afficher l'heure courante.
Affichez-y l'heure courante...

Il faut :

1. créer/définir/allouer un `QLabel` dans la `MainWindow`,
2. créer/définir/allouer un `QTimer`, lui donner un intervalle de 10 ms, le démarrer (`start`),
3. lier le `timeout()` du `QTimer` avec le remplissage du `QLabel`,
4. donner la valeur de l'heure courante au `QLabel` (allez voir la doc.) :

```
QTime timeValue;
timeValue=QTime::currentTime ();
QString format="H:m:s.z";

label->setText (timeValue.toString (format));
```

En suivant la documentation de `QStatusBar`, depuis le constructeur de `MainWindow`, ajouter une instance de barre de statut à l'`ui` qui fasse la même chose.

2. On peut avoir un développeur pour l'interface graphique et un autre pour les fonctions