QT/C++

- CIR2 -

TP2

Ayoub KARINE (ayoub.karine@isen-ouest.yncrea.fr)

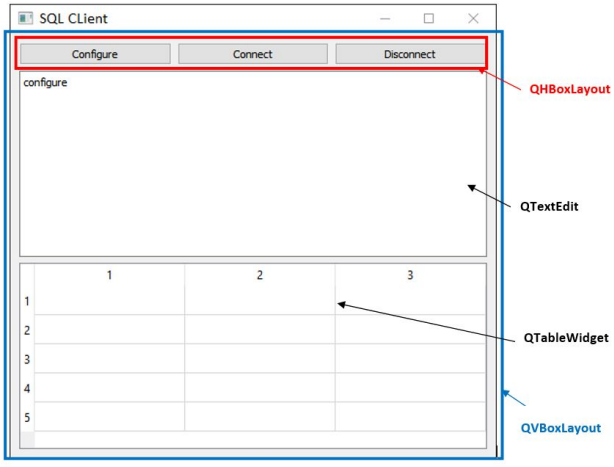
Objectifs

Le but de ce TP est de concevoir un client graphique qui va interagir avec une base de données. L'IHM sera développée en utilisant le framework Qt.

L'API Qt est documentée ici : http://doc.qt.io/qt-5/classes.html

Partie 1

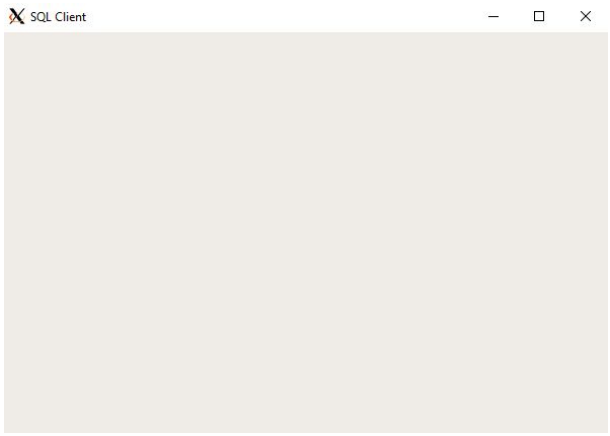
L'IHM à développer dans cette partie est :



Etape 1 : création d'une fenêtre vide + ajout du widget parent

1. Créer une classe SQLClientWindow héritant de QMainWindow. En QT, la classe QMainWindow hérite de QWidget
2. Créer un constructeur à cette classe qui :
    a. modifie le titre de la fenêtre (voir affichage ci-dessus). Utiliser la méthode setWindowTitle
    b. change la taille initiale de la fenêtre en weight=600 et height=400. Pour ce faire, utiliser la méthode setMinimumSize de la classe QWidget
    c. crée un widget parent en utilisant la classe QWidget
    d. crée un layout principal vertical en utilisant la classe QVBoxLayout.
    e. associe le layout principal au widget parent. Pour ce faire, utiliser la méthode setLayout de la classe QWidget
    f. définit le widget parent comme widget central de la fenêtre principale (utiliser la méthode setCentralWidget)

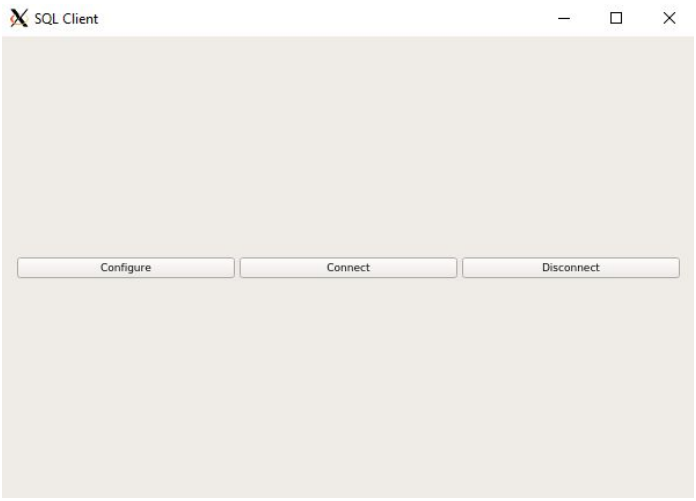
3. Dans la fonction main, tester la classe SQLClientWindow.
   L'affichage doit être comme suit :



Etape 2 : ajout du premier widget : les bouttons

1. Créer une classe ButtonsPanel héritant de QWidget. Cette classe va modéliser un groupe de boutons
2. Dans la classe ButtonsPanel :
   a. Créer, dynamiquement, trois objets (widgets) de la classe QPushButton représentant chacune les trois boutons de l'IHM (voir figure ci-dessus)
   b. Créer un constructeur qui :
      i. ajoute les textes aux boutons (voir figure ci-dessus)
      ii. crée un layout horizontal
      iii. ajoute les trois boutons (widgets) au layout horizontal. Pour ce faire, utiliser la méthode addWidget de la classe QWidget
      iv. associe le layout horizontal au widget ButtonsPanel

3. Dans la classe SQLClientWindow, créer un objet buttonsPanel de la classe ButtonsPanel
4. Dans le constructeur de la classe SQLClientWindow :
    a. initialiser buttonsPanel en utilisant le constructeur de ButtonsPanel
    b. ajouter le widget ButtonsPanel au layout principal
5. Dans la fonction main, tester la classe SQLClientWindow. L'affichage doit être comme suit :



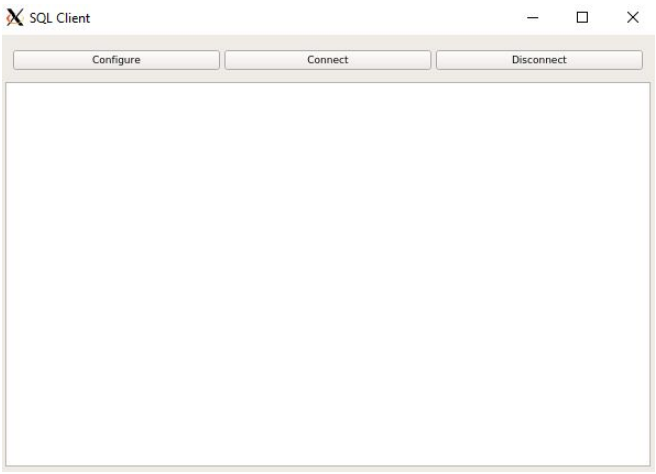
Etape 3 : ajout du deuxième widget : la zone du texte

1. Dans la classe SQLClientWindow, créer un objet notificationPanel à partir de la classe QTextEdit
2. Dans le constructeur de la classe SQLClientWindow :
    a. initialiser notificationPanel en utilisant le constructeur de la classe QTextEdit

b. ajouter le widget notificationPanel au layout principal

3. Dans la fonction main, tester la classe SQLClientWindow.

L'affichage doit être comme suit :



Etape 4 : ajout du troisième widget : le tableau

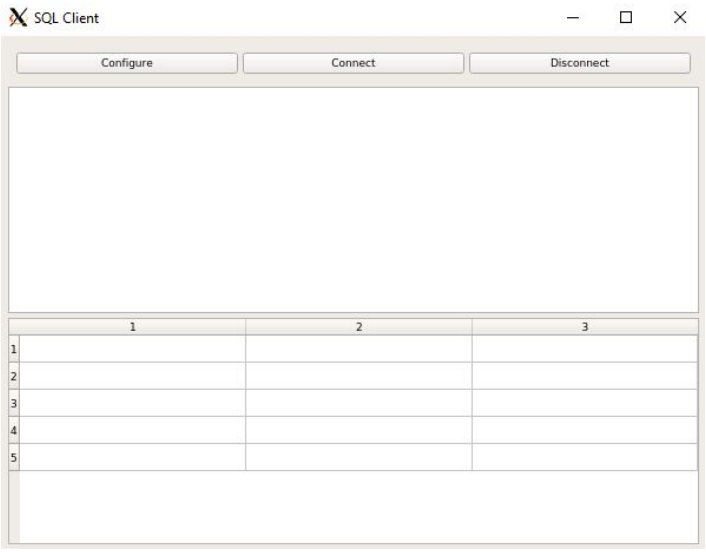
1. Dans la classe SQLClientWindow, créer un objet resultTable de la classe QTableWidget
2. Dans le constructeur de la classe SQLClientWindow :

    c. initialiser resultTable en utilisant le constructeur de la classe QTableWidget

    d. ajouter le widget resultTable au layout principal pour générer le même résultat de l'IHM ci-dessus. Pour avoir une table qui s'affiche sur toute la largeur de la fenêtre, il est nécessaire de le spécifier en utilisant la ligne suivante :

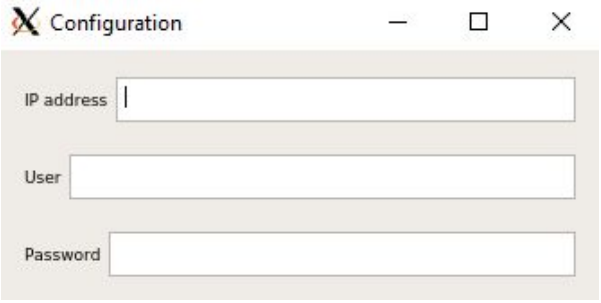
```
votreinstance->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
```

3. Dans la fonction main, tester la classe SQLClientWindow. L'affichage doit être comme suit :



Partie 2

L'IHM à développer dans cette partie est :



Etape 1

Créer une classe LabeledTextField héritant de QWidget et ayant (cette classe va modéliser un label suivi d'une zone du texte) :

1. deux attributs : un objet de la classe QLabel et un objet de la classe QTextEdit
2. un constructeur qui permet d'initialiser les deux attributs (widgets) et les ajouter dans un layout horizontal de la classe

QHBoxLayout. Utiliser la méthode setMaximumHeight de la classe QTextEdit pour agrandir la zone de texte

Etape 2

1. Créer une classe ConfigurationDialog, sans attributs, héritant de QDialog qui permet d'avoir le résultat final de l'IHM
2. Tester dans le main la classe ConfigurationDialog
3. Modifier le code de la partie 1 pour que les IHM des parties 1 et 2 s'ouvrent au même temps