

Auteur: karim NASR

Date: 15/11/2025

Objet du document:

A pour but d'expliquer quelques programmes pour approfondir QT.

Historique:

18/09/24 KNA ajout de QtThreeWidgetTutorial.Exemple sur les checkbox.

18/09/24 KNA, évolution de la première version, ajout de LCD et de la somme de la commande.

20/09/24 KNA, ajout de nouveaux projet DisplayWindows et ReadingWritingTextFile.

12/10/2024 KNA ajout exemple sur QFontDialog-utilisateur change la police de caractere dans fenetre, InputDialogMultiline, ToolBar.

Debut de manipulations avec les exemples du cours dans documentation « Qt-Widgets-Layouts »: QVBoxLayout.Ajout de QWidgetGroupBox/QWidgetSliderGroup/  
QtCheckedListWidget/QtLCDnumber.

11/11/2024 KNA ajout exemple sur la gestion d'une base de donnees contacts par SQLite (nom du projet GestionContacts)

13/11/2024 KNA Modification de la GestionContact par ajout de la gestion categorie avec une comboBox.

28/10/2025 KNA ajout TP horloge digitale compte a rebours

15/11/2025 KNA ajout Qt-QsqlRelationalModel (projet Pychart),QDataWidgetMapper extrait livre Qt framework,Qt-CreationFormulaire, utilisation QFont avec formulaire,Qt-GestionFeuilleStyle,Qt-GestionSignalSlot.

QtThreeWidgetTutorial:

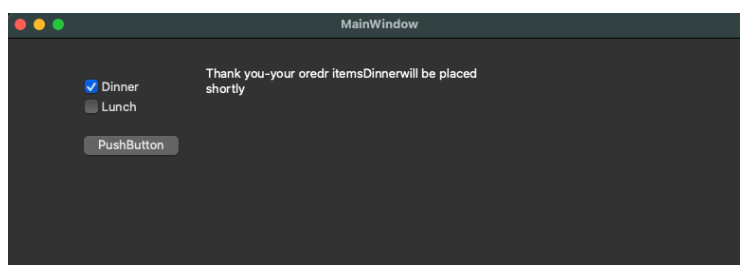
Contient un exemple basique sur les checkbox dans QtCreator.

Ici le choix est fait entre Dinner et lunch et qhand appui sur « Push button » on affiche dans le label les differents choix faits.

Extrait youtube

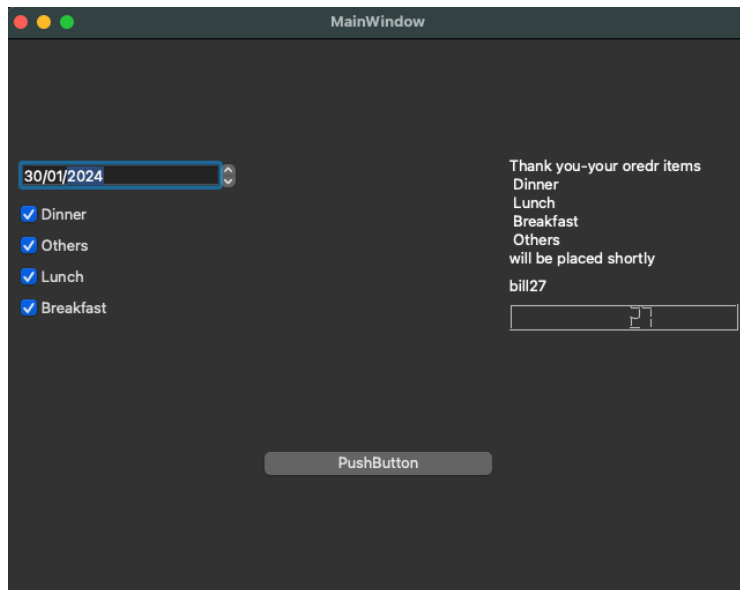
**QCheckBox | How to use QCheckBox in Qt5 | (Qt C++ Tutorial #16)**

**MacDigia**

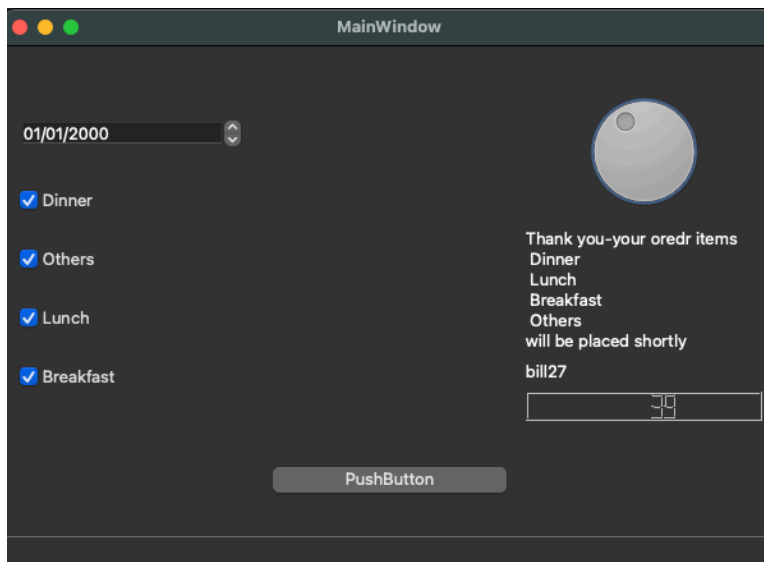


Ajout dans la deuxième version d'un label d'affichage de la somme ainsi d'un LCD d'affichage de la valeur entière de la somme.

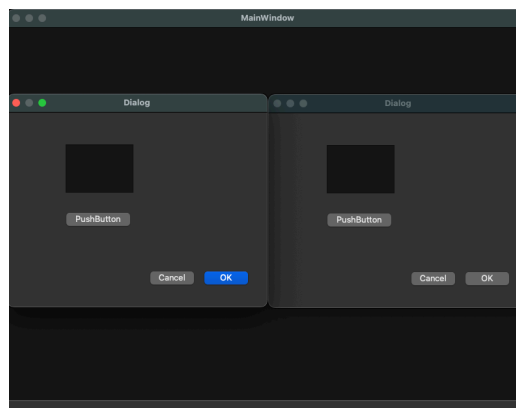
Ajout  
On  
LCD.



d'un curseur circulaire dont la valeur est affichée sur le LCD lorsque celui change. affiche également toujours la somme de la commande sur le



Le dossier displayingWindows est extrait de youtube VoidRealms  
 Creation d'un projet nommé DisplayingWindows sous Qt  
 pour la création de boites de dialogues extrait de youtube  
 VoidRealms  
 // creation d'un nouveau formulaire de dialog dans formulaire et le fichier  
 //dialog.h  
 // modification de mainwindow.h pour intégrer Dialog \*mDialog;



Le dossier ReadingWritingTextFile est extrait de YouTube

On écrit et lit dans un fichier « test.txt » avec OStream, si on change le nom du fichier alors message « file not found »

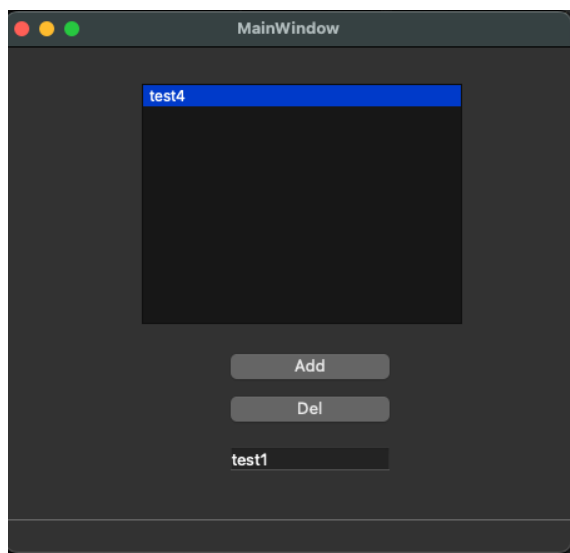
## Qt 6 - Episode 19 - Read and writing text files

VoidRealms

Le dossier QListWidgetExemple est extrait de YouTube de LearnQT (How to add items and delete selected items with QListWidget)

Montre les fonctions de list widget, on peut ajouter ou supprimer

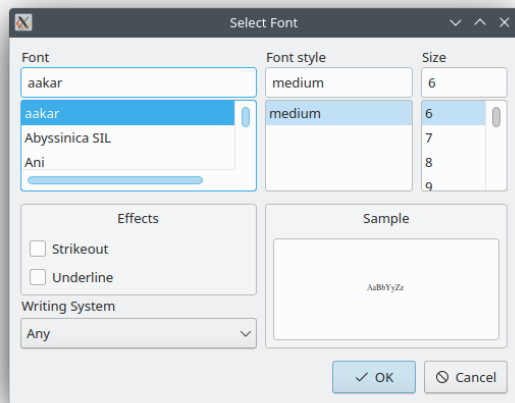
Des items saisies dans lineEdit avec boutons Add/Del



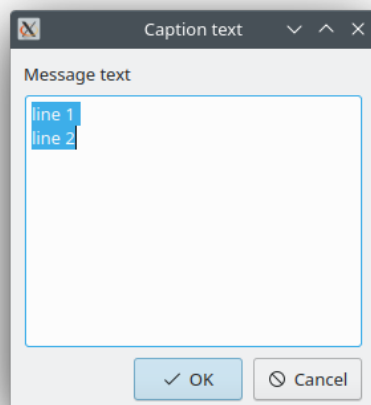
Le dossier QtFontdialog est extrait du site [https://github.com/gammasoft71/Examples\\_Qt/tree/master](https://github.com/gammasoft71/Examples_Qt/tree/master)

Montre une fenêtre où l'utilisateur change au choix la police

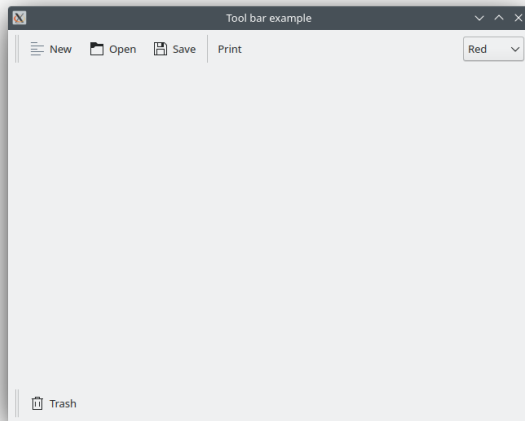
Output



Le dossier Qt-InputDialogMultiline est extrait du même site

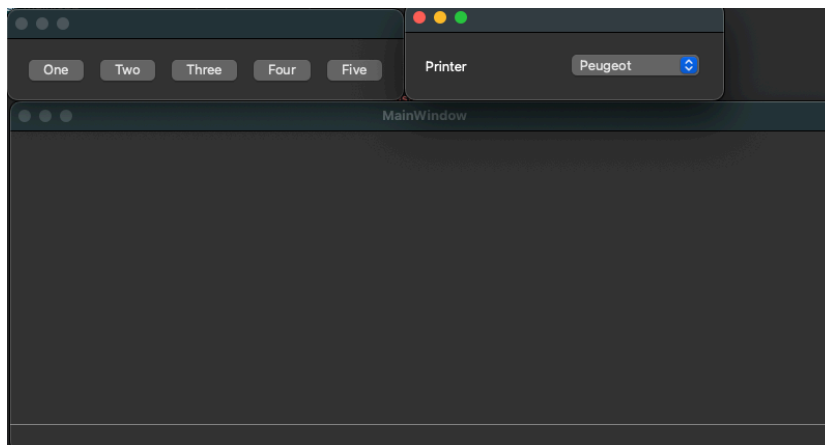


Le dossier Qt-ToolBar est extrait du même site



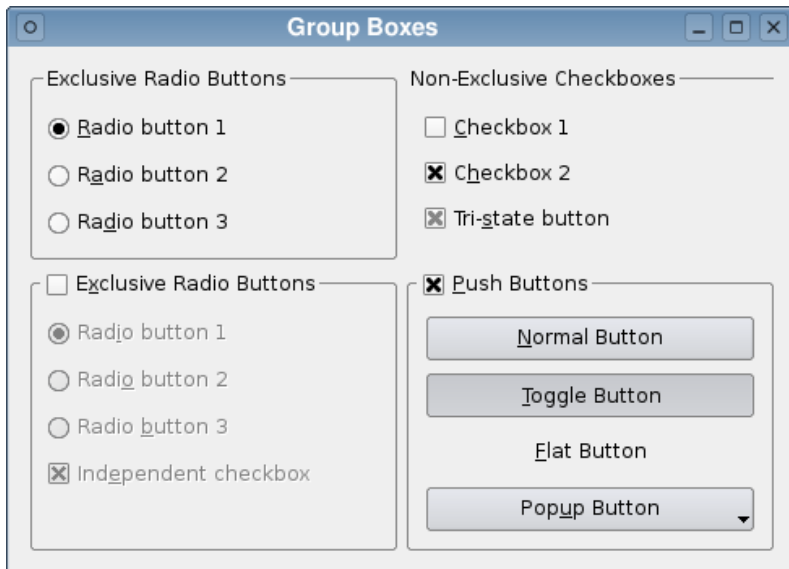
Les exemples suivants sont extraits du document Qt-Widgets-Layouts dans le répertoire de documentation.

-QtVBoxLayout (contient une ComboBox), le tout configuré dans le main du programme principal.



Exemple sur Widget Group boxes (extrait de [https://stuff.mit.edu/afs/athena/software/texmaker\\_v5.0.2/qt57/doc/qtwidgets/qtwidgets-widgets-groupbox-example.html](https://stuff.mit.edu/afs/athena/software/texmaker_v5.0.2/qt57/doc/qtwidgets/qtwidgets-widgets-groupbox-example.html))

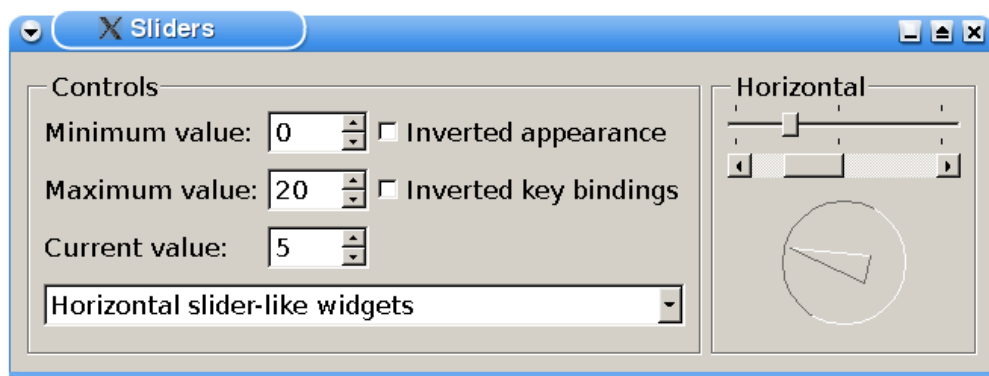
Group boxes are usually used to organize check boxes and radio buttons into exclusive groups.



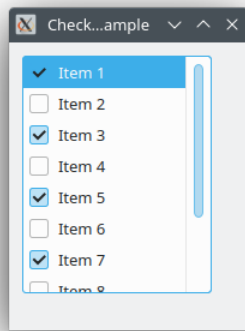
The Group Boxes example consists of a single Window class that is used to show four group boxes: an exclusive radio button group, a non-exclusive checkbox group, an exclusive radio button group with an enabling checkbox, and a group box with normal push buttons.

#### Exemple de Qt-sliderGroup

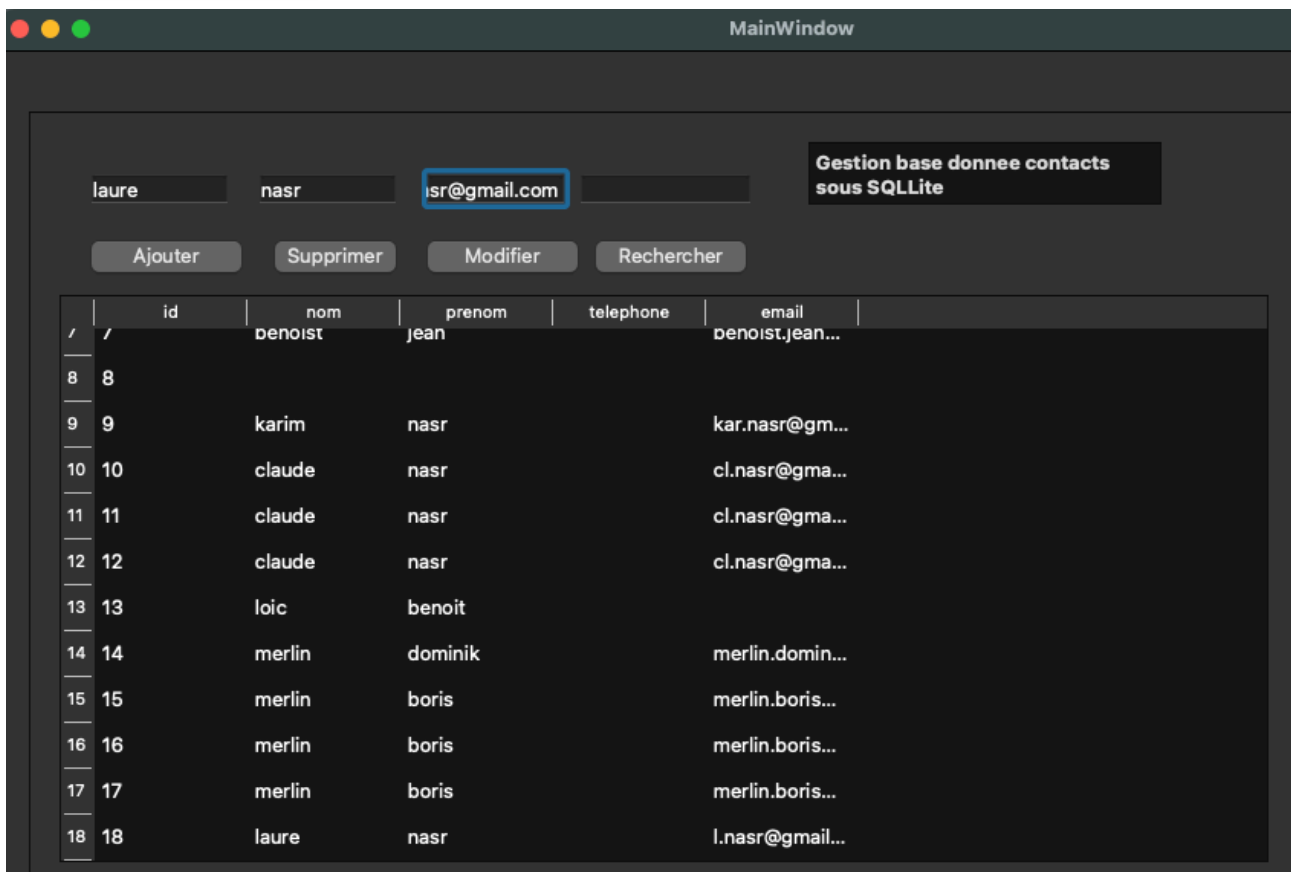
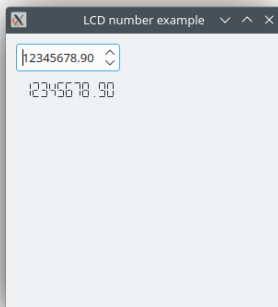
The example also demonstrates how signals and slots can be used to synchronize the behavior of two or more widgets.



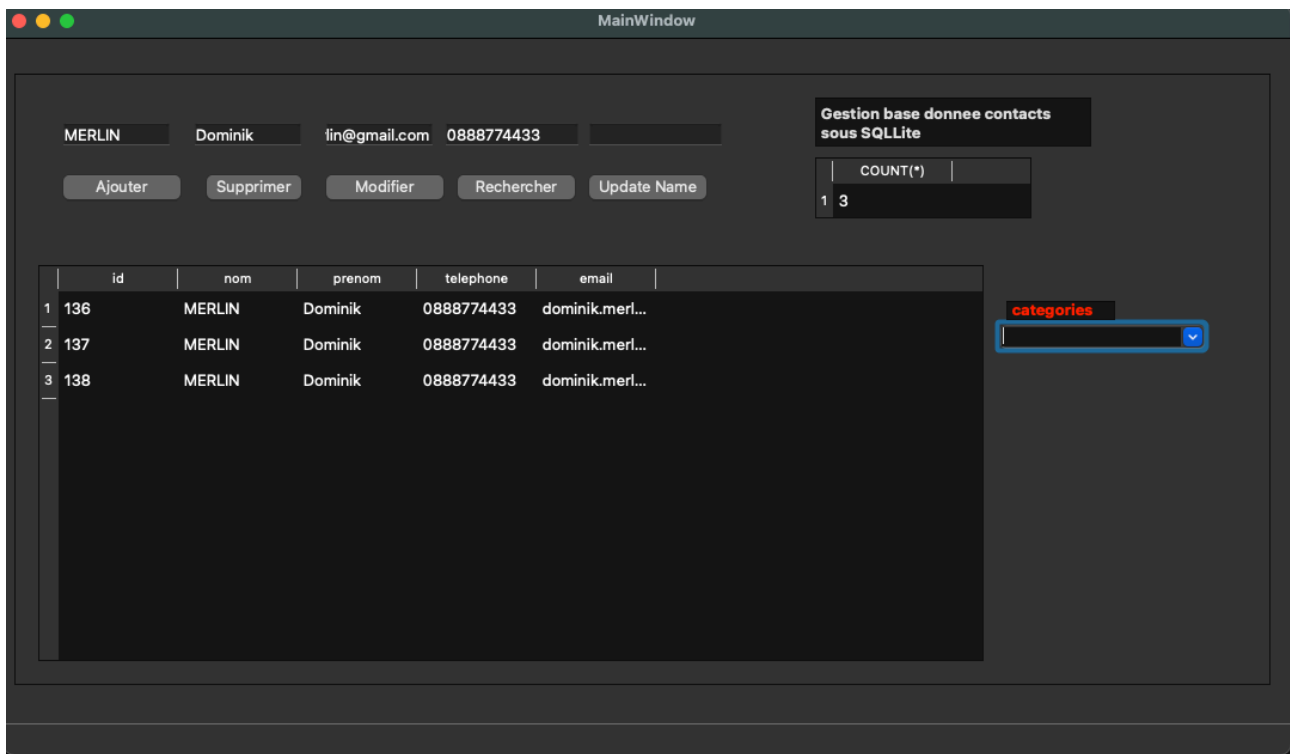
QtChecked List Wiget est extrait de [https://github.com/gammasoft71/Examples\\_Qt/blob/master/Qt.Widgets/Controls/CheckedListWidget/README.md](https://github.com/gammasoft71/Examples_Qt/blob/master/Qt.Widgets/Controls/CheckedListWidget/README.md)



QtLCDnumber



Le projet GestionContact est une IHM pour afficher une base de donnee de type SQLite utilisation de widget SQLite, utilisation à l'aide de GEMINI.



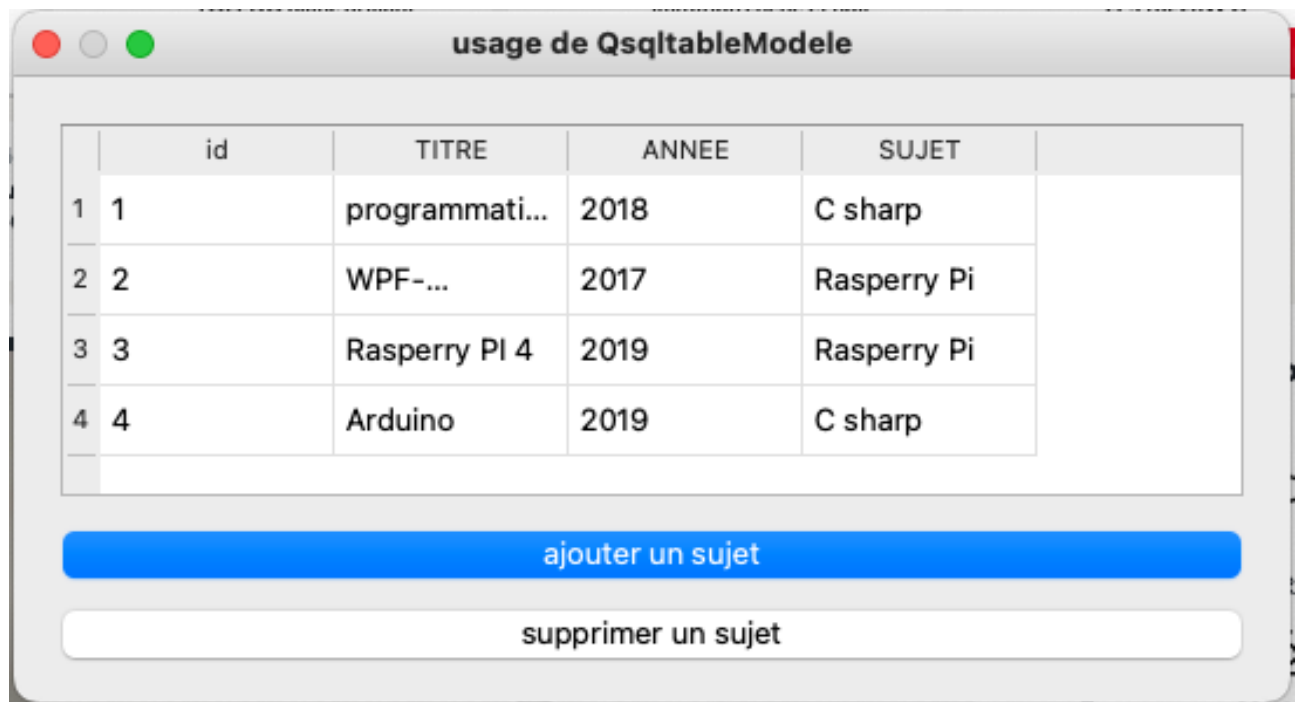
La deuxieme version integre la rubrique categorie intégrée à l aide d'une comboBox.

Exemple de Qt-horloge digitale (fichier PDF de TP joint)





Exemple QSqlTableModel , fait avec pychart , extrait cours Qt sur livre Framework PyQt (benoit prieur)



Usage de la classe QSqlRelationalTableModel (chapitre 8)

/Users/nasr/PycharmProjects/PySide/main.py-Qt-QsqlRelationalModel

```
import os
import PyQt6
```

```
from PyQt6 import QtSql
from PyQt6 import QtCore
from PyQt6 import QtGui
```

```
from PyQt6.QtSql import *
from PyQt6.QtCore import *
from PyQt6.QtGui import *
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import *
#from PyQt6 import QtWidgets
```

```
from PySide6.QtWidgets import QApplication, QLabel, QWidget,
QBoxLayout, QVBoxLayout, QTableView, QDialog, QPushButton
```

```
from PySide6.QtGui import QApplication
```

```
from PySide6.QtCore import QApplication, Qt
```

```
from PySide6.QtSql import
QSqlDatabase, QSqlRelationalTableModel, QSqlRelation, QSqlRelati
onalDelegate
```

```
from PySide6.QtSql import QSqlTableModel
```

```
class FenetreSimple(QWidget):
```

```
    def __init__(self):
```

```
        super().__init__()
```

```
        self.disposition = QVBoxLayout()
```

```
        self.clickbouton = QPushButton("Click",
clicked=self.creationDB)
```

```
        self.disposition.addWidget(self.clickbouton)
```

```
        self.setLayout(self.disposition)
```

```
        self.execute()
```

```
    #QtGui.QWindow.__init__(self, parent)
```

```
    self.resize(30, 30)
```

```
    #self.setFont(QtGui.QFont("Verdana"))
```

```
    self.setWindowTitle("Bases de données")
```

```
#         self.clickbouton =  
QPushButton("Click",clicked=self.creationDB)  
  
#         self.disposition.addWidget(self.clickbouton)  
        self.setLayout(self.disposition)  
  
#         self.execute()
```

```
def creationDB(self):
```

```
    self.db = QSql.QSqlDatabase.addDatabase('QSQLITE')  
    self.db.setDatabaseName('Baselivres.db')  
    print ("Creation base ok")  
    if not self.db.open():  
        print ("la Db ne peut pas s'ouvrir.")  
        return False
```

```
    query = QSql.QSqlQuery()
```

```
    print(query.exec("SELECT COUNT(*) FROM PERSONNE"))  
    while query.next():  
        print(query.value(0))  
        query.exec("INSERT INTO PERSONNE('NOM','PRENOM')  
VALUES('Prieur','Benoit')")  
        query.exec("INSERT INTO PERSONNE('NOM','PRENOM')  
VALUES('Mocq','François')")  
        query.exec("INSERT INTO PERSONNE('NOM','PRENOM')  
VALUES('Lacaze','Sarah')")
```

```
print(query.exec("SELECT COUNT(*) FROM  
PERSONNE"))  
  
while query.next():  
    print(query.value(0))
```

```
print(query.exec("SELECT COUNT(*) FROM SUJET"))  
while query.next():  
    print(query.value(0))  
    query.exec("INSERT INTO SUJET('SUJET')  
VALUES('Csharp')")  
    query.exec("INSERT INTO SUJET('SUJET')  
VALUES('Raspberry pi')")  
    query.exec("INSERT INTO SUJET('SUJET')  
VALUES('Scratch')")
```

```
print(query.exec("SELECT COUNT(*) FROM SUJET"))  
while query.next():  
    print(query.value(0))
```

```
print(query.exec("SELECT COUNT(*) FROM LIVRE"))  
while query.next():  
    print(query.value(0))  
    query.exec("INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation  
en C-preparation aux certifications MCSA-examen  
70-483',2018,1,1)")
```

```
        query.exec("INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper  
des applications structurees',2017,1,2)")
```

```
        query.exec("INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Raspberry PI  
4',2019,2,3)")
```

```
        query.exec("INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id')  
VALUES('Arduino',2019,1,4)")
```

```
        print(query.exec("SELECT COUNT(*) FROM LIVRE"))  
        while query.next():  
            print(query.value(0))
```

```
        query = QSqlQuery()  
        query.exec("UPDATE SUJET SET SUJET = 'MICROSOFT C#'  
WHERE id =2")
```

```
        query.exec("SELECT COUNT(*) FROM SUJET")  
        while query.next():  
            print(query.value(0))
```

```
        query.exec("SELECT SUJET FROM SUJET WHERE id = 2")  
        while query.next():  
            print(query.value(0))
```

```
        self.db.commit()  
        self.db.close()
```

```
#Creation des requêtes SQL
```

```
creationTableLivre = """create table LIVRE(  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    TITRE TEXT NOT NULL,  
    ANNEE int,  
    sujet_id INTEGER REFERENCES SUJET(id)  
);"""
```

```
creationTableSujet = """create table SUJET(  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    SUJET TEXT NOT NULL  
);"""
```

```
creationTablePersonne = """create table PERSONNE(  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    NOM TEXT NOT NULL,  
    PRENOM TEXT NOT NULL  
);"""
```

```
#Execution des requetes SQL
```

```
self.db.open()  
query = QSqlQuery()  
#self.db.close()
```

```
if query.exec(creationTableLivre):  
    print("creation table LIVRE ok")
```

```
if query.exec(creationTableSujet):
```

```
print("creation table SUJET ok")
```

```
if query.exec(creationTablePersonne):  
    print("creation table Personne ok")
```

```
self.db.close()
```

```
def execute (self):  
    self.resize(250, 300)  
    self.move(50, 500)  
    self.setWindowTitle("chapitre 8 - insertion de  
donnees")  
    self.show()
```

```
def joinDB():  
    app = QtCore.QCoreApplication(sys.argv)
```

```
db = QSql.QSqlDatabase.addDatabase('QSQLITE')  
db.setDatabaseName('Baselivres.db')  
print("connexion base ok")
```

```
if not db.open():  
    print("la db ne peut s'ouvrir.")  
    return False
```

```
query = QSql.QSqlQuery()
```

```

        #query.exec(
            #      "INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation
en C-preparation aux certifications MCSA-examen
70-483',2018,1,1)")

        #query.exec(
            #      "INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper
des applications structurees',2017,1,2)")

        #query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Raspberry PI
4',2019,2,3)")

        #query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id')
VALUES('Arduino',2019,1,4)")

```

```

#query.exec( ""
#      SELECT PERSONNE.PRENOM, PERSONNE.NOM
#      FROM PERSONNE, LIVRE, SUJET
#      WHERE PERSONNE.id = LIVRE.id
#      AND SUJET.id = LIVRE.id
#      AND SUJET.SUJET = 'Arduino'
#      AND LIVRE.ANNEE = 2019
#""")

```

```

#affiche jointure
#connexion
#base ok

```



```
#Benoit Prieur  
#François Mocq  
#Sarah Lacaze
```

```
sujet = 'Scratch'  
annee = 2019
```

```
requete=" "
```

```
SELECT PERSONNE.PRENOM, PERSONNE.NOM  
FROM PERSONNE, LIVRE, SUJET  
WHERE PERSONNE.id = LIVRE.id  
AND SUJET.id = LIVRE.id  
"
```

```
query.prepare(requete)  
query.bindValue(":sujet",sujet)  
query.bindValue(":annee",annee)
```

```
if query.exec():  
    while query.next():  
        print(query.value(0),query.value(1))  
else:  
    print("erreur dans l execution de la  
requete:",query.lastError().text())
```

```
query.exec( " "
```

```
SELECT PERSONNE.PRENOM, PERSONNE.NOM
```

```
FROM PERSONNE, LIVRE, SUJET
WHERE PERSONNE.id = LIVRE.id
AND SUJET.id = LIVRE.id
""")
```

```
while query.next():
    print(query.value(0),query.value(1))
```

```
def print_hi(name):
    # Use a breakpoint in the code line below to debug your
    script.
    print(f'Hi, {name}') # Press ⌘F8 to toggle the
    breakpoint.
```

```
# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    # affiche jointure
    # connexion
    # base ok
    # Benoit Prieur
    # François Mocq
    # Sarah Lacaze
    #joinDB()
    #Application.instance()
    app = QApplication(sys.argv)
```

```
base = QSql.QSqlDatabase.addDatabase('QSQLITE')
base.setDatabaseName('Baselivres.db')
```

```
#definition du modele
#modele = QtCore.QAbstractItemModel
modele = QSqlRelationalTableModel()
modele.setTable('LIVRE')
```

```
modele.setEditStrategy(QSqlRelationalTableModel.EditStrategy.
OnFieldChange)
```

```
modele.setRelation(3,QSqlRelation("SUJET","id","SUJET"))
modele.setRelation(4, QSqlRelation("PERSONNE", "id",
"PRENOM"))
```

```
#modification immediate
#modele.select()
modele.setHeaderData(3,Qt.Horizontal,"SUJET")
modele.setHeaderData(3, Qt.Horizontal, "PRENOM")
#modele.setHeaderData(3,
QtCore.Qt.Orientation.Horizontal, "SUJET")
```

```
#modele.setHeaderData(4,QtCore.Qt.Orientation.Horizontal,"PRE
NOM")
modele.select()
```

```
#creation de la vue et association au modele
vue = QTableView()
```

```
vue.setModel(modele)
vue.setItemDelegate(QSqlRelationalDelegate(vue))
```

```
#dialogue = QDialog()
#disposition = QVBoxLayout()
#disposition.addWidget(vue)
```

```
#fenetre = QtWidgets.QWidget()
```

```
#fenetre.QTableView().setModel(QtSql.QSqlTableModel())
#vue.setItemDelegate(QSqlRelationalDelegate(vue))
```

```
#creation de la boite de dialogue
dialogue = QDialog()
disposition = QVBoxLayout()
disposition.addWidget(vue)
```

```
#bouton d ajout
bouton_ajout = QPushButton("ajouter un livre")
```

```
bouton_ajout.clicked.connect(lambda :modele.insertRows(modele
.rowCount(),1))
disposition.addWidget(bouton_ajout)
```

```
#bouton de suppression
bouton_suppression = QPushButton("supprimer un livre")
bouton_suppression.clicked.connect(lambda :
modele.removeRow(vue.currentIndex().row()))
```

```
disposition.addWidget(bouton_suppression)
```

```
#finalisation de l interface  
dialogue.setLayout(disposition)  
dialogue.setWindowTitle("usage de QsqRelationalModele")  
dialogue.show()
```

```
#app = QtCore.QCoreApplication(sys.argv)
```

```
#app = QtCore.QCoreApplication(sys.argv)  
#app = QtGui.QGuiApplication(sys.argv)  
#app = QtCore.QCoreApplication(sys.argv)
```

```
#app= QtCore.QCoreApplication.arguments()
```

```
# ...  
sys.exit(app.exec())
```

```
import sys  
import os  
import PyQt6
```

```
from PyQt6 import QtSql
```

```
from PyQt6 import QtCore
from PyQt6 import QtGui
```

```
from PyQt6.QtSql import *
from PyQt6.QtCore import *
from PyQt6.QtGui import *
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import *
#from PyQt6 import QtWidget
```

```
from PySide6.QtWidgets import QApplication, QLabel, QWidget,
QBoxLayout, QVBoxLayout, QTableView, QDialog, QPushButton
```

```
from PySide6.QtGui import QtGuiApplication
from PySide6.QtCore import QApplication, Qt
from PySide6.QtSql import
QSqlDatabase, QSqlRelationalTableModel, QSqlRelation, QSqlRelati
onalDelegate
from PySide6.QtSql import QSqlTableModel
```

```
class FenetreSimple(QWidget):
    def __init__(self):
        super().__init__()
```

```
        self.disposition = QVBoxLayout()
```

```
        self.clickbouton = QPushButton("Click",
clicked=self.creationDB)

        self.disposition.addWidget(self.clickbouton)

        self.setLayout(self.disposition)

        self.execute()
```

```
#QtGui.QWindow.__init__(self,parent)

self.resize(30, 30)

#self.setFont(QtGui.QFont("Verdana"))

self.setWindowTitle("Bases de données")
```

```
#        self.clickbouton =
QPushButton("Click",clicked=self.creationDB)

#        self.disposition.addWidget(self.clickbouton)

        self.setLayout(self.disposition)

#        self.execute()
```

```
def creationDB(self):
```

```
        self.db = QSqlDatabase.addDatabase('QSQLITE')

        self.db.setDatabaseName('Baselivres.db')

        print ("Creation base ok")

        if not self.db.open():

            print ("la Db ne peut pas s'ouvrir.")

            return False
```

```
query = QSqlQuery()
```

```
print(query.exec("SELECT COUNT(*) FROM PERSONNE"))
while query.next():
    print(query.value(0))
    query.exec("INSERT INTO PERSONNE('NOM', 'PRENOM')
VALUES('Prieur', 'Benoit')")
    query.exec("INSERT INTO PERSONNE('NOM', 'PRENOM')
VALUES('Mocq', 'François')")
    query.exec("INSERT INTO PERSONNE('NOM', 'PRENOM')
VALUES('Lacaze', 'Sarah')")
```

```
print(query.exec("SELECT COUNT(*) FROM
PERSONNE"))
while query.next():
    print(query.value(0))
```

```
print(query.exec("SELECT COUNT(*) FROM SUJET"))
while query.next():
    print(query.value(0))
    query.exec("INSERT INTO SUJET('SUJET')
VALUES('Csharp')")
    query.exec("INSERT INTO SUJET('SUJET')
VALUES('Raspberry pi')")
    query.exec("INSERT INTO SUJET('SUJET')
VALUES('Scratch')")
```

```
print(query.exec("SELECT COUNT(*) FROM SUJET"))
while query.next():
```



```
print(query.value(0))
```

```
print(query.exec("SELECT COUNT(*) FROM LIVRE"))
while query.next():
    print(query.value(0))
    query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation
en C-preparation aux certifications MCSA-examen
70-483',2018,1,1)")
    query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper
des applications structurees',2017,1,2)")
    query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Raspberry PI
4',2019,2,3)")
    query.exec("INSERT INTO
LIVRE('TITRE','ANNEE','sujet_id','id')
VALUES('Arduino',2019,1,4)")
```

```
print(query.exec("SELECT COUNT(*) FROM LIVRE"))
while query.next():
    print(query.value(0))
```

```
query = QSql.QSqlQuery()
query.exec("UPDATE SUJET SET SUJET = 'MICROSOFT C#'
WHERE id =2")
```

```
query.exec("SELECT COUNT(*) FROM SUJET")
while query.next():
    print(query.value(0))
```

```
query.exec("SELECT SUJET FROM SUJET WHERE id = 2")
while query.next():
    print(query.value(0))
```

```
self.db.commit()
self.db.close()

#Creation des requêtes SQL
creationTableLivre = """create table LIVRE(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    TITRE TEXT NOT NULL,
    ANNEE int,
    sujet_id INTEGER REFERENCES SUJET(id)
);"""
```

```
creationTableSujet = """create table SUJET(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    SUJET TEXT NOT NULL
);"""

creationTablePersonne = """create table PERSONNE(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    NOM TEXT NOT NULL,
    PRENOM TEXT NOT NULL
);"""
```

```
#Execution des requetes SQL
```

```
self.db.open()  
  
query = QSql.QSqlQuery()  
  
#self.db.close()
```

```
if query.exec(creationTableLivre):  
    print("creation table LIVRE ok")
```

```
if query.exec(creationTableSujet):  
    print("creation table SUJET ok")
```

```
if query.exec(creationTablePersonne):  
    print("creation table Personne ok")
```

```
self.db.close()
```

```
def execute (self):  
    self.resize(250, 300)  
    self.move(50, 500)  
    self.setWindowTitle("chapitre 8 - insertion de  
donnees")  
    self.show()
```

```
def joinDB():  
    app = QtCore.QCoreApplication(sys.argv)
```

```
db = QSql.QSqlDatabase.addDatabase('QSQLITE')
```

```
db.setDatabaseName('Baselivres.db')  
print("connexion base ok")
```

```
if not db.open():  
    print("la db ne peut s'ouvrir.")  
    return False
```

```
query = QSqlQuery()
```

```
#query.exec(  
#    "INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('programmation  
en C-preparation aux certifications MCSA-examen  
70-483',2018,1,1)")  
#query.exec(  
#    "INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-developper  
des applications structurees',2017,1,2)")  
#query.exec("INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('Raspberry PI  
4',2019,2,3)")  
#query.exec("INSERT INTO  
LIVRE('TITRE','ANNEE','sujet_id','id')  
VALUES('Arduino',2019,1,4)")
```

```
#query.exec(""  
#    SELECT PERSONNE.PRENOM, PERSONNE.NOM  
#    FROM PERSONNE, LIVRE, SUJET  
#    WHERE PERSONNE.id = LIVRE.id
```

```
# AND SUJET.id = LIVRE.id
# AND SUJET.SUJET = 'Arduino'
# AND LIVRE.ANNEE = 2019
#"")
```

```
#affiche jointure
#connexion
#base ok
#Benoit Prieur
#François Mocq
#Sarah Lacaze
```

```
sujet = 'Scratch'
annee = 2019
```

```
requete=" "
```

```
SELECT PERSONNE.PRENOM, PERSONNE.NOM
FROM PERSONNE, LIVRE, SUJET
WHERE PERSONNE.id = LIVRE.id
AND SUJET.id = LIVRE.id
"
```

```
query.prepare(requete)
query.bindValue(":sujet",sujet)
query.bindValue(":annee",annee)
```

```
if query.exec():
```

```

        while query.next():
            print(query.value(0),query.value(1))
        else:
            print("erreur dans l execution de la
requete:",query.lastError().text())

```

```

query.exec( """
        SELECT PERSONNE.PRENOM, PERSONNE.NOM
        FROM PERSONNE, LIVRE, SUJET
        WHERE PERSONNE.id = LIVRE.id
        AND SUJET.id = LIVRE.id
        """ )

```

```

        while query.next():
            print(query.value(0),query.value(1))

```

```

def print_hi(name):
    # Use a breakpoint in the code line below to debug your
script.
    print(f'Hi, {name}') # Press ⌘F8 to toggle the
breakpoint.

```

```

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    # affiche jointure
    # connexion
    # base ok

```

```
# Benoit Prieur
# François Mocq
# Sarah Lacaze
#joinDB()
#Application.instance()
app = QApplication(sys.argv)
```

```
base = QSql.QSqlDatabase.addDatabase('QSQLITE')
base.setDatabaseName('Baselivres.db')
```

```
#definition du modele
#modele = QtCore.QAbstractItemModel
modele = QSqlRelationalTableModel()
modele.setTable('LIVRE')
```

```
modele.setEditStrategy(QSqlRelationalTableModel.EditStrategy.
OnFieldChange)
```

```
modele.setRelation(3,QSqlRelation("SUJET","id","SUJET"))
modele.setRelation(4, QSqlRelation("PERSONNE", "id",
"PRENOM"))
```

```
#modification immediate
#modele.select()
modele.setHeaderData(3,Qt.Horizontal,"SUJET")
modele.setHeaderData(3, Qt.Horizontal, "PRENOM")
#modele.setHeaderData(3,
QtCore.Qt.Orientation.Horizontal, "SUJET")
```

```
#modele.setHeaderData(4,QtCore.Qt.Orientation.Horizontal,"PRE  
NOM")
```

```
modele.select()
```

```
#creation de la vue et association au modele
```

```
vue = QTableView()
```

```
vue.setModel(modele)
```

```
vue.setItemDelegate(QSqlRelationalDelegate(vue))
```

```
#dialogue = QDialog()
```

```
#disposition = QVBoxLayout()
```

```
#disposition.addWidget(vue)
```

```
#fenetre = QWidget()
```

```
#fenetre.QTableView().setModel(QSql.QSqlTableModel())
```

```
#vue.setItemDelegate(QSqlRelationalDelegate(vue))
```

```
#creation de la boite de dialogue
```

```
dialogue = QDialog()
```

```
disposition = QVBoxLayout()
```

```
disposition.addWidget(vue)
```

```
#bouton d ajout
```

```
bouton_ajout = QPushButton("ajouter un livre")
```



```
bouton_ajout.clicked.connect(lambda :modele.insertRows(modele
    .rowCount(),1))
    disposition.addWidget(bouton_ajout)
```

```
#bouton de suppression
    bouton_suppression = QPushButton("supprimer un livre")
    bouton_suppression.clicked.connect(lambda :
modele.removeRow(vue.currentIndex().row()))
    disposition.addWidget(bouton_suppression)
```

```
#finalisation de l interface
    dialogue.setLayout(disposition)
    dialogue.setWindowTitle("usage de QSqlRelationalModele")
    dialogue.show()
```

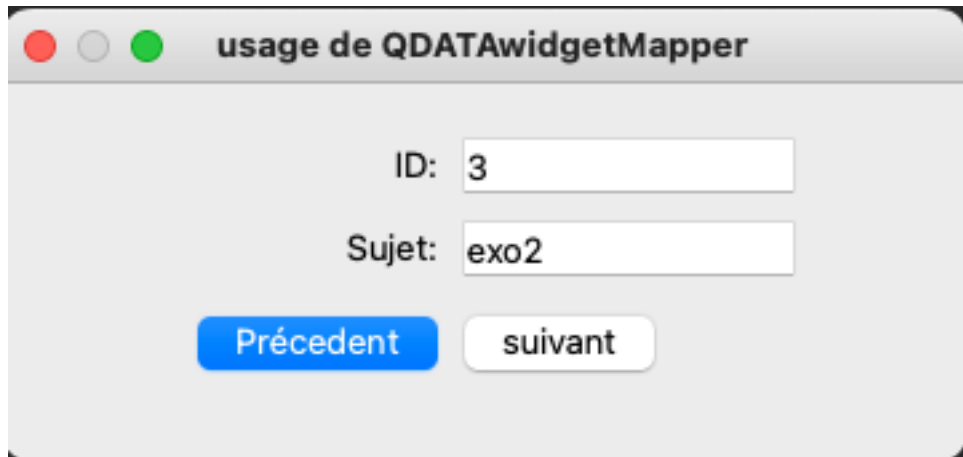
```
#app = QtCore.QCoreApplication(sys.argv)
```

```
#app = QtCore.QCoreApplication(sys.argv)
#app = QtGui.QGuiApplication(sys.argv)
#app = QtCore.QCoreApplication(sys.argv)
```

```
#app= QtCore.QCoreApplication.arguments()
```

```
# ...
    sys.exit(app.exec())
```

Exemple QDATAWidgetMapper , fait avec pychart , extrait cours Qt sur livre Framework PyQt (benoit prieur)



```
import sys
import os
import PyQt6
```

```
from PyQt6 import QSql
from PyQt6 import QtCore
from PyQt6 import QtGui
```

```
from PyQt6.QtSql import *
from PyQt6.QtCore import *
from PyQt6.QtGui import *
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import *
#from PyQt6 import QtWidget
```

```
from PySide6.QtWidgets import QApplication, QLabel, QWidget,
QBoxLayout, QVBoxLayout, QTableView, QDialog, QLineEdit,
QPushButton, QFormLayout, QDataWidgetMapper
```

```
from PySide6.QtGui import QtGuiApplication
from PySide6.QtCore import QApplication, Qt
```

```
from PySide6.QtSql import
QSqlDatabase, QSqlRelationalTableModel, QSqlRelation, QSqlRelationalDelegate
from PySide6.QtSql import QSqlTableModel
```

```
class FenetreSimple(QWidget):
    def __init__(self):
        super().__init__()
```

```
        self.disposition = QVBoxLayout()
        self.clickbouton = QPushButton("Click", clicked=self.creationDB)
        self.disposition.addWidget(self.clickbouton)
        self.setLayout(self.disposition)
        self.execute()
```

```
        #QtGui.QWindow.__init__(self,parent)
        self.resize(30, 30)
        #self.setFont(QtGui.QFont("Verdana"))
        self.setWindowTitle("Bases de données")
```

```
#        self.clickbouton = QPushButton("Click",clicked=self.creationDB)
#        self.disposition.addWidget(self.clickbouton)
        self.setLayout(self.disposition)
#        self.execute()
```

```
    def creationDB(self):
```

```
        self.db = QSqlDatabase.addDatabase('QSQLITE')
        self.db.setDatabaseName('Baselivres.db')
        print ("Creation base ok")
        if not self.db.open():
            print ("la Db ne peut pas s'ouvrir.")
            return False
```

```
query = QSqlQuery()
```

```
print(query.exec("SELECT COUNT(*) FROM PERSONNE"))  
while query.next():  
    print(query.value(0))  
    query.exec("INSERT INTO PERSONNE('NOM','PRENOM') VALUES('Prieur','Benoit')")  
    query.exec("INSERT INTO PERSONNE('NOM','PRENOM') VALUES('Mocq','François')")  
    query.exec("INSERT INTO PERSONNE('NOM','PRENOM') VALUES('Lacaze','Sarah')")
```

```
print(query.exec("SELECT COUNT(*) FROM PERSONNE"))  
while query.next():  
    print(query.value(0))
```

```
print(query.exec("SELECT COUNT(*) FROM SUJET"))  
while query.next():  
    print(query.value(0))  
    query.exec("INSERT INTO SUJET('SUJET') VALUES('Csharp')")  
    query.exec("INSERT INTO SUJET('SUJET') VALUES('Rasperry pi')")  
    query.exec("INSERT INTO SUJET('SUJET') VALUES('Scratch')")
```

```
print(query.exec("SELECT COUNT(*) FROM SUJET"))  
while query.next():  
    print(query.value(0))
```

```
print(query.exec("SELECT COUNT(*) FROM LIVRE"))  
while query.next():  
    print(query.value(0))  
    query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id')  
VALUES('programmation en C-preparation aux certifications MCSA-examen 70-483',2018,1,1)")  
    query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id') VALUES('WPF-  
developper des applications structurees',2017,1,2)")  
    query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id')  
VALUES('Rasperry PI 4',2019,2,3)")  
    query.exec("INSERT INTO LIVRE('TITRE','ANNEE','sujet_id','id')  
VALUES('Arduino',2019,1,4)")
```

```
print(query.exec("SELECT COUNT(*) FROM LIVRE"))  
  
while query.next():  
    print(query.value(0))
```

```
query = QSqlQuery()  
  
query.exec("UPDATE SUJET SET SUJET = 'MICROSOFT C#' WHERE id =2")
```

```
query.exec("SELECT COUNT(*) FROM SUJET")  
  
while query.next():  
    print(query.value(0))
```

```
query.exec("SELECT SUJET FROM SUJET WHERE id = 2")  
  
while query.next():  
    print(query.value(0))
```

```
self.db.commit()  
  
self.db.close()  
  
#Creation des requêtes SQL  
  
creationTableLivre = """create table LIVRE(  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    TITRE TEXT NOT NULL,  
    ANNEE int,  
    sujet_id INTEGER REFERENCES SUJET(id)  
);"""
```

```
creationTableSujet = """create table SUJET(  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    SUJET TEXT NOT NULL  
);"""
```

```
creationTablePersonne = """create table PERSONNE(  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    NOM TEXT NOT NULL,  
    PRENOM TEXT NOT NULL
```

```
);
```

```
#Execution des requetes SQL
```

```
self.db.open()
```

```
query = QSqlQuery()
```

```
#self.db.close()
```

```
if query.exec(creationTableLivre):
```

```
    print("creation table LIVRE ok")
```

```
if query.exec(creationTableSujet):
```

```
    print("creation table SUJET ok")
```

```
if query.exec(creationTablePersonne):
```

```
    print("creation table Personne ok")
```

```
self.db.close()
```

```
def execute (self):
```

```
    self.resize(250, 300)
```

```
    self.move(50, 500)
```

```
    self.setWindowTitle("chapitre 8 - insertion de donnees")
```

```
    self.show()
```

```
def prec():
```

```
    print("Bouton précédent")
```

```
    mapping.toPrevious()
```

```
    id.repaint()
```

```
    sujet.repaint()
```

```
def suiv():
```

```
    print("Bouton suivant")
```

```
    mapping.toNext()
```

```
id.repaint()

sujet.repaint()
```

```
def joinDB():

    app = QtCore.QCoreApplication(sys.argv)
```

```
    db = QSql.QSqlDatabase.addDatabase('QSQLITE')

    db.setDatabaseName('Baselivres.db')

    print("connexion base ok")
```

```
    if not db.open():

        print("la db ne peut s'ouvrir.")

        return False
```

```
    query = QSql.QSqlQuery()
```

```
    sujet = 'Scratch'

    annee = 2019
```

```
    requete=" "

    SELECT PERSONNE.PRENOM, PERSONNE.NOM

    FROM PERSONNE, LIVRE, SUJET

    WHERE PERSONNE.id = LIVRE.id

    AND SUJET.id = LIVRE.id

    " " "
```

```
    query.prepare(requete)

    query.bindValue(":sujet",sujet)

    query.bindValue(":annee",annee)
```

```
    if query.exec():

        while query.next():
```

```

        print(query.value(0),query.value(1))
    else:
        print("erreur dans l execution de la requete:",query.lastError().text())

```

```

query.exec("
SELECT PERSONNE.PRENOM, PERSONNE.NOM
FROM PERSONNE, LIVRE, SUJET
WHERE PERSONNE.id = LIVRE.id
AND SUJET.id = LIVRE.id
")

```

```

while query.next():
    print(query.value(0),query.value(1))

```

```

def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}') # Press F8 to toggle the breakpoint.

```

```

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    # affiche jointure
    # connexion
    # base ok
    # Benoit Prieur
    # François Moccia
    # Sarah Lacaze
    #joinDB()
    #Application.instance()
    app = QApplication(sys.argv)

```

```

    # creation de la boite de dialogue
    dialogue = QDialog()
    disposition = QFormLayout()

```



```
#champ d'édition  
  
id = QLineEdit()  
id.setReadOnly(True)  
  
sujet = QLineEdit()
```

```
disposition.addRow("ID:",id)  
disposition.addRow("Sujet:",sujet)
```

```
precedentBouton = QPushButton("Précédent")  
suivantBouton = QPushButton("suivant")  
disposition.addRow(precedentBouton,suivantBouton)
```

```
dialogue.setLayout(disposition)  
dialogue.setWindowTitle("usage de QDataWidgetMapper")
```

```
base = QSqlDatabase.addDatabase('QSQLITE')  
base.setDatabaseName('Baselivres.db')
```

```
#definition du modele  
  
#modele = QtCore.QAbstractItemModel  
  
modele = QSqlTableModel()  
modele.setTable('SUJET')  
  
modele.setEditStrategy(QSqlTableModel.EditStrategy.OnFieldChange)
```

```
#modification immediate  
  
modele.select()
```

```
mapping = QDataWidgetMapper()  
mapping.setModel(modele)  
mapping.addMapping(id, 0)  
mapping.addMapping(sujet, 1)  
mapping.toFirst()
```

```
precedentBouton.clicked.connect(prec)
suivantBouton.clicked.connect(suiv)
```

```
modele.select()
```

```
dialogue.show()
```

```
# ...
sys.exit(app.exec())
```

Exemple QTCreationFormulaire , fait avec pychart , extrait cours Qt sur livre Framework PyQt (benoit prier)

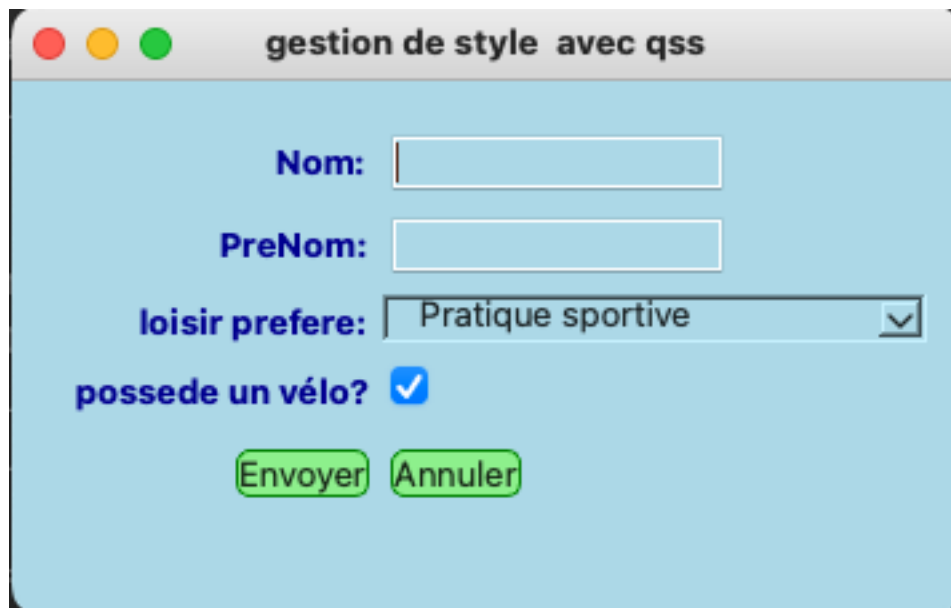
The screenshot shows a Qt window titled "chapitre 3 - formulaire". Inside the window, there is a form with the following elements:

- A label "Nom:" followed by a text input field.
- A label "PreNom:" followed by a text input field.
- A label "loisir prefere:" followed by a combobox showing "Pratique sportive".
- A label "possede un vélo?" followed by a checked checkbox.
- Two buttons at the bottom: "Envoyer" and "Annuler".

dentique au cas précédent avec QFont et nouvelle police

This screenshot shows the same Qt window as the previous one, but with the text labels rendered in a bold, italicized font using the QFont class. The labels are now "Nom:", "PreNom:", "Loisir Prefere:", and "Possede Un Vélo?". The input fields, combobox, checkbox, and buttons remain the same.

Exemple QT-Gestion Feuille Style (voir exemple avec QBRUSH) extrait cours Qt sur livre Framework PyQt (benoit prieur)



The image shows a Qt application window with a light blue background and a title bar containing three colored buttons (red, yellow, green) and the text "gestion de style avec qss". The window contains the following elements:

- A label "Nom:" followed by a text input field.
- A label "PreNom:" followed by a text input field.
- A label "loisir prefere:" followed by a combobox showing "Pratique sportive" with a dropdown arrow.
- A label "possede un vélo?" followed by a checked checkbox.
- Two buttons at the bottom: "Envoyer" and "Annuler", both with a green gradient and rounded corners.

Exemple QT-Pyside-Signal&slot chapitre 4 du livre Framework PyQt gere la fermeture de la fenetre avec appel au slot quand clique sur bouton fermer.