

Auteur: karim NASR Date: 23/11/2025

Objet du document:

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

Historique:

18/09/024 KNA ajout de QtThreeWidgetTutorial.Exemple sur les checkbox.

18/09/024 KNA, évolution de la première version, ajout de LCD et de la somme de la commande. 20/09/024 KNA, ajout de nouveaux projet DisplayWindows et ReadingWritingTextFile. 12/10/2024 KNA ajout exemple sur FontDialog-utilisateur change la police de caractere dans fenêtre, InputDialogMultiline, ToolBar.

Début 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 données 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 à 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.

21/11/2025 KNA ajout Qt_StateMachine utilisation de QTimer pour afficher des fenêtres temporisées. Ajout de projet Qt-timer (scheduler de fenêtre avec QTimer).

23/11/2025 Ajout de LedIndicatorWidget simulant des feux R/V pendant 1 minute.

QtThreeWidgetTutorial:

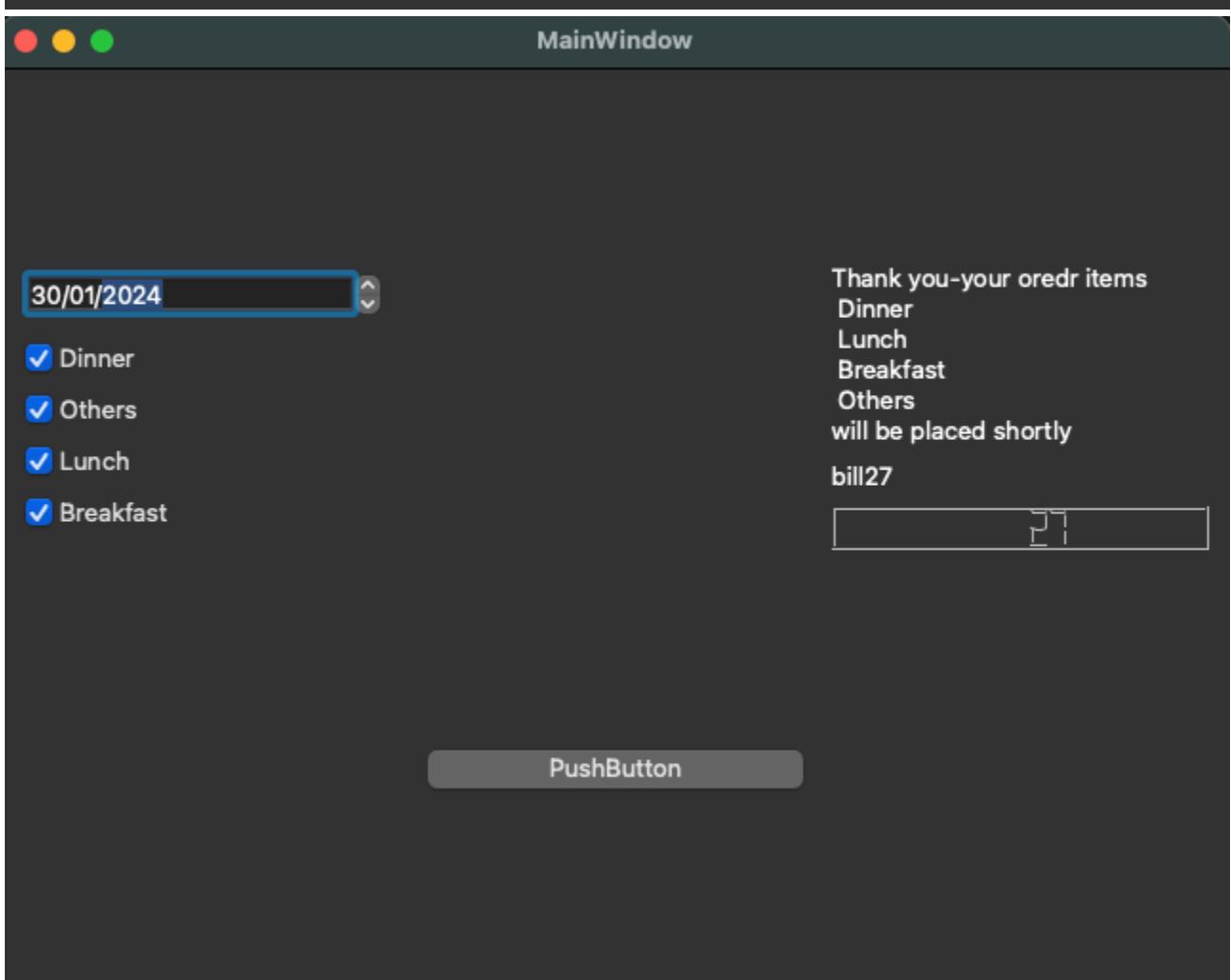
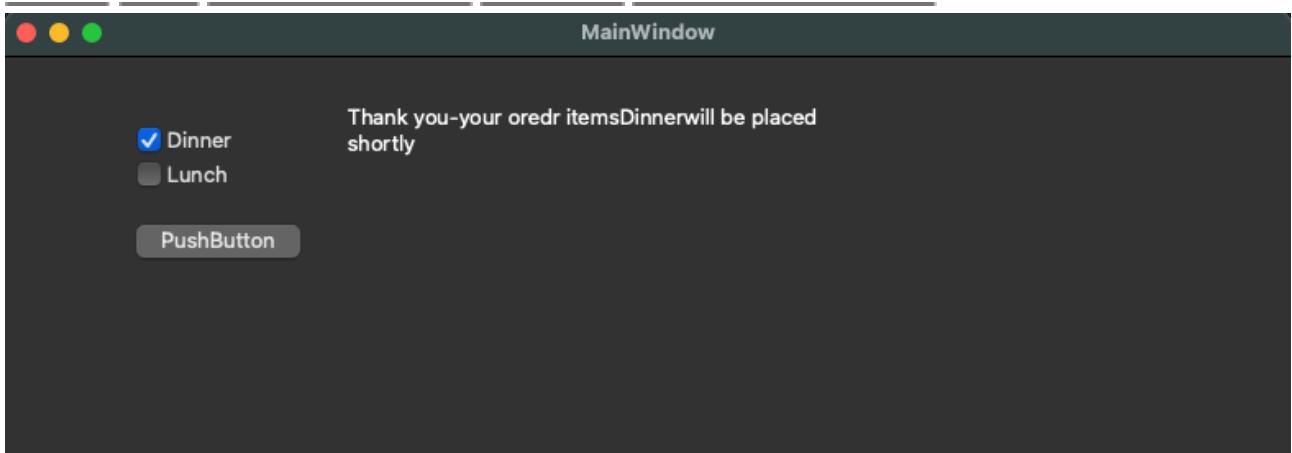
Contient un exemple basique sur les checkbox dans QtCreator.

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

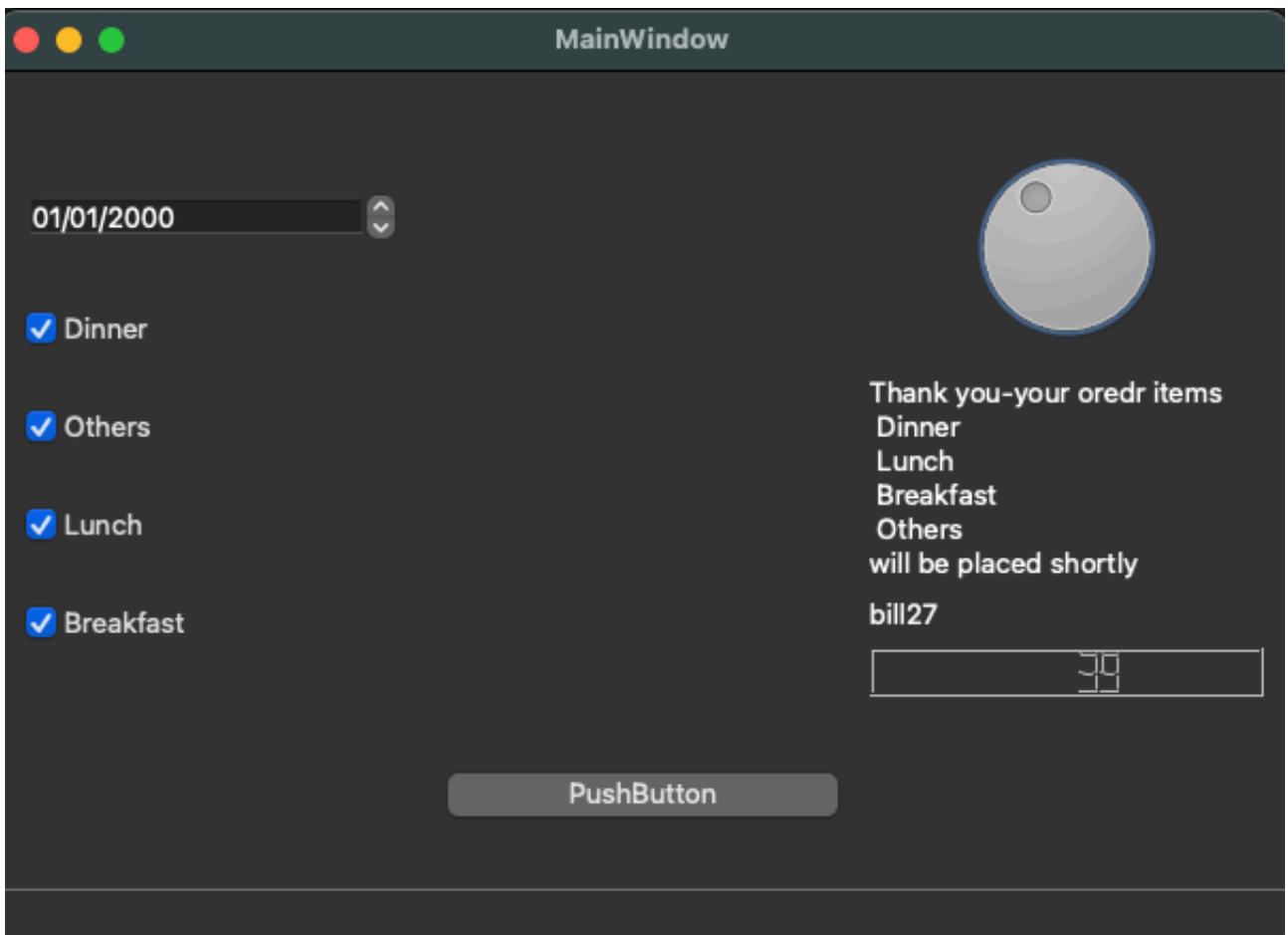


Ajout

On somme de la commande sur le LCD.

d'un curseur circulaire dont la valeur est affichée sur le LCD lorsque celui change.

affiche également toujours la

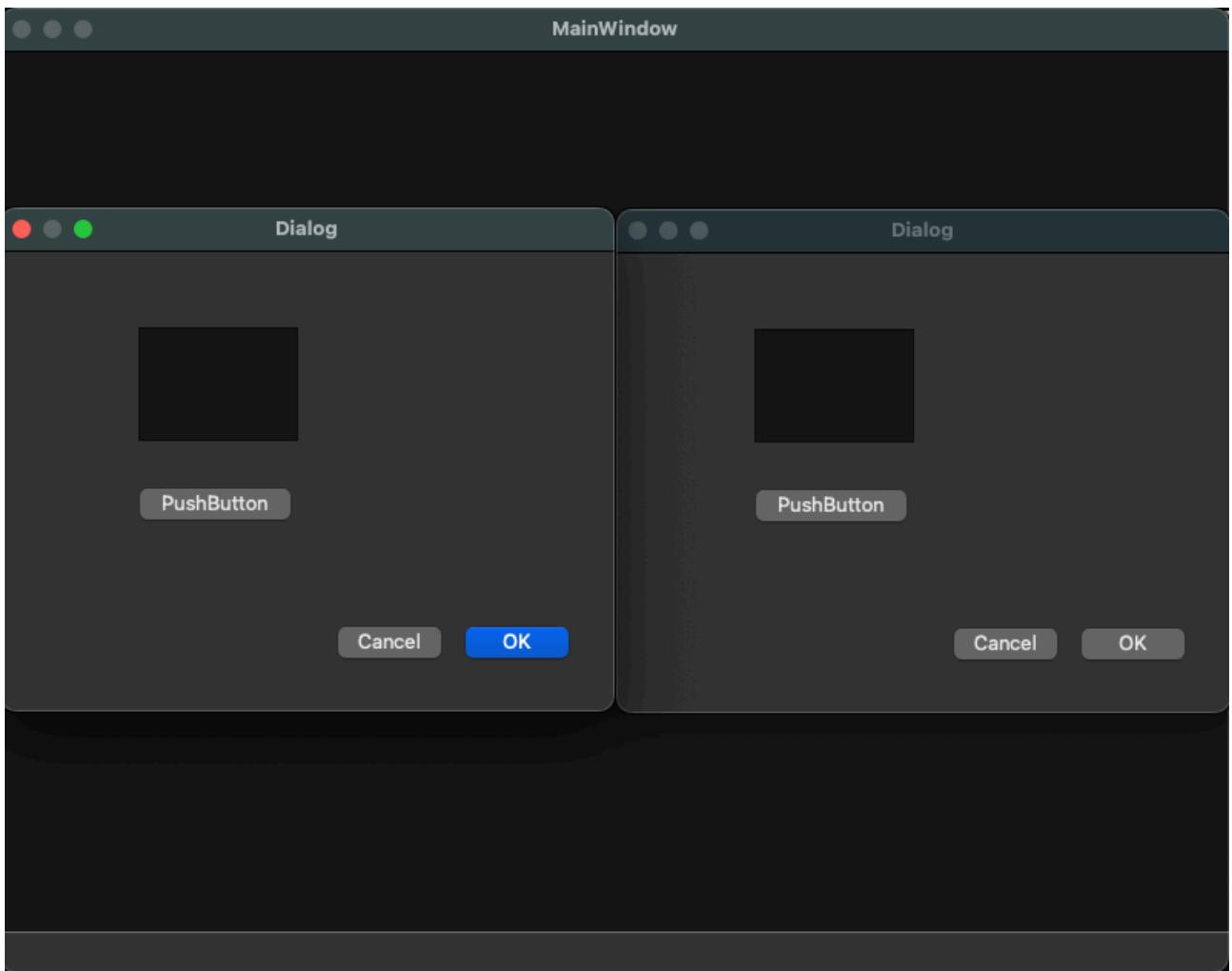


Le dossier displayingWindows est extrait de youtube VoidRealms Creation d'un projet nommé DisplayingWindows sous Qt

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

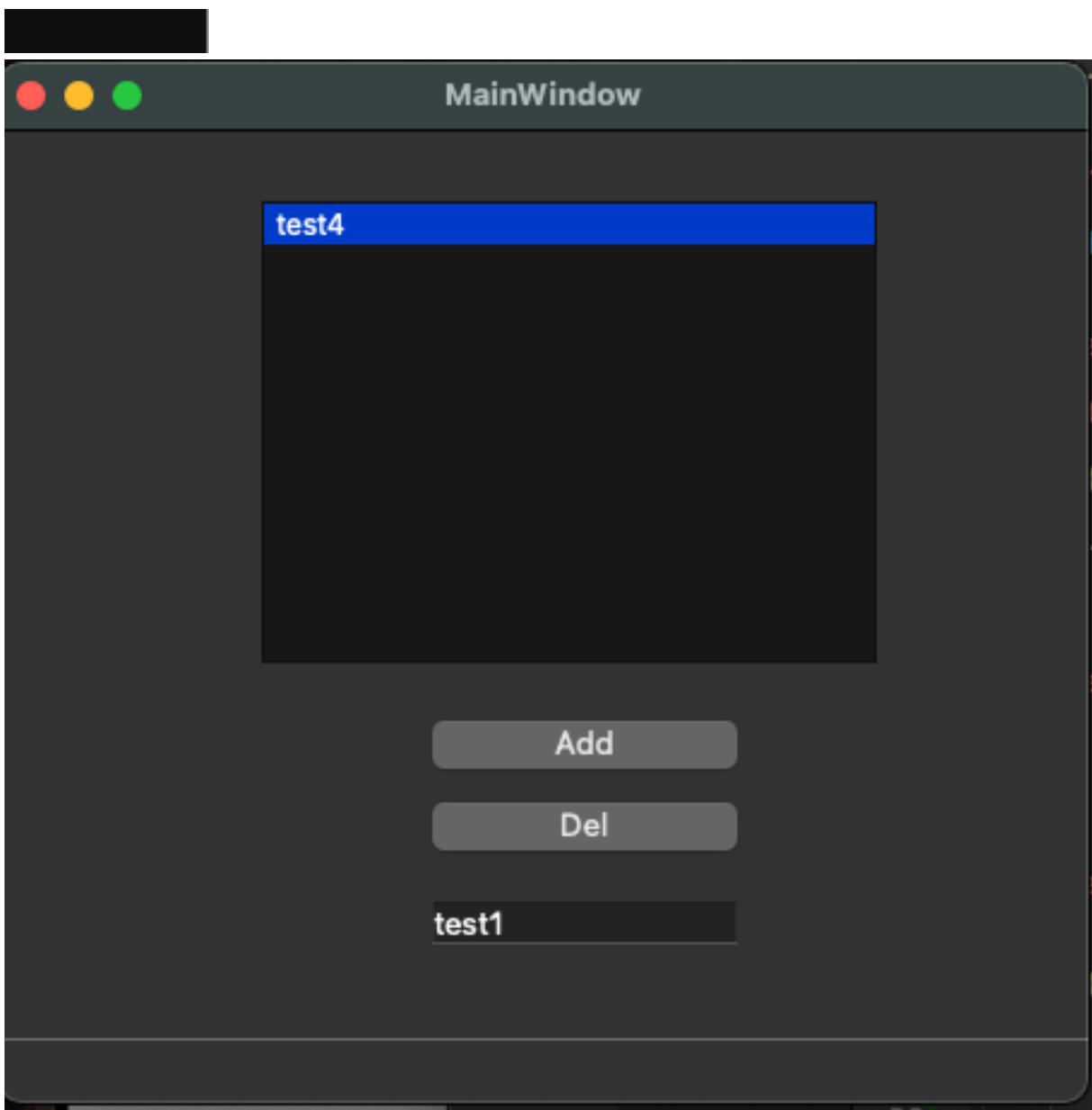
// modification de mainwindow.h pour intégrer Dialog *mDialog;

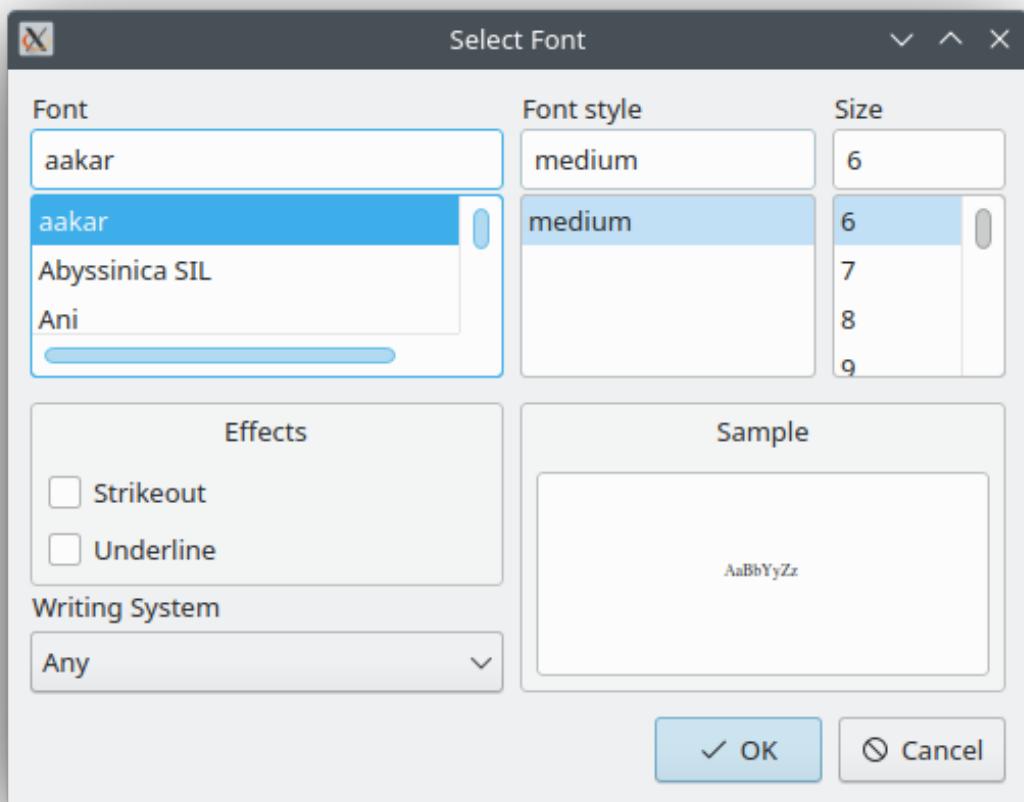


Le dossier `ReadingWritingTextFile` est extrait de YouTube
On écrit et lit dans un fichier « `test.txt` » avec `OStream`, si on change le nom du fichier alors message « `file not found` »

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`

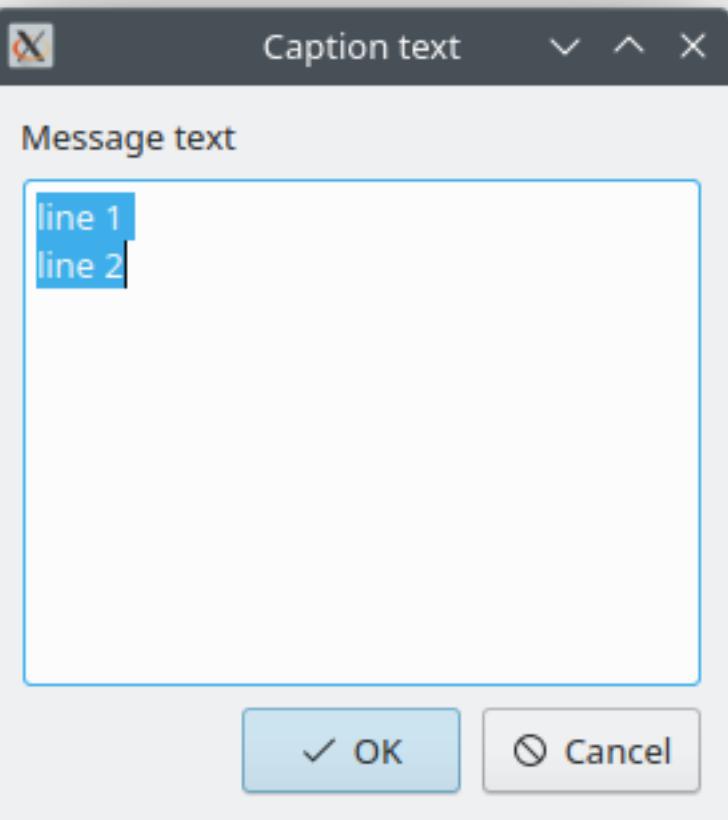
Le dossier `QtFontdialog` est extrait du site https://github.com/gammasoft71/Examples_Qt/tree/master
Montre une fenêtre où l'utilisateur change au choix la police Output

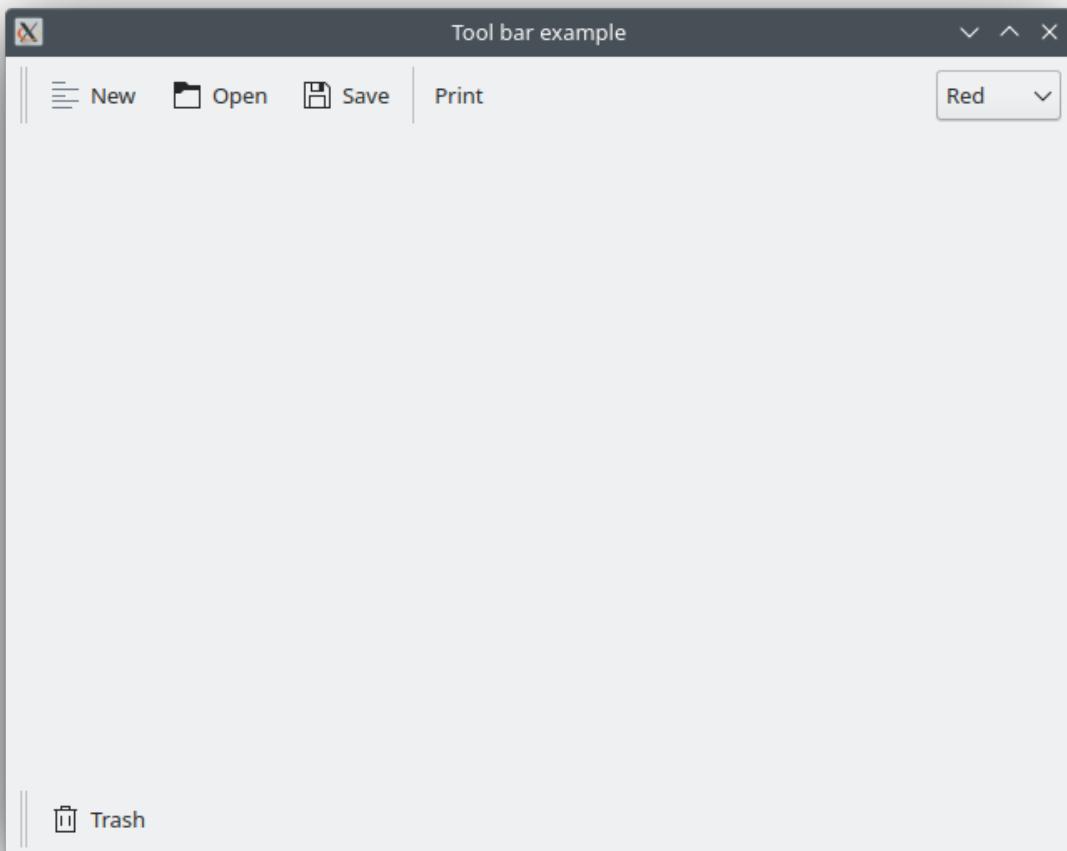




Le dossier Qt-InputDialogMultiline est extrait du même site

Le dossier Qt-ToolBar est extrait du même site



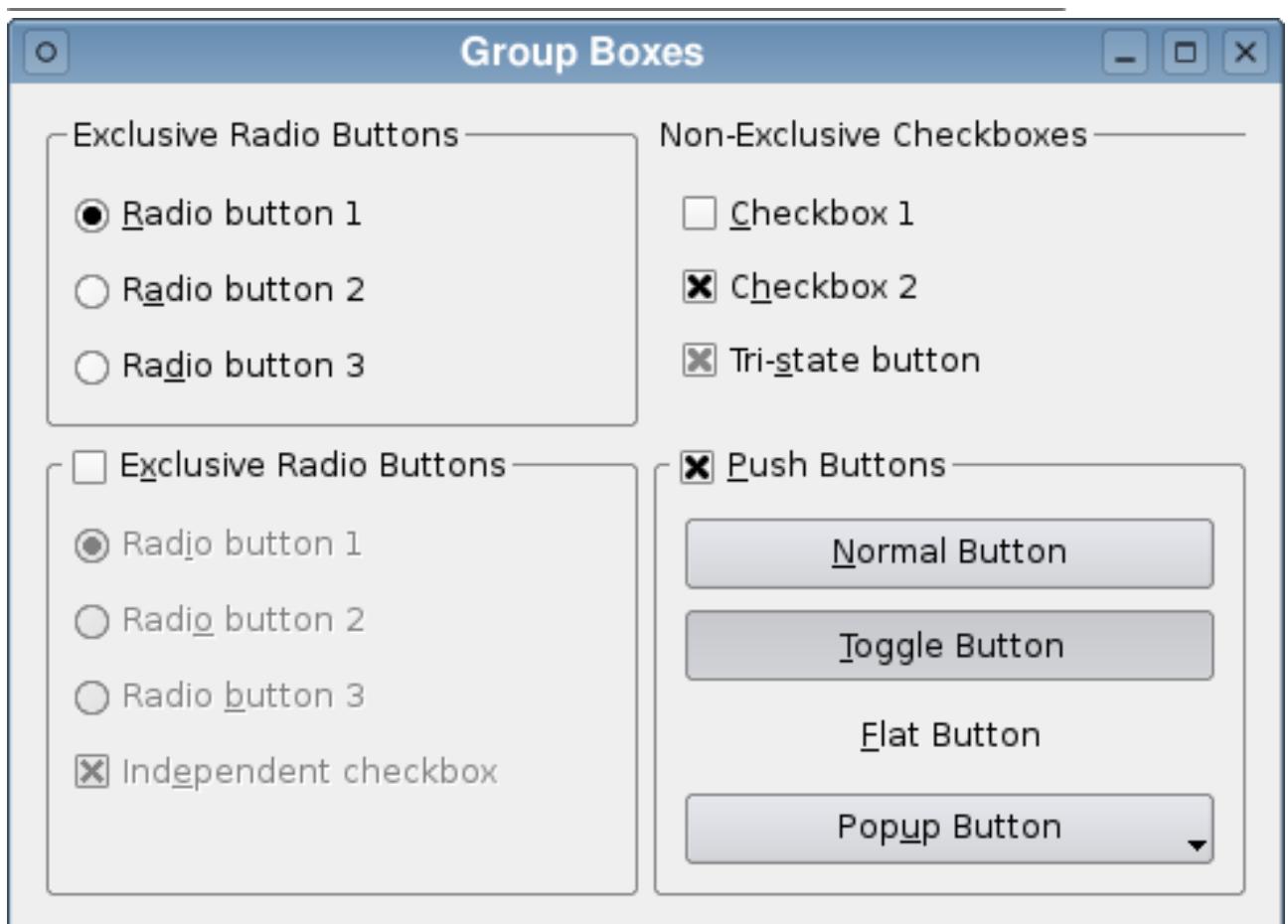
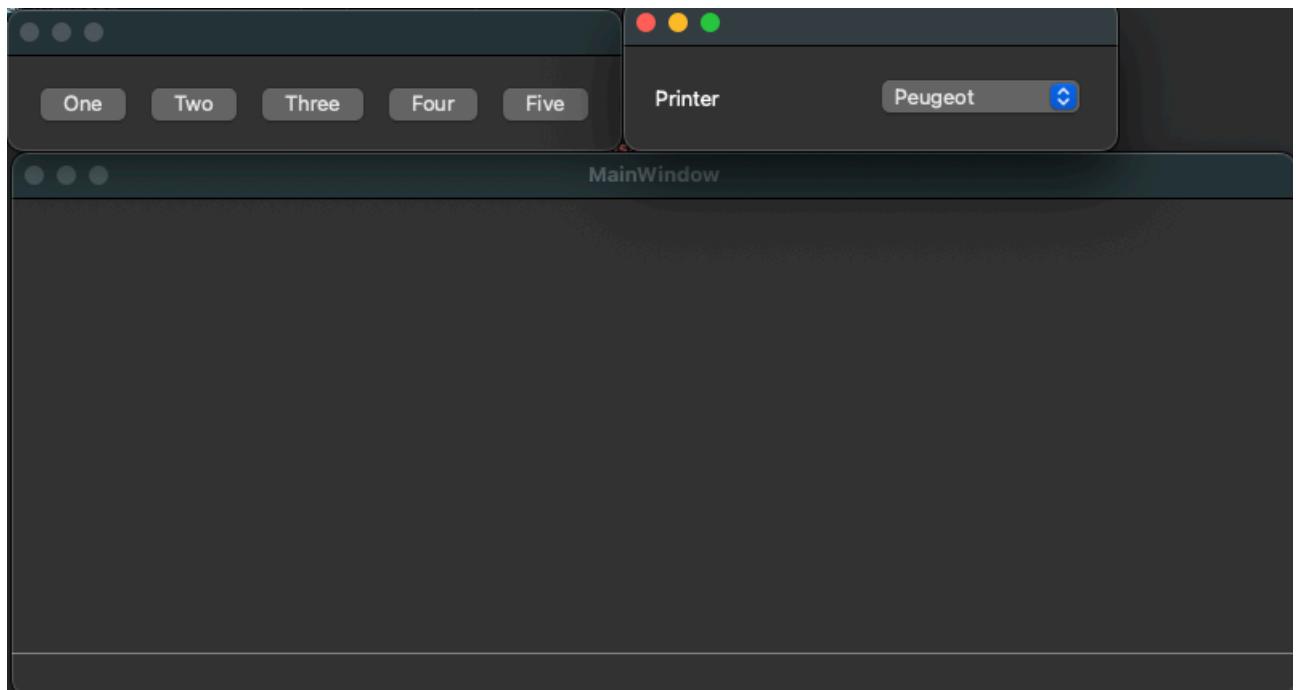


Les exemples suivants sont extrait du document Qt-Widgets-Layouts.dans repertoire documentation.

-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)

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

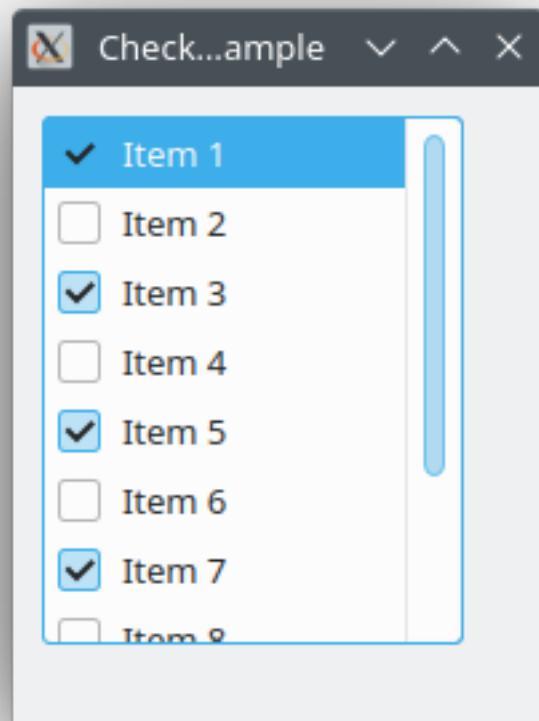
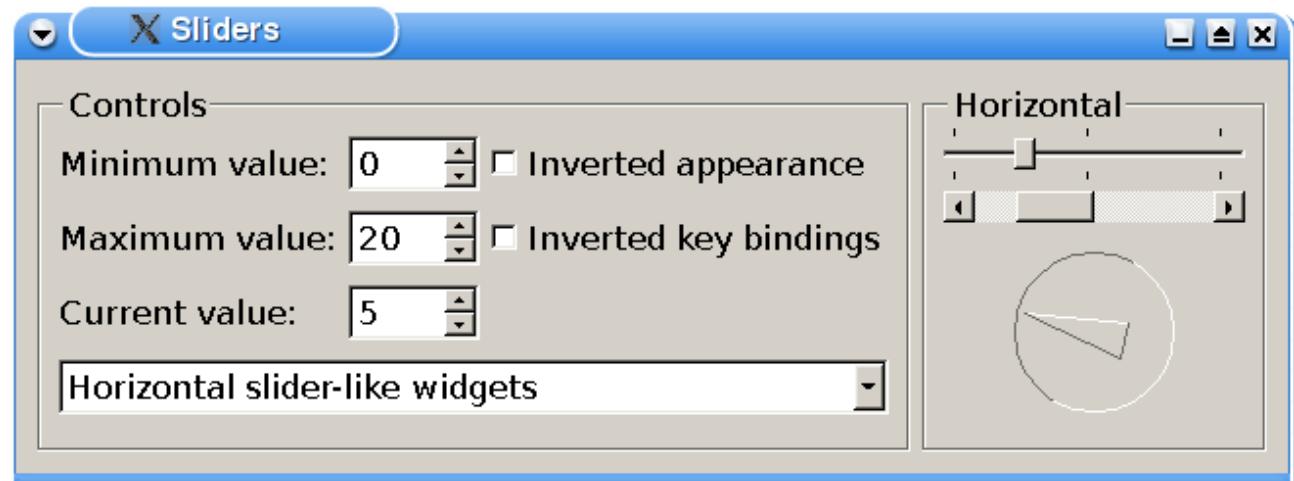


The Group Boxes example consists of a single `Window` class that is used to show four group boxes: an exclusive radio button group, a non-exclusive checkbox group, an exclusive radio button group with an enabling checkbox, and a group box with normal push buttons.

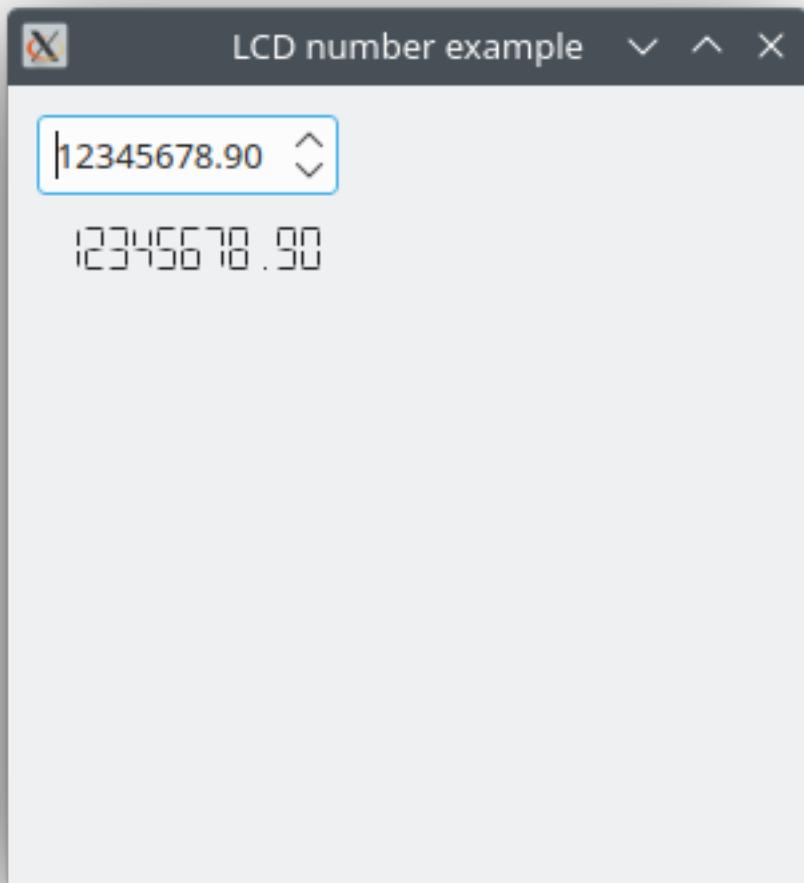
Exemple de Qt-sliderGroup

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



QtLCDnumber



MainWindow

	id	nom	prenom	telephone	email
1	8	benoist	jean		benoist.jean...
2	9	karim	nasr		kar.nasr@gm...
3	10	claude	nasr		cl.nasr@gma...
4	11	claude	nasr		cl.nasr@gma...
5	12	claude	nasr		cl.nasr@gma...
6	13	loic	benoit		
7	14	merlin	dominik		merlin.domin...
8	15	merlin	boris		merlin.boris...
9	16	merlin	boris		merlin.boris...
10	17	merlin	boris		merlin.boris...
11	18	laure	nasr		l.nasr@gmail...

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

MainWindow

	id	nom	prenom	telephone	email
1	136	MERLIN	Dominik	0888774433	dominik.merl...
2	137	MERLIN	Dominik	0888774433	dominik.merl...
3	138	MERLIN	Dominik	0888774433	dominik.merl...

Gestion base donnee contacts sous SQLite

categories

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

	id	TITRE	ANNEE	SUJET
1	1	programmati...	2018	C sharp
2	2	WPF...	2017	Raspberry Pi
3	3	Raspberry Pi 4	2019	Raspberry Pi
4	4	Arduino	2019	C sharp

ajouter un sujet

supprimer un sujet

Usage de la classe QsqlRelationaltablemodel (chapitre 8) /Users/nasr/PycharmProjects/PySide/main.py-Qt-QsqlRelationalModel

```
[REDACTED]  
import os  
[REDACTED]  
import PyQt6  
[REDACTED]  
from PyQt6 import QtSql  
[REDACTED]  
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 QWidget
from PySide6.QtWidgets import
QApplication, QLabel, QWidget,
QBoxLayout, QVBoxLayout, QTableView, QDialog,
QPushButton
from PySide6.QtGui import QGuiApplication
from PySide6.QtCore import QCoreApplication, Qt
from PySide6.QtSql import
 QSqlDatabase, QSqlRelationalTableModel, QSqlRelati
tion, QSqlRelati
onalDelegate
from PySide6.QtSql import QSqlTableModel
[REDACTED]
class FenetreSimple(QWidget):
[REDACTED]
def __init__(self):
[REDACTED]
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(
        "VALUES('Mocq','François')"
    )
    query.exec(
        "INSERT INTO PERSONNE('NOM','PRENOM') "
        "VALUES('Lacaze','Sarah')"
    )
    )

    " )  " )
    " ) )
PERSONNE
```

```
print(query.exec("
SELECT COUNT(*) FROM
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')
VALUES('Raspberry pi')
query.exec(
INSERT INTO SUJET('SUJET')
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))
self.db.commit()
" )

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

```
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 = """
id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
create table SUJET(
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")
    else:
        print("echec de la creation de la table SUJET")

if query.exec(creationTablePersonne):
    print("creation table PERSONNE ok")
    self.db.close()

def joinDB():
    print("creation table Personne ok")
    self.db.close()
def execute (self):
    self.resize(250, 300)
    self.move(50, 500)
    donnees")
    self.setWindowTitle("chapitre 8 - insertion de
    self.show()
```

```
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):
script.
```

```
# Use a breakpoint in the code line below to
debug your
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)
[REDACTED]

base =
QtSql.QSqlDatabase.addDatabase('QSQLITE')
[REDACTED]

base.setDatabaseName('Baselivres.db')
#definition du modele
#modele = QtCore.QAbstractItemModel
modele = QSqlRelationalTableModel()
modele.setTable('LIVRE')
[REDACTED]

modele.setEditStrategy(QSqlRelationalTableModel.EditStrategy.
[REDACTED]

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  
[REDACTED]  
NOM" )  
  
[REDACTED]  
modele.select()  
[REDACTED]  
#creation de la vue et association au modele  
[REDACTED]  
vue = QTableView()  
[REDACTED]  
vue.setModel(modele)  
[REDACTED]  
vue.setItemDelegate(QSqlRelationalDelegate(vue  
))  
[REDACTED]  
#dialogue = QDialog()  
[REDACTED]  
#disposition = QVBoxLayout()  
[REDACTED]  
#disposition.addWidget(vue)  
#fenetre = QtWidgets.QWidget()  
#fenetre.QTableView().setModel(QtSql.QSqlTable  
Model())  
#vue.setItemDelegate(QSqlRelationalDelegate(vu  
e))  
#creation de la boite de dialogue  
dialogue = QDialog()  
disposition = QVBoxLayout()  
disposition.addWidget(vue)
```

```
#bouton d ajout
[REDACTED]
bouton_ajout = QPushButton("ajouter un livre")
[REDACTED]
bouton_ajout.clicked.connect(lambda :modele.insertRows(modele
[REDACTED]
.rowCount(),1))
[REDACTED]
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()
[REDACTED]
#
# ...
[REDACTED]
sys.exit(app.exec())
[REDACTED]
```

```
import sys  
[REDACTED]  
import os  
[REDACTED]  
import PyQt6  
  
from PyQt6 import QSql  
[REDACTED]  
from PyQt6 import QtCore  
[REDACTED]  
  
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 QWidget  
from PySide6.QtWidgets import  
QApplication, QLabel, QWidget,  
QBoxLayout, QVBoxLayout, QTableView, QDialog,  
QPushButton  
from PySide6.QtGui import QGuiApplication  
from PySide6.QtCore import QCoreApplication, Qt  
from PySide6.QtSql import  
QSqlDatabase, QSqlRelationalTableModel, QSqlRela  
tion, QSqlRelati  
onalDelegate  
from PySide6.QtSql import QSqlTableModel  
[REDACTED]  
  
class FenetreSimple(QWidget):  
[REDACTED]  
  
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 =
QtSql.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(
```

```
VALUES( 'Mocq' , 'Fran ois' )
INSERT INTO PERSONNE( 'NOM' , 'PRENOM' )
query.exec( "
INSERT INTO PERSONNE( 'NOM' , 'PRENOM' )
VALUES( 'Lacaze' , 'Sarah' )
")
") ")
"))
```

PERSONNE

```
print(query.exec(
SELECT COUNT(*) FROM
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' )
VALUES( 'Raspberry pi' )
query.exec(
INSERT INTO SUJET('SUJET')
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("while query.next():  
        query.exec("SELECT COUNT(*) FROM LIVRE  
        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)  
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 = """
id INTEGER PRIMARY KEY AUTOINCREMENT,
create table SUJET(
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
[REDACTED]
self.db.open()
[REDACTED]
query = QSqlQuery()
[REDACTED]
#self.db.close()
[REDACTED]
if query.exec(creationTableLivre):
[REDACTED]
```

```
print("creation table LIVRE ok")
[REDACTED]
if query.exec(creationTableSujet):
[REDACTED]
print("creation table SUJET ok")
[REDACTED]
if query.exec(creationTablePersonne):
[REDACTED]
print("creation table Personne ok")
self.db.close()
def execute (self):
    self.resize(250, 300)
    self.move(50, 500)
    donnees")
self.setWindowTitle("chapitre 8 - insertion de
self.show()

[REDACTED]

def joinDB():
[REDACTED]
app = QtCore.QCoreApplication(sys.argv)
db = QSql.QSqlDatabase.addDatabase('QSQLITE')
[REDACTED]
db.setDatabaseName('Baselivres.db')
[REDACTED]
print("connexion base ok")
[REDACTED]
if not db.open():
[REDACTED]
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
[REDACTED]
sujet = 'Scratch'
[REDACTED]
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():
else:

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

[REDACTED]
while query.next():
[REDACTED]
```

```
print(query.value(0),query.value(1))
def print_hi(name):
script.

# Use a breakpoint in the code line below to
debug your
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)
[REDACTED]
base =
QtSql.QSqlDatabase.addDatabase('QSQLITE')
[REDACTED]

base.setDatabaseName('Baselivres.db')
#definition du modele
#modele = QtCore.QAbstractItemModel
modele = QSqlRelationalTableModel()
modele.setTable('LIVRE')
[REDACTED]

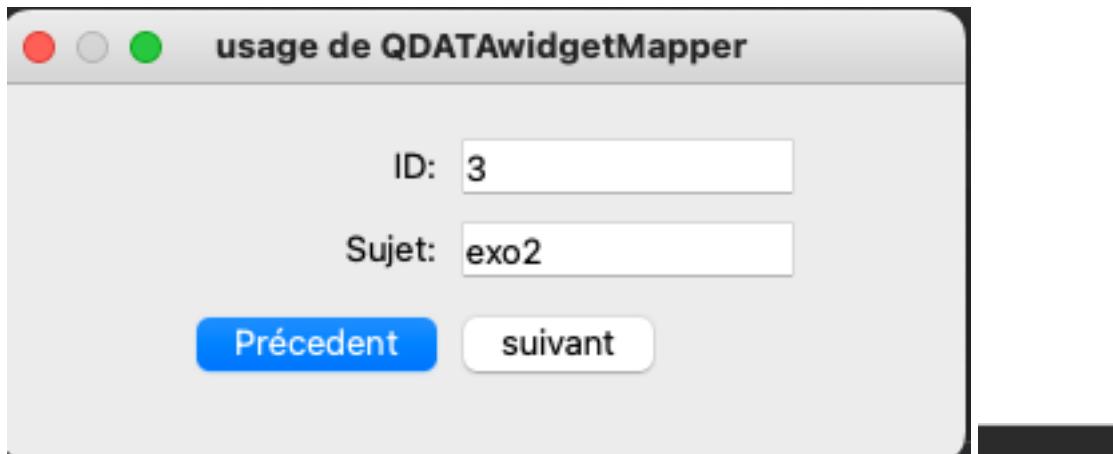
modele.setEditStrategy(QSqlRelationalTableModel.EditStrategy.
```

```
[REDACTED]  
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")  
[REDACTED]  
#modele.setHeaderData(4,QtCore.Qt.Orientation.  
Horizontal, "PRE  
[REDACTED]  
NOM")  
[REDACTED]  
modele.select()  
#creation de la vue et association au modele  
vue = QTableView()  
vue.setModel(modele)  
vue.setItemDelegate(QSqlRelationalDelegate(vue))  
[REDACTED]  
#dialogue = QDialog()  
[REDACTED]  
#disposition = QVBoxLayout()  
[REDACTED]  
#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)
[REDACTED]
#bouton d ajout
[REDACTED]
bouton_ajout = QPushButton("ajouter un livre")
[REDACTED]
bouton_ajout.clicked.connect(lambda :modele.insertRows(modele
[REDACTED]
.rowCount(),1))
[REDACTED]
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 *
[REDACTED]
from PyQt6 import QtWidgets
[REDACTED]
from PyQt6.QtWidgets import *
[REDACTED]
#from PyQt6 import QWidget
[REDACTED]
from PySide6.QtWidgets import QApplication, QLabel, QWidget,
[REDACTED]
QBoxLayout, QVBoxLayout, QTableView, QDialog, QLineEdit,
[REDACTED]
QPushButton, QFormLayout, QDataWidgetMapper
[REDACTED]
from PySide6.QtGui import QGuiApplication
[REDACTED]
from PySide6.QtCore import QCoreApplication, Qt
[REDACTED]
from PySide6.QtSql import
[REDACTED]
QSqlDatabase, QSqlRelationalTableModel, QSqlRelation, QSqlRelationalD
elegate
[REDACTED]
from PySide6.QtSql import QSqlTableModel
[REDACTED]
class FenetreSimple(QWidget):
[REDACTED]
def __init__(self):
[REDACTED]
super().__init__()
self.disposition = QVBoxLayout()
self.clickbouton = QPushButton("Click", clicked=self.creationDB)
self.disposition.addWidget(self.clickbouton)
self.setLayout(self.disposition)
self.execute()
[REDACTED]
#QtGui.QWindow.__init__(self, parent)
[REDACTED]
self.resize(30, 30)
[REDACTED]
#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(
while query.next():
    print(query.value(0))
    SELECT COUNT(*) FROM PERSONNE
    query.exec(
    query.exec(
    query.exec(
        INSERT INTO PERSONNE('NOM', 'PRENOM') VALUES('Prieur', 'Benoit')
        INSERT INTO PERSONNE('NOM', 'PRENOM') VALUES('Mocq', 'Fran ois')
        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(
        query.exec(
        query.exec(
            INSERT INTO SUJET('SUJET') VALUES('Csharp')
```

```
INSERT INTO SUJET( 'SUJET' ) VALUES('Raspberry pi')
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)
[REDACTED] [REDACTED]
print(query.exec(
query.exec(
"""
"""
"""
"""
```

```
SELECT COUNT(*) FROM LIVRE

while query.next():

print(query.value(0))

query = QSqlQuery()

UPDATE SUJET SET SUJET = 'MICROSOFT C#' WHERE id =2

query.exec(
SELECT COUNT(*) FROM SUJET
" )

" ))
```

```
while query.next():

print(query.value(0))

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

" )

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 = """
id INTEGER PRIMARY KEY AUTOINCREMENT,
create table SUJET(
```

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

def prec():

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

```
self.show( )  
  
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 = QSqlQuery()
[REDACTED]
sujet = 'Scratch'
[REDACTED]
annee = 2019
requeste="""
SELECT PERSONNE.PRENOM, PERSONNE.NOM
FROM PERSONNE, LIVRE, SUJET
WHERE PERSONNE.id = LIVRE.id
AND SUJET.id = LIVRE.id
""")
```

query.prepare(requeste)

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

if query.exec():

```
while query.next():
[REDACTED]
print(query.value(0), query.value(1))
[REDACTED]
else:
```

```
[REDACTED]
print("erreur dans l execution de la
requeste:", 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)

# creation de la boite de dialogue

dialogue = QDialog()

disposition = QFormLayout()
[REDACTED]

#champ d'édition
id = QLineEdit()
[REDACTED]
id.setReadOnly(True)
[REDACTED]

sujet = QLineEdit()
[REDACTED]

disposition.addRow("ID:", id)
[REDACTED]

disposition.addRow("Sujet:", sujet)
[REDACTED]

precedentBouton = QPushButton("Précedent")
```

```
[REDACTED]
suivantBouton = QPushButton("suivant")
[REDACTED]
disposition.addRow(precedentBouton, suivantBouton)
[REDACTED]
dialogue.setLayout(disposition)
[REDACTED]
dialogue.setWindowTitle("usage de QDataWidgetMapper")
[REDACTED]
base = QSql.QSqlDatabase.addDatabase('QSQLITE')
[REDACTED]
base.setDatabaseName('Baselivres.db')
#definition du modele
#modele = QtCore.QAbstractItemModel
modele = QSqlTableModel()
modele.setTable('SUJET')
modele.setEditStrategy(QSqlTableModel.EditStrategy.OnFieldChange)
[REDACTED]
#modification immediate
[REDACTED]
modele.select()
[REDACTED]
mapping = QDataWidgetMapper()
[REDACTED]
mapping.setModel(modele)
[REDACTED]
mapping.addMapping(id, 0)
[REDACTED]
mapping.addMapping(sujet, 1)
[REDACTED]
mapping.toFirst()
[REDACTED]
precedentBouton.clicked.connect(prec)
[REDACTED]
suivantBouton.clicked.connect(suiv)
    modele.select()
    dialogue.show()
```

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

identique au cas précédent avec QFONT et nouvelle police



...



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

chapitre 3 - formulaire

Nom:

Prénom:

Loisir préféré: Pratique sportive

possède un vélo?

chapitre 3 - formulaire

Nom:

Prénom:

Loisir Préfere: Pratique sportive

Possede Un Vélo?

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

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

Qt-PythonStateMachine utilise un timer scheduler avec QTimer pour afficher et fermer des fenêtres.

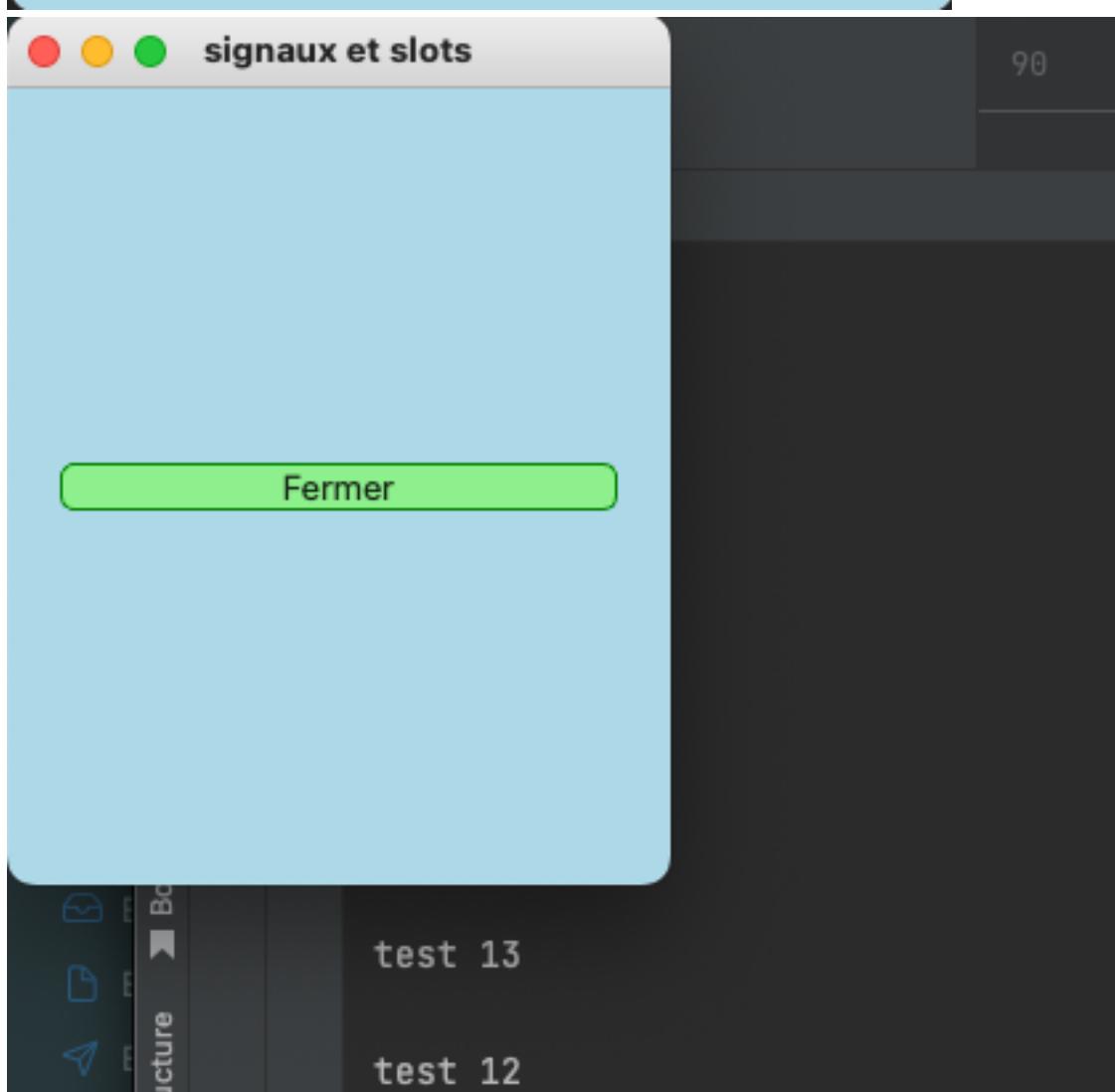
gestion de style avec qss

Nom:

Prénom:

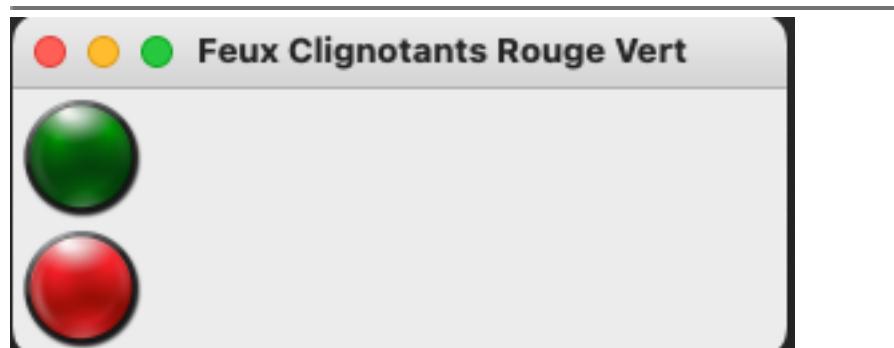
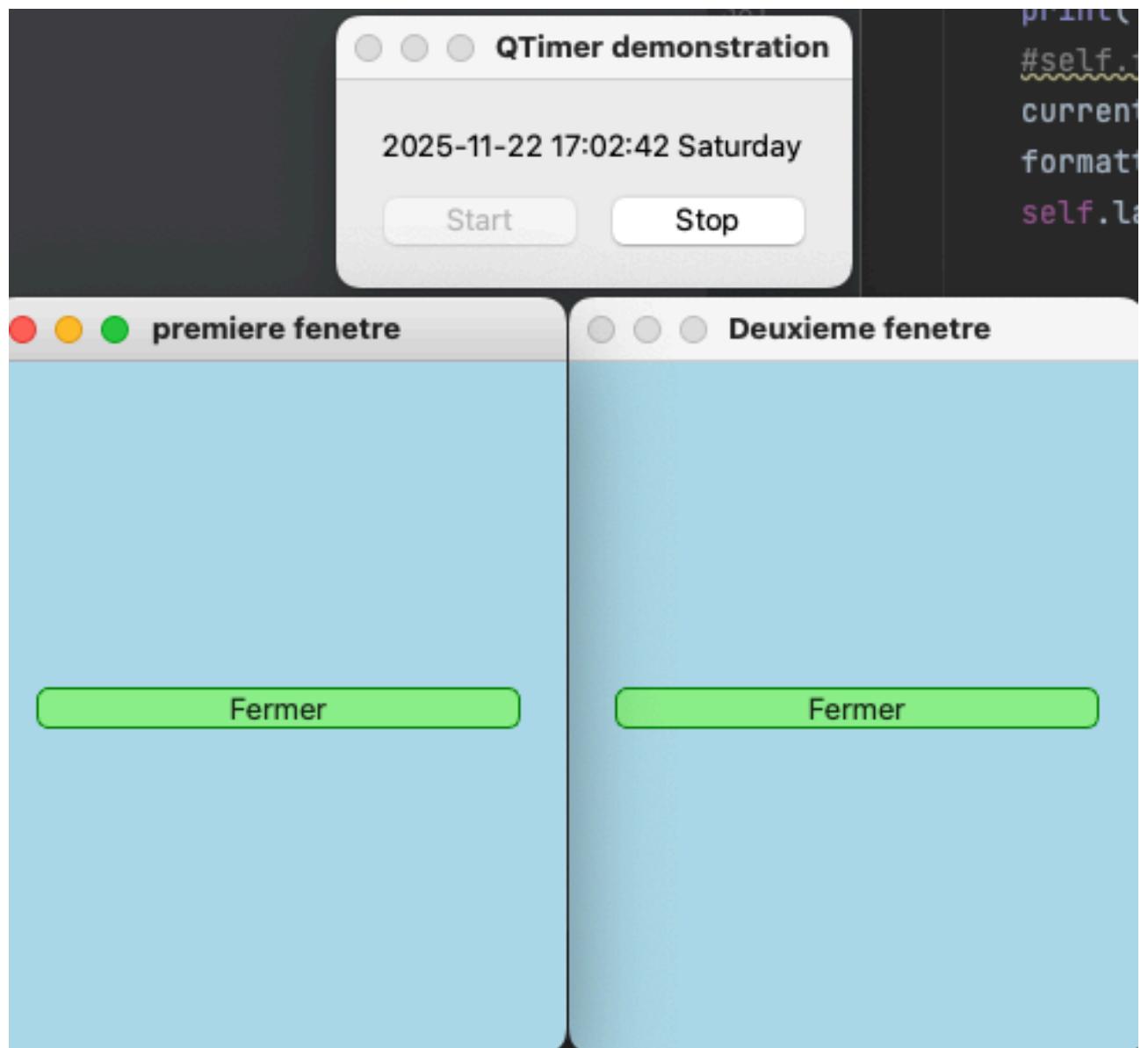
loisir prefere: Pratique sportive

possede un vélo?



Qt-timer (sous pycharm) est un scheduler de fenetre, la premiere fenetre s'affiche en premier et sur click du bouton fermer , la deuxieme fenetre est lancée. A noter sur le timeout du timer , il faut faire un show de la deuxieme fenetre sinon elle disparait (a investiguer davantage).

LedIndicatorWidget simule les feux rouge/vert pendant 1 minutes et affiche sur terminal l etat, projet complet extrait de https://taurus-scada.org/_modules/taurus/qt/qtgui/display/qled.html Se lance sous pycharm.



30/11 New token

New classic token on GitHub

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