1 PyQt教學(0)-認識常用的UI元件

範例一 （完整代碼如下）

```
import sys
from PyQt4.QtGui import *

#開始 GUI 程式
app = QApplication(sys.argv)

#主視窗元件
widget = QWidget()

#(視窗座標 x,視窗座標 y,視窗寬,視窗高)
widget.setGeometry(500,200,400,300)

#宣告一個繼承於 widget 的標籤
lab = QLabel('label',widget)

#宣告一個繼承於 widget 的文本框
txt = QTextEdit('text',widget)

#宣告一個繼承於 widget 的勾選框
chk = QCheckBox(widget)

#宣告一個繼承於 widget 的按鈕
btn = QPushButton('button',widget)

#宣告一個繼承於 widget 的下拉選單
com = QComboBox(widget)

#調整原件的絕對位置
lab.move(0,0)
txt.move(100,0)
chk.move(0,40)
btn.move(0,80)
com.move(0,120)

#顯示 widget 以及底下的元件
```

```
widget.show()

#維持 GUI 程式的運作
sys.exit(app.exec_())
```

範例二 （完整代碼如下）

```
import sys
from PyQt4.QtGui import *

app = QApplication(sys.argv)

#QInputDialog(第一參數 None 表示不繼承任何物件)
text, ok = QInputDialog.getText(
    None,
    'title',
    'hello world!',
    QLineEdit.Normal,
    'you can set default text here.')

sys.exit(app.exec_())
```

2 PyQt 教學(1)-認識常用的函數

#引入常用的函數

```
from PyQt4.QtGui  import *
from PyQt4.QtCore import *
```

#定時器使用範例(一秒後打印!)

```
QTimer.singleShot(1000, lambda: print('!'))
```

#配置主視窗

```
widget.showMaximized()   #最大螢幕，不隱藏上方工具列
widget.showFullScreen()  #最大螢幕，隱藏上方工具列
widget.setFixedSize(1,1)#固定視窗大小
```

#QTextEdit-文本框常用函數

```
text.setReadOnly(True)                      #唯讀
text.moveCursor(QTextCursor.Start)          #移動游標
text.moveCursor(QTextCursor.Down)           #游標往下一行
text.setLineWrapMode(QTextEdit.NoWrap)      #水平滾軸
text.toPlainText()                          #文本內容
text.setTextColor(QColor('#003DF5'))        #變換文字顏色
text.setAcceptRichText(False)               #禁用 HTML
text.verticalScrollBar().setEnabled(0)      #禁用滾軸
text.insertPlainText('hello')               #在游標處插入字串
text.setVerticalScrollBarPolicy(
    Qt.ScrollBarAlwaysOff)                  #隱藏垂直滾軸

#兩文本框垂直滾軸在滑鼠拉動時同步
textA.verticalScrollBar().valueChanged.connect(
    textB.verticalScrollBar().setValue)

#將兩文本框的垂直滾軸同步
textA.verticalScrollBar().setValue(
    textB.verticalScrollBar().value())
```

#QTextCursor-游標常用函數

```
cursor = text.textCursor()          #取得文本框的游標
curS = cursor.selectionStart()      #選取中，開頭字元的位置
curE = cursor.selectionEnd()        #選取中，結束字元的位置
cursor.setPosition(curE)            #設置游標的字元位置

#將游標移動到該行最後面
cursor.movePosition(QTextCursor.EndOfLine)
cursor.columnNumber()               #取得游標所在行的位置(行)
cursor.blockNumber()                #取得游標所在行的位置(列)

#選取該行的文本(表面看不到)
cursor.select(QTextCursor.LineUnderCursor)
```

```
cursor.selectedText()           #取得選取的文本內容
cursor.hasSelection()           #判斷是否有內容被選取
cursor.deletePreviousChar()     #刪除游標所在的前一個字元
cursor.insertText()             #在游標處插入字串
```

#QFileDialog-開檔，存檔

```
QFileDialog.getOpenFileName(
    widget, 'Open file', 'default name')

QFileDialog.getSaveFileName(
    widget, "Save file", "", ".txt")

QFileDialog.getOpenFileName(
    None, "", "","Text files (*.txt);;XML files (*.xml)" )
```

#QMessageBox-詢問

```
reply = QMessageBox.question(
    widget,'Message',"Quit?",
    QMessageBox.Yes,
    QMessageBox.No)

print('yes' if reply == QMessageBox.Yes else 'no')
```

#obj 可能為任何種類的 UI 元件

```
obj.setEnabled(False)                                   #禁用元件
obj.show()                                              #顯示元件
obj.hide()                                              #隱藏元件
obj.setFocus()                                          #設置焦點
obj.setCursor(QCursor(Qt.PointingHandCursor))           #更換游標
obj.setFont(QFont("consolas", 20, 80, False))           #字型配置
obj.setPixmap(QPixmap('mute.png'))                      #載入圖片
obj.setPixmap(QPixmap(''))                              #移除圖片
obj.setText('string')                                   #設置字串
obj.width()                                             #元件寬
obj.height()                                            #元件高
```

#取得螢幕的寬和高

```
screen = app.desktop().size()
print(screen.height(),screen.width())
```

#用 CSS 語法改變樣式(若調用第二次則會覆蓋先前那次!)

```
obj.setStyleSheet('color: yello')
obj.setStyleSheet('color: red; border: 2px solid blue;')
```

#QCheckBox(調整框框大小)

```
widget.setStyleSheet('''
    QCheckBox::indicator{
    width: 40px;
    height: 40px;
}''')
```

#操控剪貼板

```
clipboard = QApplication.clipboard()
clipboard.setText('apple')
```

#此時可在編輯器中用 ctrl+v，會貼出'apple'的字樣

3 PyQt 教學(2)-調整 UI 元件的排列

使用相對排列法可以使控件在伸縮視窗時自動伸縮與排列，而最常用的相對位置排列法是 QGridLayout !(完整代碼如下)

```
import sys
from PyQt4.QtGui import *

app = QApplication(sys.argv)
widget = QWidget()
widget.setGeometry(300, 100, 500, 450)
```

#使用 CSS 語法調整樣式

```
widget.setStyleSheet("""
    QLineEdit{
        border : 5px solid brown;
    }
    QTextEdit{
        border : 5px double;
    }
```

```
    """)

#標籤內可以夾雜 css 語法
title = QLabel('<font color=blue>Title</font>')
author = QLabel('<font color=blue>Author</font>')
review = QLabel('<font color=red>Review</font>')
titleEdit = QLineEdit()
authorEdit = QLineEdit()
reviewEdit = QTextEdit()

grid = QGridLayout()

#設置排列後每列的間隔
grid.setSpacing(10)

#指定每個元件的列和行(後兩個參數為列、行)
grid.addWidget(title, 1, 0)
grid.addWidget(titleEdit, 1, 1)
grid.addWidget(author, 2, 0)
grid.addWidget(authorEdit, 2, 1)
grid.addWidget(review, 3, 0)

#四個參數分別為 row, col, rowspan, colspan
grid.addWidget(reviewEdit, 3, 1, 3, 3)

#讓主視窗套用剛剛設定的排列方法
widget.setLayout(grid)

widget.show()
app.exec_()
```

4 PyQt 教學(3)-以物件來美化程式碼

我們可將物件的樣式、功能的調整都包裝在 class 裡面!(完整代碼如下)

```
import sys, os
from PyQt4.QtGui import *
from PyQt4.QtCore import *

class QWindow(QWidget):
    def __init__(self):
```

```
        super(QWindow, self).__init__()
        self.setWindowTitle('title')
        self.setGeometry(400,100,500,400)
        self.show()

class QLab(QLabel):
    def __init__(self, parent, text=''):
        QLabel.__init__(self, parent)
        self.setText(text)
        self.setFont(QFont('Arial', 16, 30, False))
        self.setStyleSheet(
           'background-color:black;' +
           'color:white;' +
           'border:brown;'+
           'text-align: right;')
        self.show()

class QBtn(QPushButton):
    def __init__(
        self, parent, text='click', enable=True):
        QPushButton.__init__(self, parent)
        self.setText(text)
        self.setFont(QFont('Arial', 16, 30, False))
        self.setEnabled(enable)
        self.setCursor(
            QCursor(Qt.PointingHandCursor))
        self.show()

if __name__ == '__main__':
    app = QApplication(sys.argv)

    widget = QWindow()
    labelr = QLab(widget,'Hello world!')
    butn_1 = QBtn(widget,'enable')
    butn_2 = QBtn(widget,'enable')
    butn_3 = QBtn(widget,'disable',False)

    grid = QGridLayout()
    grid.setSpacing(1)
    grid.addWidget(butn_1,0,0)
    grid.addWidget(butn_2,1,0)
    grid.addWidget(butn_3,2,0)
    grid.addWidget(labelr,0,1,3,3)
    widget.setLayout(grid)
```

```
    sys.exit(app.exec_())
```

5 PyQt 教學(4)-抓取滑鼠以及鍵盤事件

完整範例代碼如下!

```
import sys, os
from PyQt4.QtGui  import *
from PyQt4.QtCore import *

class QWindow(QWidget):
    def __init__(self):
        super(QWindow, self).__init__()
        self.setWindowTitle("Hello World")
        self.setWindowIcon(QIcon('image.ico'))
        self.setGeometry(480, 200, 400, 300)
        self.label = QLabel(
            '點擊滑鼠左鍵可顯示當前滑鼠的座標!',self)
        self.label.setFont(QFont('Arial',12,0,False))
        self.show()

    #重定義滑鼠事件，讓左鍵點擊時打印滑鼠座標
    def mousePressEvent(self, event):
        if event.button() == Qt.LeftButton:
            p = self.mapFromGlobal(QCursor.pos())
            self.label.setText(
                '(%d , %d)'%(p.x(),p.y()))

    #重定義鍵盤事件
    def keyPressEvent(self, event):
        if  QApplication.keyboardModifiers() == \
            Qt.ShiftModifier:
            if  event.key() == Qt.Key_A:
                self.label.setText(
                    'you press Shift+A !')

        elif event.key() == Qt.Key_F1:
            self.label.setText('you press F1 !')
```

```
        elif event.key() == Qt.Key_Return:
            self.label.setText('you press enter !')

        else:
            #使用該按鍵的預設動作
            QWidget.keyPressEvent(self, event)

    #增加離開事件(按右上角叉叉時的動作)
    def closeEvent(self, event):
        QMessageBox.warning(None,'mesage','GoodBye!')

if __name__ == '__main__':
    app = QApplication(sys.argv)
    widget = QWindow()
    sys.exit(app.exec_())
```