Multiple Document Interface
(giao diện nhiều tài liệu)

Nội dung

Chúng ta sẽ tìm hiểu nội dung như sau:

- Giao diện nhiều tài liệu
- Hiển thị thông tin trong các phần bằng Tab Widget
- Tạo một thanh menu (menu bar)

Giao Diện Nhiều Tài Liệu

- **SDI (Single Document Interface):** Mỗi cửa sổ chính chỉ hiển thị **1 tài liệu**.
- **MDI (Multiple Document Interface):** Một cửa sổ chính có thể hiển thị **nhiều tài liệu cùng lúc**.
- Trong MDI, các tài liệu được hiển thị dưới dạng **cửa sổ con (subwindow)** trong vùng trung tâm.
- **QMdiArea:** Vùng chứa và quản lý các cửa sổ con.
- **QMdiSubWindow:** Đại diện cho mỗi cửa sổ/tài liệu con (có tiêu đề, nút thu nhỏ, phóng to…).
- Các cửa sổ con có thể được **xếp chồng (cascade)** hoặc **xếp gạch (tile)**.

=> Phù hợp cho ứng dụng như: trình soạn thảo, IDE, phần mềm quản lý nhiều tài liệu.

QMdiArea (PyQt)

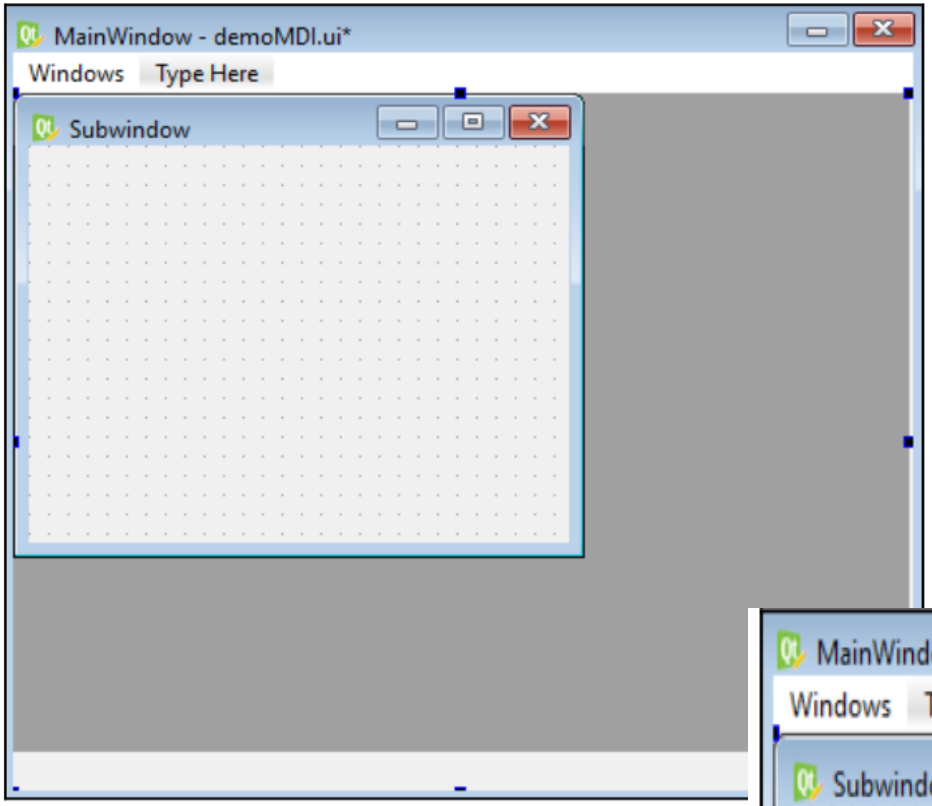
- **QMdiArea** quản lý nhiều cửa sổ con (SubWindow) trong giao diện MDI.
- **subWindowList()**: Lấy danh sách các cửa sổ con, có thể sắp xếp theo:
  - **CreationOrder**: theo thứ tự tạo (mặc định)
  - **StackingOrder**: theo thứ tự xếp chồng
  - **ActivationHistoryOrder**: theo lịch sử kích hoạt
- **activateNextSubWindow() / activatePreviousSubWindow()**: Chuyển focus sang cửa sổ kế tiếp / trước đó.
- **cascadeSubWindows()**: Sắp xếp cửa sổ con dạng **thác nước**.
- **tileSubWindows()**: Sắp xếp cửa sổ con dạng **xếp ô**.
- **closeAllSubWindows()**: Đóng tất cả cửa sổ con.
- **setViewMode()**: Chọn chế độ hiển thị:
  - **SubWindow View (0)**: Mỗi cửa sổ có khung riêng (mặc định).
  - **Tabbed View (1)**: Hiển thị bằng các tab, mỗi lần xem 1 cửa sổ.

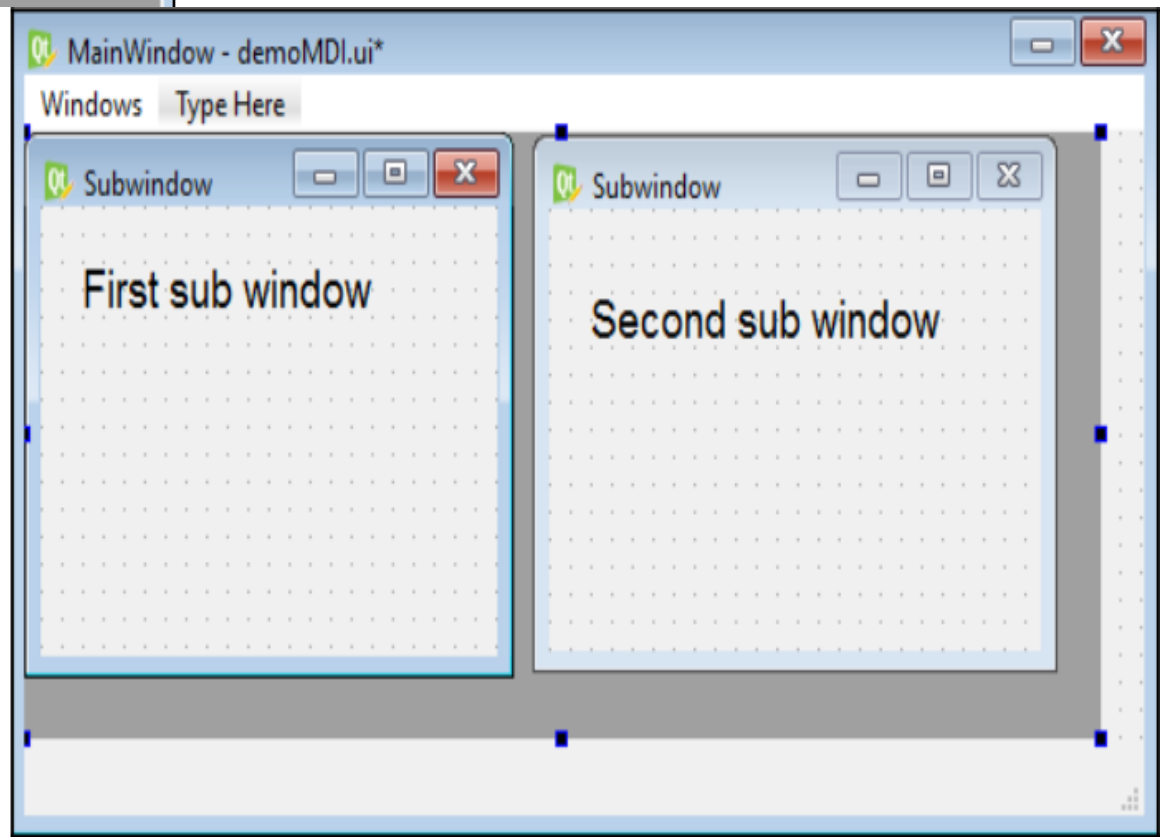
=> Phù hợp để giới thiệu nhanh cách **quản lý & sắp xếp nhiều cửa sổ** trong PyQt MDI.

Tạo ứng dụng MDI

- Tạo ứng dụng **Main Window** bằng **Qt Designer**.
- Thêm **QMdiArea** vào giao diện.
- Thêm **2 subwindow** vào QMdiArea (chuột phải → Add Subwindow).
- Mỗi subwindow hiển thị nội dung riêng (tương ứng các luồng con).
- Đặt **QLabel** trong từng subwindow để phân biệt:
  - Subwindow 1: *First subwindow*
  - Subwindow 2: *Second subwindow*

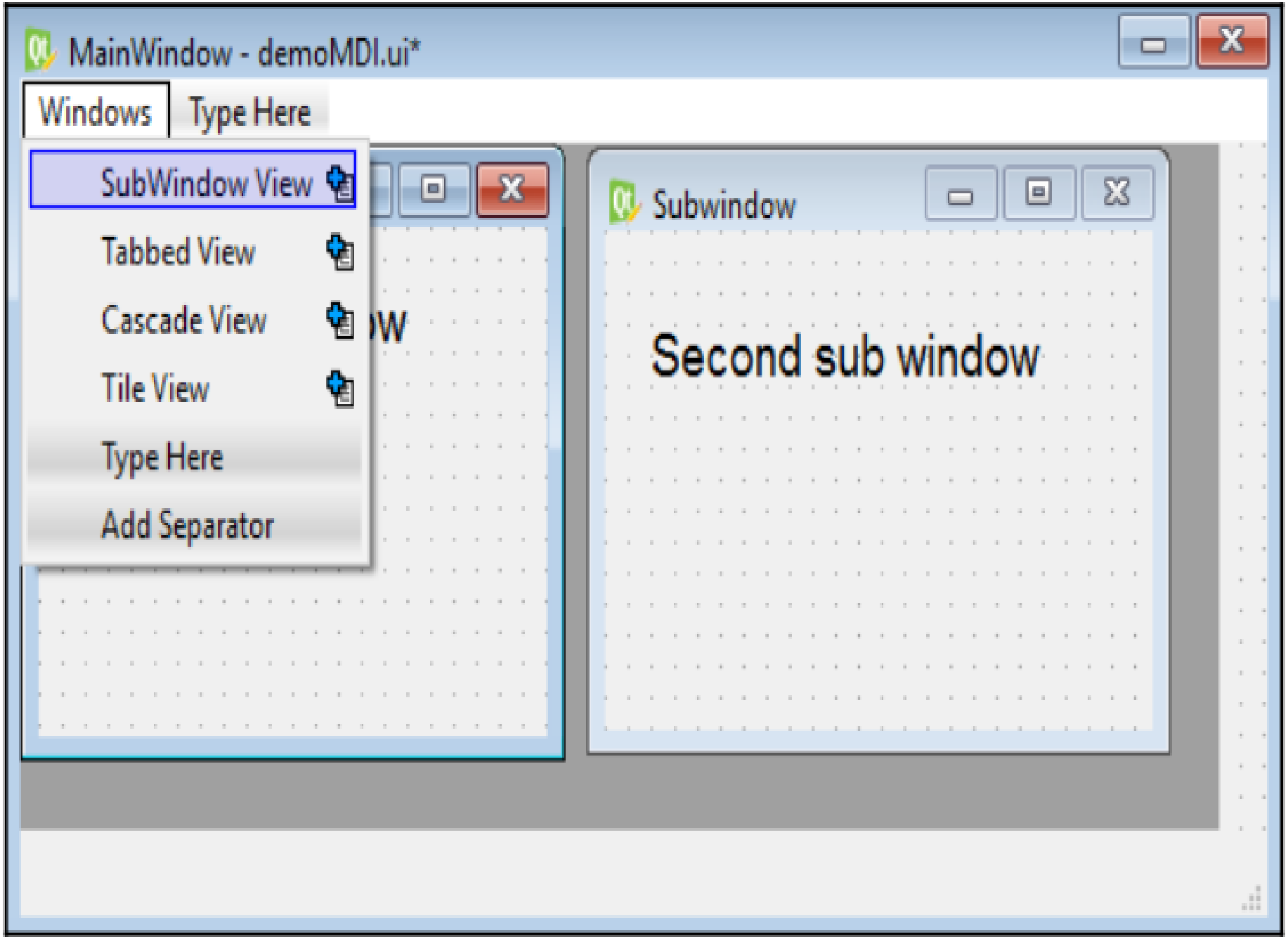
=> Giúp quản lý và sắp xếp nhiều cửa sổ con trong cùng một ứng dụng PyQt.


MainWindow - demoMDI.ui*
Windows
Type Here
Subwindow


MainWindow - demoMDI.ui*
Windows
Type Here
Subwindow
First sub window
Subwindow
Second sub window

Giao Diện Nhiều Tài Liệu

- **QMdiArea** hiển thị nhiều cửa sổ con (*subwindows*) theo 2 chế độ:
- **SubWindow View** (mặc định): các cửa sổ con có thể **xếp chồng** hoặc **xếp kiểu thác nước (cascade/tile)**, xem được **nhiều cửa sổ cùng lúc**.
- **Tabbed View**: các cửa sổ con hiển thị dưới dạng **tab**, **mỗi lần chỉ xem được một cửa sổ**.
- Có thể **chuyển đổi chế độ** bằng menu: *SubWindow View* ↔ *Tabbed View*.
- Thêm các mục menu **Cascade View** và **Tile View** để thay đổi cách sắp xếp cửa sổ con.


MainWindow - demoMDI.ui*
Windows
Type Here
SubWindow View
Tabbed View
Cascade View
Tile View
Type Here
Add Separator
Subwindow
Second sub window

Giao Diện Nhiều Tài Liệu

Lưu ứng dụng dưới dạng demoMDI.ui. Giao diện người dùng được tạo bằng Qt Designer được lưu trữ trong file .ui là file XML và cần được chuyển đổi thành code Python.

C:\Pythonbook\PyQt5>pyuic5 demoMDI.ui -o demoMDI.py

Giao Diện Nhiều Tài Liệu

- Dùng file demoMDI.py làm **code nền (template)** cho ứng dụng.
- Giao diện sử dụng **QMdiArea** để hiển thị các **subwindow** và widget con.
- Viết script mới **goiMDI.pyw**, nhập lại code từ demoMDI.py.
- Script này xử lý **menu chức năng**: xếp tầng, xếp chồng subwindow, và chuyển giữa **SubWindow View** ↔ **Tabbed View**.

Giao Diện Nhiều Tài Liệu

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication, QAction,
 QFileDialog
 from demoMDI import *
 class MyForm(QMainWindow):
     def __init__(self):
         super().__init__()
         self.ui = Ui_MainWindow()
         self.ui.setupUi(self)
         self.ui.mdiArea.addSubWindow(self.ui.subwindow)
         self.ui.mdiArea.addSubWindow(self.ui.subwindow_2)
         self.ui.actionSubWindow_View.triggered.connect
         (self.SubWindow_View)
         self.ui.actionTabbed_View.triggered.connect(self.
         Tabbed_View)
         self.ui.actionCascade_View.triggered.connect(self.
         cascadeArrange)
         self.ui.actionTile_View.triggered.connect(self.tileArrange)
         self.show()
     def SubWindow_View(self):
         self.ui.mdiArea.setViewMode(0)
     def Tabbed_View(self):
         self.ui.mdiArea.setViewMode(1)
     def cascadeArrange(self):
         self.ui.mdiArea.cascadeSubWindows()
     def tileArrange(self):
         self.ui.mdiArea.tileSubWindows()
 if __name__=="__main__":
     app = QApplication(sys.argv)
     w = MyForm()
     w.show()
     sys.exit(app.exec_())
```

Hiển Thị Thông Tin Trong Các Phần Bằng Tab Widget

- Ứng dụng minh họa một **giỏ mua hàng** gồm 3 tab:

1. Chọn sản phẩm
2. Chọn phương thức thanh toán
3. Nhập địa chỉ giao hàng

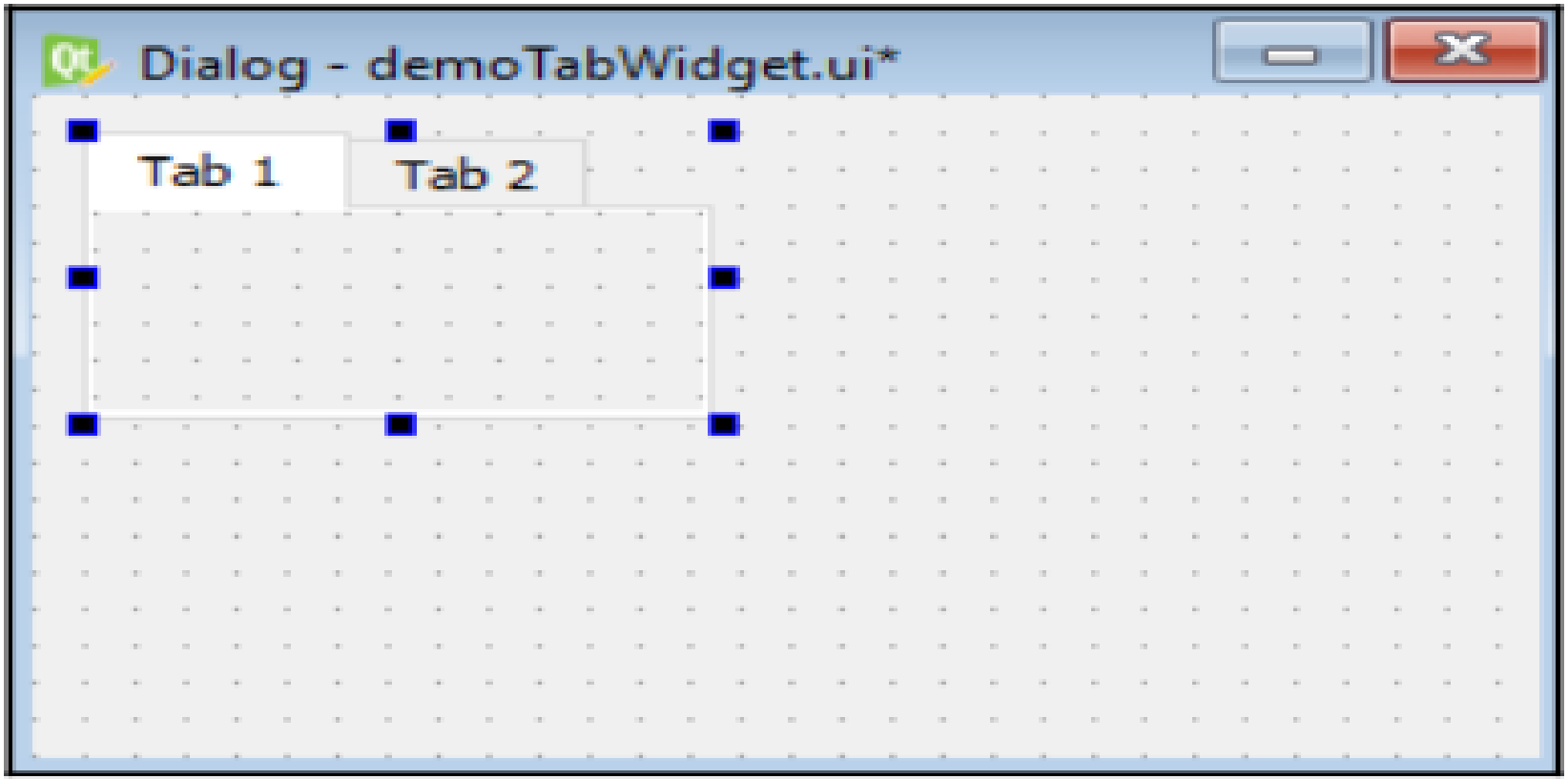
- **Tab Widget** dùng để chia nội dung thành các phần rõ ràng, dễ quản lý.
- Khi người dùng chọn một tab, **nội dung tương ứng** của tab đó sẽ được hiển thị.
- Tab Widget phù hợp khi ứng dụng có **nhiều bước hoặc nhiều loại thông tin** cần nhập/hiển thị.
- Ứng dụng được xây dựng theo **quy trình từng bước** dựa trên các tab.

Ứng dụng với Tab Widget

- Tạo ứng dụng mới với mẫu **Dialog without Buttons**.
- Thêm **Tab Widget** vào form (mặc định có **Tab1** và **Tab2**).
- Có thể **thêm/xóa tab** bằng cách nhấp chuột phải vào tab → **Insert Page (Before/After)**.
- Ứng dụng gồm **3 tab chính**:
  - **Tab 1 – Sản phẩm**: Hiển thị danh sách sản phẩm và giá, người dùng chọn và nhấn **Add to Cart**.
  - **Tab 2 – Thanh toán**: Chọn hình thức thanh toán (Debit Card, Credit Card, Net Banking, Cash).
  - **Tab 3 – Giao hàng**: Nhập địa chỉ giao hàng (địa chỉ đầy đủ, bang/tỉnh, quốc gia, số liên lạc).

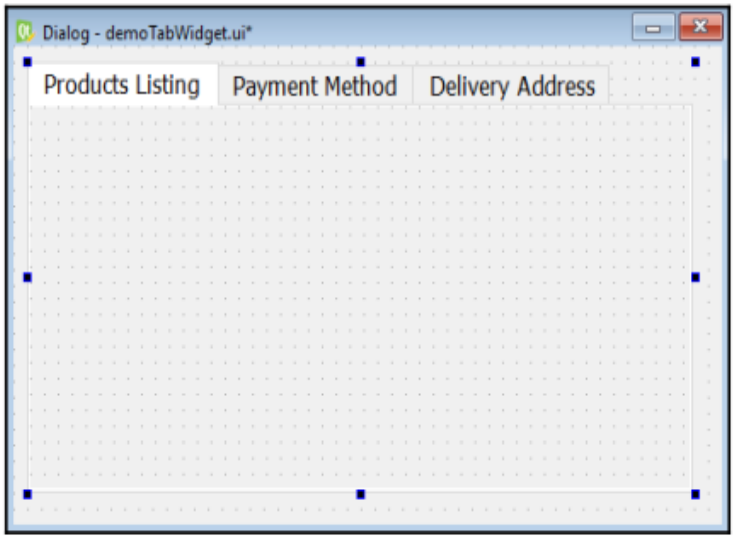
=> Phù hợp để minh họa cách sử dụng **Tab Widget trong PyQt** cho ứng dụng nhiều chức năng.

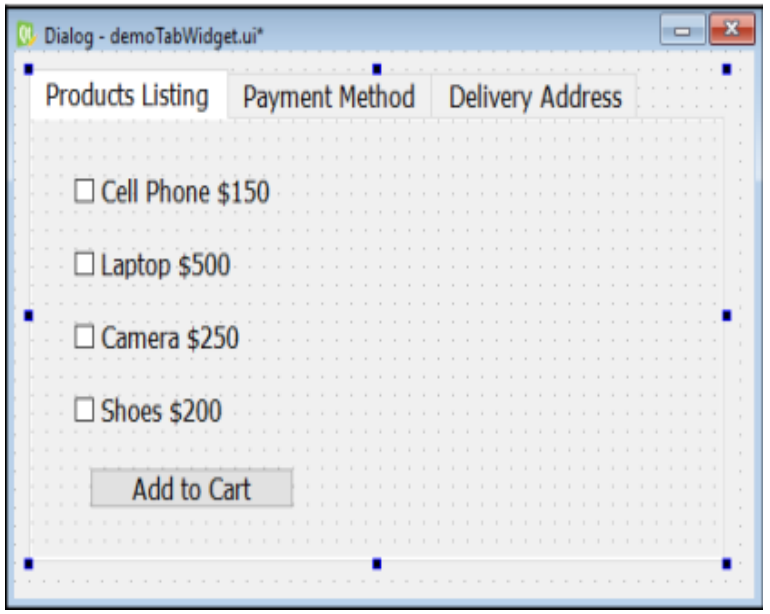
Ứng dụng với Tab Widget

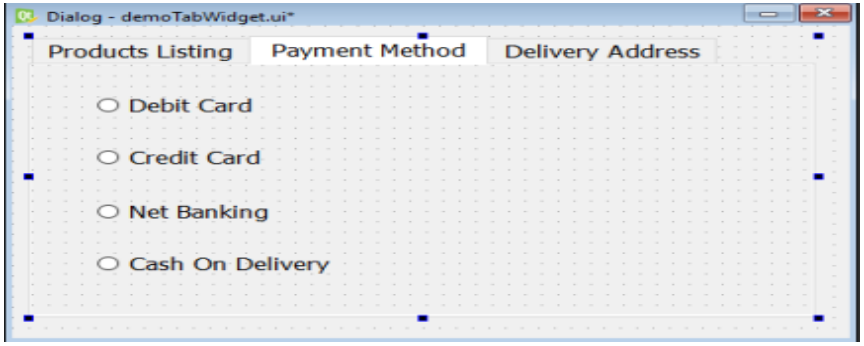


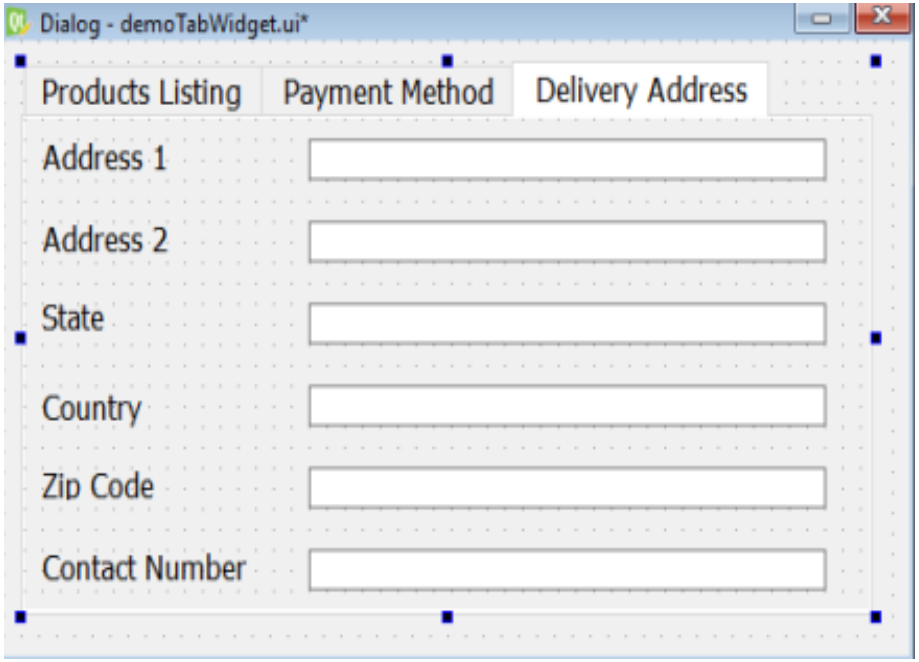
Xây dựng giao diện bán hàng nhiều tab

- Đổi tên các tab của **Tab Widget** bằng thuộc tính **currentTabText**
→ Tab 1: *Product Listing*, Tab 2: *Payment Method*
- Thêm tab mới bằng **Insert Page** → **After Current Page**
→ Đặt tên tab mới: *Delivery Address*
- Mở rộng vùng hiển thị của Tab Widget để đặt các widget bên trong
- **Tab Product Listing**:
  - Thêm 4 *CheckBox* (Cell Phone $150, Laptop $500, Camera $250, Shoes $200)
  - Thêm nút *Push Button*: **Add to Cart**
- **Tab Payment Method**:
  - Thêm 4 *Radio Button*: Debit Card, Credit Card, Net Banking, Cash
- **Tab Delivery Address**:
  - Thêm *Label* và *Line Edit* để nhập thông tin: Address 1, Address 2, State, Country, Zip Code, Contact Number


Dialog - demoTabWidget.ui*
Products Listing
Payment Method
Delivery Address


Dialog - demoTabWidget.ui*
Products Listing
Payment Method
Delivery Address
Cell Phone $150
Laptop $500
Camera $250
Shoes $200
Add to Cart


Dialog - demoTabWidget.ui*
Products Listing
Payment Method
Delivery Address
Debit Card
Credit Card
Net Banking
Cash On Delivery


Dialog - demoTabWidget.ui*
Products Listing
Payment Method
Delivery Address
Address 1
Address 2
State
Country
Zip Code
Contact Number

Hiển Thị Thông Tin Trong Các Phần Bằng Tab Widget

Lưu ứng dụng dưới dạng demoTabWidget.ui. Giao diện người dùng được tạo bằng Qt Designer được lưu trữ trong tệp .ui là tệp XML và cần được chuyển đổi thành mã Python.

C:PythonbookPyQt5>pyuic5 demoTabWidget.ui -o demoTabWidget.py

Tạo một file Python khác có tên callTabWidget.pyw và nhập mã demoTabWidget.py vào đó:

Hiển Thị Thông Tin Trong Các Phần Bằng Tab Widget

```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from demoTabWidget import *
class MyForm(QDialog):
    def __init__(self):
        super().__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.show()
if __name__=="__main__":
    app = QApplication(sys.argv)
    w = MyForm()
    w.show()
    sys.exit(app.exec_())
```

Tạo Menu Bar trong PyQt (Qt Designer)

- Ứng dụng PyQt lớn thường được chia thành **nhiều mô-đun**, mỗi mô-đun có thể được gọi thông qua **menu bar** hoặc **toolbar**.
- **Menu bar** cho phép người dùng kích hoạt các chức năng bằng cách nhấp vào **menu** và **menu item**.

Những nội dung chính cần học:

- Tạo **menu bar** trong Qt Designer
- Thêm **menu**, **menu con (submenu)**
- Thêm **menu item**, **separator (dấu phân cách)**
- Gán **shortcut (&)**, **tooltip** và **action** cho menu item
- Mỗi menu item tương ứng với **một hành động (action)**

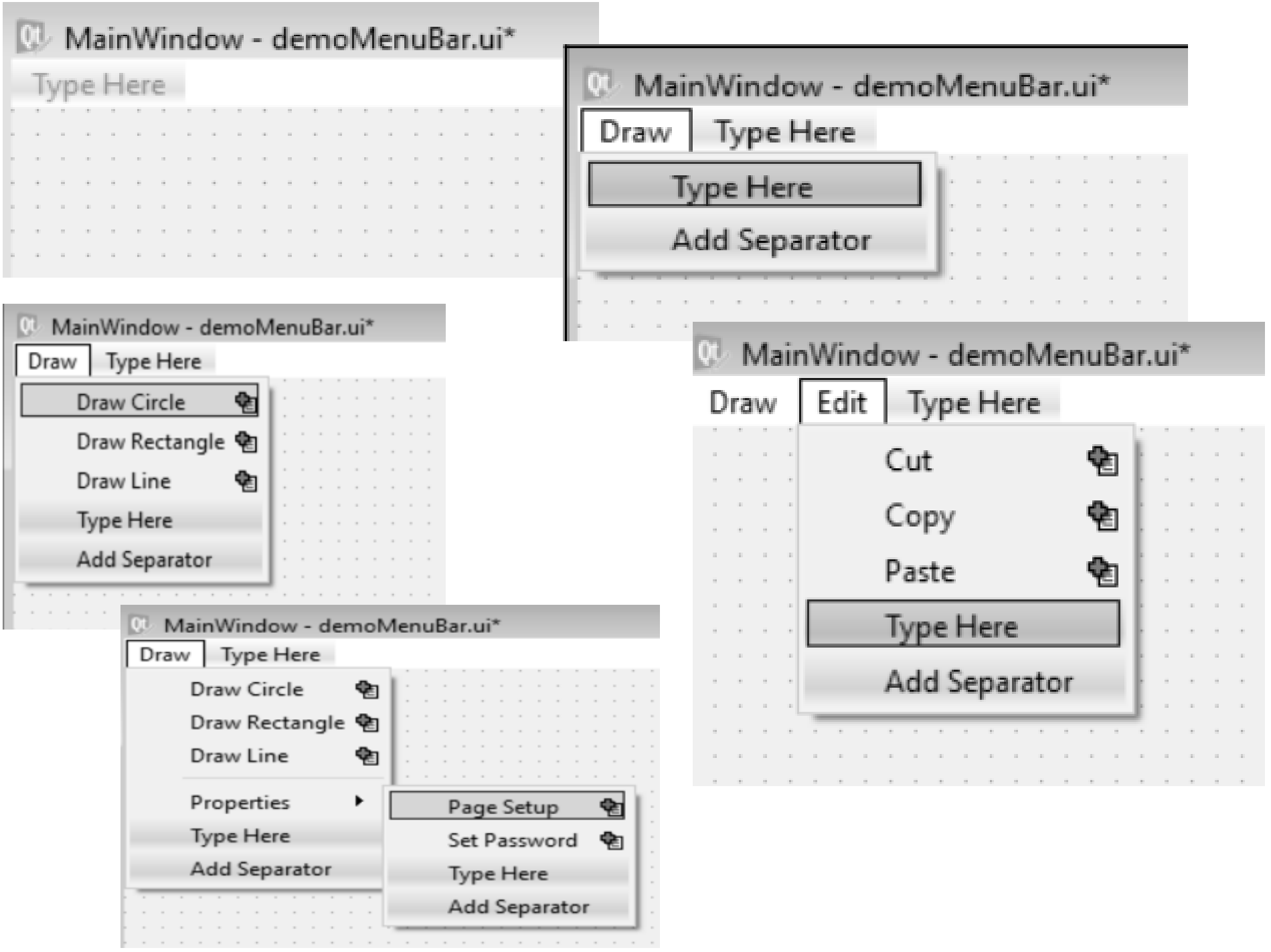
Ví dụ menu bar minh họa:

- **Menu Draw**
  - Draw Circle
  - Draw Rectangle
  - Draw Line
  - Properties
    - Page Setup
    - Set Password
- **Menu Edit**
  - Cut
  - Copy
  - Paste

Ứng dụng menu trong Qt Designer

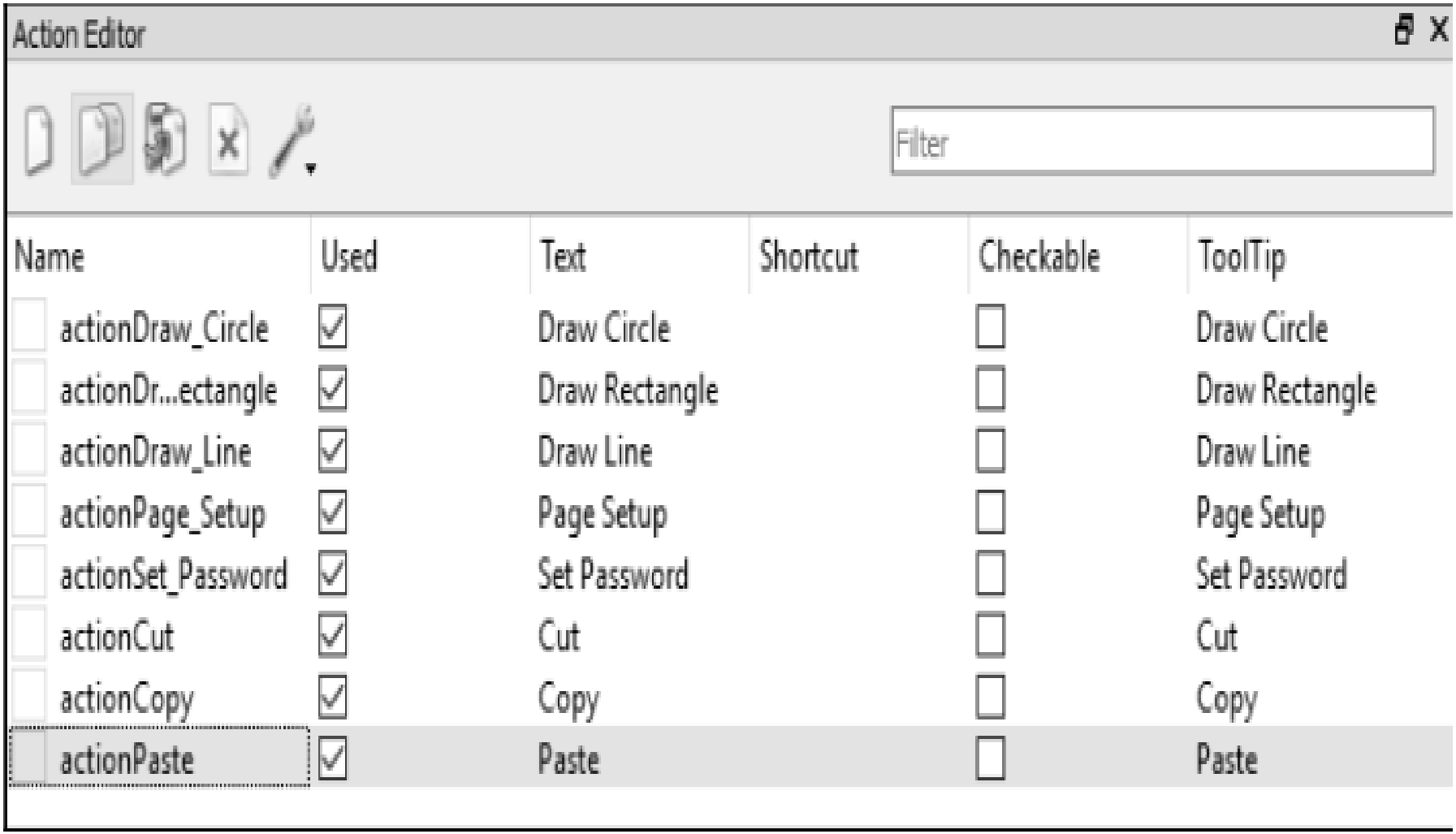
1. Tạo ứng dụng với mẫu **Main Window** (có sẵn menu bar).
2. Có thể **xóa hoặc tạo mới menu bar** bằng chuột phải.
3. Nhấp vào **Type Here** để thêm menu và menu item.
4. Dùng **Add Separator** để tạo dấu phân cách.
5. Kéo–thả để **sắp xếp menu**.
6. Mỗi menu item sẽ xuất hiện trong **Action Editor** để cấu hình chi tiết.

=> **Kết luận:** Menu bar giúp ứng dụng PyQt **rõ ràng, chuyên nghiệp và dễ mở rộng**, đặc biệt phù hợp cho các chương trình có nhiều chức năng.


MainWindow - demoMenuBar.ui*
Type Here
MainWindow - demoMenuBar.ui*
Draw
Type Here
Type Here
Add Separator
MainWindow - demoMenuBar.ui*
Draw
Type Here
Draw Circle
Draw Rectangle
Draw Line
Type Here
Add Separator
MainWindow - demoMenuBar.ui*
Draw
Edit
Type Here
Cut
Copy
Paste
Type Here
Add Separator
MainWindow - demoMenuBar.ui*
Draw
Type Here
Draw Circle
Draw Rectangle
Draw Line
Properties
Type Here
Add Separator
Page Setup
Set Password
Type Here
Add Separator

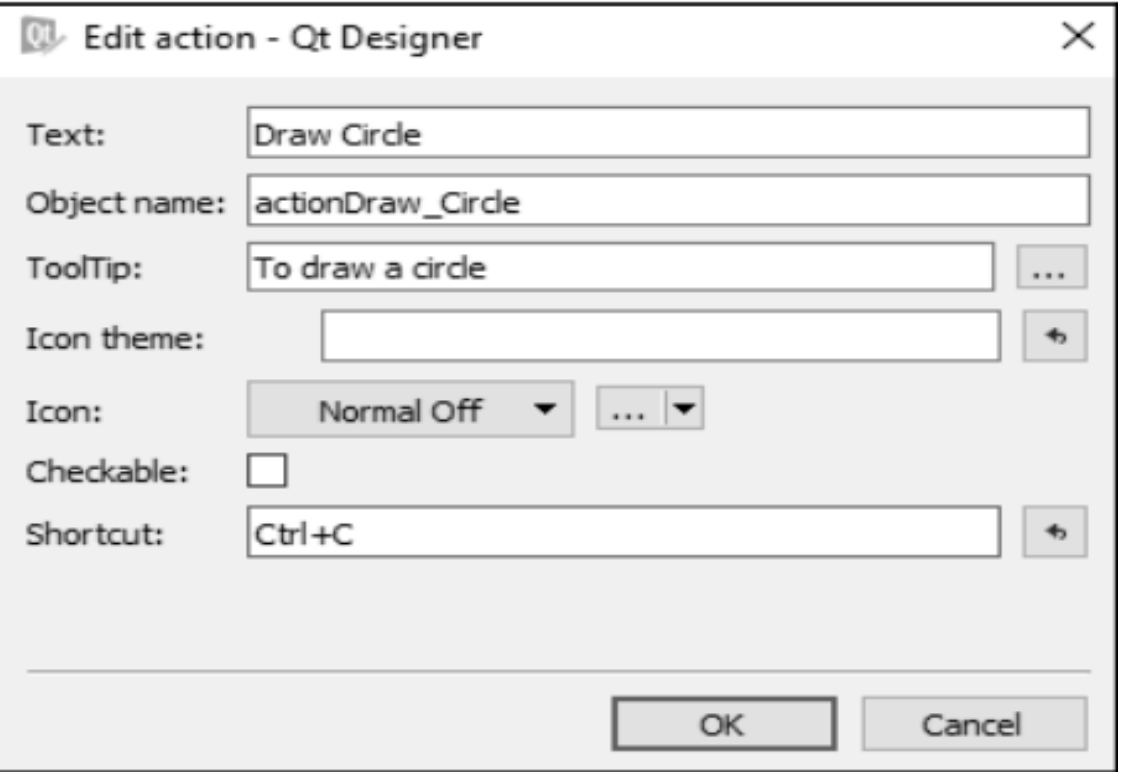
Action Editor (PyQt)

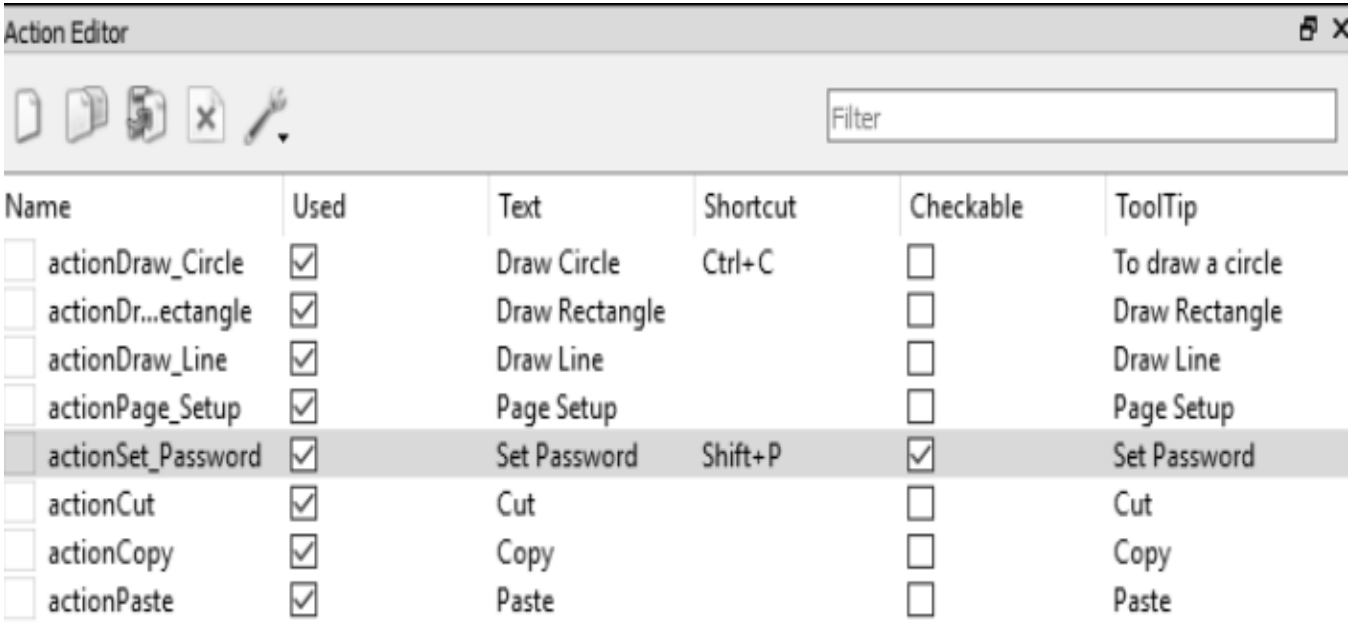
- Mỗi mục menu tương ứng với **một Action** trong **Action Editor**
- Tên Action tự động tạo từ tên menu (actionTên_Menu)
- Action dùng để cấu hình:
- Text hiển thị
- Tooltip (chú thích khi rê chuột)
- Shortcut (phím tắt)
- Checkable (bật / tắt)
- Mọi thuộc tính menu đều được quản lý tập trung trong **Action Editor**


Action Editor
Filter
Name
Used
Text
Shortcut
Checkable
ToolTip
actionDraw_Circle
Draw Circle
Draw Circle
actionDr...ectangle
Draw Rectangle
Draw Rectangle
actionDraw_Line
Draw Line
Draw Line
actionPage_Setup
Page Setup
Page Setup
actionSet_Password
Set Password
Set Password
actionCut
Cut
Cut
actionCopy
Copy
Copy
actionPaste
Paste
Paste

Menu Behavior (Hành vi Menu)

- Có thể gán **phím tắt** bằng Ctrl / Alt / Shift hoặc kết hợp
- Menu có thể đặt ở chế độ **Checkable** (menu chuyển đổi)
- Các mục **Draw Circle / Rectangle / Line** được gắn code để vẽ hình
- Các menu khác khi được chọn sẽ **hiển thị thông báo bằng Label**
- Giao diện được lưu dưới dạng **.ui** và chuyển sang Python bằng **pyuic5**


Edit action - Qt Designer
Text:
Draw Circle
Object name:
actionDraw_Circle
ToolTip:
To draw a circle
...
Icon theme:
Icon:
Normal Off
...
Checkable:
Shortcut:
Ctrl+C
OK
Cancel


Action Editor
Filter
Name
Used
Text
Shortcut
Checkable
ToolTip
actionDraw_Circle
Draw Circle
Ctrl+C
To draw a circle
actionDr...ectangle
Draw Rectangle
Draw Rectangle
actionDraw_Line
Draw Line
Draw Line
actionPage_Setup
Page Setup
Page Setup
actionSet_Password
Set Password
Shift+P
Set Password
actionCut
Cut
Cut
actionCopy
Copy
Copy
actionPaste
Paste
Paste

Chương trình tạo **menu** và hiển thị **thông báo bằng QLabel**

- Tạo file **callMothyBar.pyw**, sử dụng lại code từ **demoMothyBar.py**.
- Khi chọn các mục **Draw Circle / Draw Rectangle / Draw Line**, ứng dụng sẽ **vẽ hình tròn, hình chữ nhật hoặc đường thẳng** tương ứng.
- Mục tiêu: minh họa **xử lý sự kiện menu** và **vẽ hình cơ bản trong PyQt**.

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.QtGui import QPainter

from demoMenuBar import *

class AppWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.pos1 = [0,0]
        self.pos2 = [0,0]
        self.toDraw=""
        self.ui.actionDraw_Circle.triggered.connect(self.
        drawCircle)
        self.ui.actionDraw_Rectangle.triggered.connect(self.
        drawRectangle)
        self.ui.actionDraw_Line.triggered.connect(self.drawLine)
        self.ui.actionPage_Setup.triggered.connect(self.pageSetup)
        self.ui.actionSet_Password.triggered.connect(self.
        setPassword)
        self.ui.actionCut.triggered.connect(self.cutMethod)
        self.ui.actionCopy.triggered.connect(self.copyMethod)
        self.ui.actionPaste.triggered.connect(self.pasteMethod)
        self.show()

    def paintEvent(self, event):
        qp = QPainter()
        qp.begin(self)
        if self.toDraw=="rectangle":
            width = self.pos2[0]-self.pos1[0]
            height = self.pos2[1] - self.pos1[1]
            qp.drawRect(self.pos1[0], self.pos1[1], width, height)
```

Tạo Thanh Menu

```
        if self.toDraw=="line":
            qp.drawLine(self.pos1[0], self.pos1[1], self.pos2[0],
            self.pos2[1])
        if self.toDraw=="circle":
            width = self.pos2[0]-self.pos1[0]
            height = self.pos2[1] - self.pos1[1]
            rect = QtCore.QRect(self.pos1[0], self.pos1[1], width,
            height)
            startAngle = 0
            arcLength = 360 *16
            qp.drawArc(rect, startAngle,
            arcLength)
        qp.end()

    def mousePressEvent(self, event):
        if event.buttons() & QtCore.Qt.LeftButton:
            self.pos1[0], self.pos1[1] = event.pos().x(),
            event.pos().y()

    def mouseReleaseEvent(self, event):
        self.pos2[0], self.pos2[1] = event.pos().x(),
        event.pos().y()
        self.update()

    def drawCircle(self):
        self.ui.label.setText("")
        self.toDraw="circle"

    def drawRectangle(self):
        self.ui.label.setText("")
        self.toDraw="rectangle"
```

Tạo Thanh Menu

```
    def drawLine(self):
        self.ui.label.setText("")
        self.toDraw="line"

    def pageSetup(self):
        self.ui.label.setText("Page Setup menu item is selected")

    def setPassword(self):
        self.ui.label.setText("Set Password menu item is selected")

    def cutMethod(self):
        self.ui.label.setText("Cut menu item is selected")

    def copyMethod(self):
        self.ui.label.setText("Copy menu item is selected")

    def pasteMethod(self):
        self.ui.label.setText("Paste menu item is selected")

app = QApplication(sys.argv)
w = AppWindow()
w.show()
sys.exit(app.exec_())
```