**Creating Basic Windows and Layouts in PyQt6-PySide6**

**1. Understanding Window Structures in Qt**

In PyQt6 and PySide6, a **window** is the main container for a GUI application. It typically consists of the following components:

**a) Main Window Structure**

A **window** in Qt can be:

1. **QWidget-Based Window** (Simple window using QWidget)
2. **QMainWindow-Based Window** (Advanced window with menus, toolbars, and status bars)

**i) QWidget as a Window**

QWidget is the base class for all UI elements in Qt. It can be used to create a simple standalone window.

from PyQt6.QtWidgets import QApplication, QWidget

import sys

app = QApplication(sys.argv) # Create the application

window = QWidget() # Create a window

window.setWindowTitle("Simple Window")

window.resize(400, 300) # Set window size (width, height)

window.show() # Display the window

sys.exit(app.exec()) # Run the event loop

**ii) QMainWindow: Advanced Window**

QMainWindow is used for applications requiring menus, toolbars, and status bars.

from PyQt6.QtWidgets import QApplication, QMainWindow

import sys

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Main Window Example")

self.resize(500, 400)

app = QApplication(sys.argv)

window = MainWindow()

window.show()

sys.exit(app.exec())

from PyQt6.QtWidgets import QApplication, QMainWindow, QToolBar, QLabel

from PyQt6.QtGui import QAction # Correct import for QAction

import sys

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("QMainWindow Example")

self.resize(600, 400)

# Create Menu Bar

menu\_bar = self.menuBar()

file\_menu = menu\_bar.addMenu("File")

edit\_menu = menu\_bar.addMenu("Edit")

help\_menu = menu\_bar.addMenu("Help")

# Add Actions to Menu Bar

open\_action = QAction("Open", self)

save\_action = QAction("Save", self)

exit\_action = QAction("Exit", self)

exit\_action.triggered.connect(self.close)

file\_menu.addAction(open\_action)

file\_menu.addAction(save\_action)

file\_menu.addSeparator()

file\_menu.addAction(exit\_action)

# Create Tool Bar

tool\_bar = QToolBar("Main Toolbar")

self.addToolBar(tool\_bar)

tool\_bar.addAction(open\_action)

tool\_bar.addAction(save\_action)

# Create Status Bar

self.status\_bar = self.statusBar()

self.status\_bar.showMessage("Ready")

# Set Central Widget (Optional)

self.label = QLabel("Welcome to QMainWindow Example", self)

self.label.setStyleSheet("font-size: 16px;")

self.setCentralWidget(self.label)

app = QApplication(sys.argv)

window = MainWindow()

window.show()

sys.exit(app.exec())

**b) Components of a Window**

* **Title Bar**: Displays the window title.
* **Menu Bar**: Used in QMainWindow to provide a menu system.
* **Tool Bar**: Contains quick-access buttons.
* **Central Widget**: The main UI component placed inside a window.
* **Status Bar**: Displays status messages at the bottom.

**2. Layouts: Organizing Widgets**

Qt provides **layout managers** to arrange widgets dynamically, instead of using fixed coordinates.

**a) Types of Layouts**

1. **QVBoxLayout** (Vertical Layout): Arranges widgets in a vertical stack.
2. **QHBoxLayout** (Horizontal Layout): Arranges widgets side by side.
3. **QGridLayout** (Grid Layout): Arranges widgets in a table-like grid.
4. **QFormLayout** (Form Layout): Arranges widgets in label-input pairs.

**b) Example: Using Layouts**

Here’s how to use layouts to manage widgets dynamically.

**i) Vertical Layout (QVBoxLayout)**

from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout

import sys

app = QApplication(sys.argv)

window = QWidget()

window.setWindowTitle("Vertical Layout")

layout = QVBoxLayout() # Create a vertical layout

layout.addWidget(QPushButton("Button 1"))

layout.addWidget(QPushButton("Button 2"))

layout.addWidget(QPushButton("Button 3"))

window.setLayout(layout) # Apply the layout to the window

window.show()

sys.exit(app.exec())

* This layout stacks buttons from top to bottom.

**ii) Grid Layout (QGridLayout)**

from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QGridLayout

import sys

app = QApplication(sys.argv)

window = QWidget()

window.setWindowTitle("Grid Layout")

layout = QGridLayout() # Create a grid layout

layout.addWidget(QPushButton("Button 1"), 0, 0) # Row 0, Column 0

layout.addWidget(QPushButton("Button 2"), 0, 1) # Row 0, Column 1

layout.addWidget(QPushButton("Button 3"), 1, 0) # Row 1, Column 0

layout.addWidget(QPushButton("Button 4"), 1, 1) # Row 1, Column 1

window.setLayout(layout)

window.show()

sys.exit(app.exec())

* This layout places buttons in a 2x2 grid.

**3. Setting Up GUI Components**

GUI components (widgets) such as buttons, labels, text inputs, and combo boxes can be added inside layouts.

**a) Adding Basic Widgets**

from PyQt6.QtWidgets import QApplication, QWidget, QLabel, QPushButton, QVBoxLayout

import sys

app = QApplication(sys.argv)

window = QWidget()

window.setWindowTitle("Basic Widgets")

layout = QVBoxLayout()

layout.addWidget(QLabel("Hello, PyQt!"))

layout.addWidget(QPushButton("Click Me"))

window.setLayout(layout)

window.show()

sys.exit(app.exec())

* A **label** (QLabel) displays text.
* A **button** (QPushButton) can trigger actions.

**4. Connecting Events (Signals & Slots)**

In PyQt/PySide, user interactions are handled through **signals and slots**.

**Example: Handling Button Click**

from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout, QLabel

import sys

class MyWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Signals and Slots")

self.layout = QVBoxLayout()

self.label = QLabel("Press the button")

self.button = QPushButton("Click Me")

self.button.clicked.connect(self.change\_text) # Connect event

self.layout.addWidget(self.label)

self.layout.addWidget(self.button)

self.setLayout(self.layout)

def change\_text(self):

self.label.setText("Button Clicked!")

app = QApplication(sys.argv)

window = MyWindow()

window.show()

sys.exit(app.exec())

* .clicked.connect(self.change\_text) links the button click to a function.

**Conclusion**

* **Window structure**: QWidget for basic windows, QMainWindow for advanced applications.
* **Layouts**: QVBoxLayout, QHBoxLayout, QGridLayout, and QFormLayout help organize widgets.
* **GUI components**: Buttons, labels, text fields, and combo boxes can be added dynamically.
* **Events (Signals & Slots)**: Used to handle user interactions.