Fourth Part

ویجت‌های پیشرفته (Advanced Widgets)

در این فصل با چهار ویجت قدرتمند و پرکاربرد PyQt6 آشنا می‌شویم: **QListWidget** برای نمایش لیست‌های آیتم‌محور، **QTabWidget** برای تب‌بندی صفحات، **QTreeWidget** برای نمایش داده‌های سلسله‌مراتبی و **QTableWidget** برای نمایش جداول دوبعدی. هر ویجت را از جنبه‌ی ویژگی‌ها، متدهای مهم، و مثال کد با کامنت‌های انگلیسی بررسی می‌کنیم تا در پروژه‌های واقعی بهره‌مند شوی.

۱. QListWidget

خب ، QListWidget یک کلاس راحت برای نمایش لیست آیتم‌ها با رابط مبتنی بر آیتم است که امکان افزودن، حذف و انتخاب چندگانه را می‌دهد GeeksforGeeks. هر عنصر لیست از نوع `QListWidgetItem` است و می‌توان مشخصاتش را دلخواه تنظیم کرد Python Tutorial.

متدها و ویژگی‌های کلیدی

- `addItems(list_of_texts)` / `addItem(text)`
- `currentRow()` / `currentItem()`
- `takeItem(row)` برای حذف یک آیتم GeeksforGeeks
- `setSelectionMode(mode)` برای تعیین حالت انتخاب (Single, Multi, Extended) pythonpyqt.com
- `itemClicked` سیگنال برای واکنش به کلیک YouTube

مثال کاربردی

```
import sys
from PyQt6.QtWidgets import QApplication, QWidget, QListWidget

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('QListWidget Example')
window.resize(300, 200)
```

```
list_widget = QListWidget(window)
list_widget.addItems(['Item A', 'Item B', 'Item C'])  # add multiple item
list_widget.setSelectionMode(QListWidget.MultiSelection)  # allow multi-s

def on_item_clicked(item):
    print('Clicked:', item.text())  # print clicked item text

list_widget.itemClicked.connect(on_item_clicked)  # connect signal

list_widget.resize(280, 160)
list_widget.move(10, 10)
window.show()
sys.exit(app.exec())
```

۲. QTabWidget

باید بدانیم QTabWidget ویجتی برای مدیریت چند صفحه (tab) است، هر تب می‌تواند یک layout یا ویجت‌های دیگر را در خود جای دهد GeeksforGeeks. با متدهای `addTab()` و `insertTab()` می‌توان تب‌ها را پویا مدیریت کرد TutorialsPoint.

متدها و ویژگی‌های کلیدی

- `insertTab(index, widget, title)` / `addTab(widget, title)`
- `setCurrentIndex(index)` / `removeTab(index)`
- `tabBar().setTabsClosable(True)` برای اجازه بستن تب‌ها
- `currentChanged` سیگنال هنگام تغییر تب YouTube

مثال کاربردی

```
import sys
from PyQt6.QtWidgets import QApplication, QWidget, QTabWidget, QLabel, Q\

app = QApplication(sys.argv)
window = QWidget()
```

```
window.setWindowTitle('QTabWidget Example')
window.resize(400, 250)

tabs = QTabWidget(window)

# first tab content
tab1 = QWidget()
layout1 = QVBoxLayout(tab1)
layout1.addWidget(QLabel('This is Tab 1'))
tabs.addTab(tab1, 'Tab One')  # add first tab

# second tab content
tab2 = QWidget()
layout2 = QVBoxLayout(tab2)
layout2.addWidget(QLabel('Content of Tab 2'))
tabs.addTab(tab2, 'Tab Two')  # add second tab

tabs.resize(380, 230)
tabs.move(10, 10)
window.show()
sys.exit(app.exec())
```

۳. QTreeWidget

برای نمایش داده‌های سلسله‌مراتبی (مانند فایل‌سیستم یا ساختار درختی) مناسب است. QTreeWidget می‌توان ستون‌های متعدد و آیتم‌های فرزند تعریف کرد Qt Documentation. هر گره از نوع `QTreeWidgetItem` است و می‌تواند زیردسته داشته باشد Stack Overflow.

متدها و ویژگی‌های کلیدی

- `setHeaderLabels(list_of_labels)` / `setColumnCount(n)`
- `addChild(child_item)` / `addTopLevelItem(item)`
- `childCount()` / `topLevelItemCount()`
- `itemClicked` سیگنال برای واکنش به کلیک روی گره

مثال کاربردی

```python
import sys
from PyQt6.QtWidgets import QApplication, QWidget, QTreeWidget, QTreeWidg

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('QTreeWidget Example')
window.resize(400, 300)

tree = QTreeWidget(window)
tree.setColumnCount(2)
tree.setHeaderLabels(['Name', 'Value'])  # set column headers

# create root item
root = QTreeWidgetItem(tree, ['Root', '0'])
# create child items
child1 = QTreeWidgetItem(root, ['Child 1', '100'])
child2 = QTreeWidgetItem(root, ['Child 2', '200'])
root.addChild(child1)
root.addChild(child2)

tree.expandAll()  # expand all nodes
tree.resize(380, 280)
tree.move(10, 10)

def on_item_click(item, col):
    print(f'Clicked on {item.text(col)} in column {col}')

tree.itemClicked.connect(on_item_click)

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

۴. QTableWidget

کاربرد QTableWidget برای نمایش و ویرایش داده‌ها در قالب جدول (ردیف × ستون) کاربرد دارد. هر خانه از نوع `QTableWidgetItem` است و می‌توان سلول‌ها را ادغام (span) یا سفارشی کرد GeeksforGeeks.

متدها و ویژگی‌های کلیدی

- `setColumnCount(c)` / `setRowCount(r)`
- `setVerticalHeaderLabels(list)` / `setHorizontalHeaderLabels(list)`
- `setItem(row, col, QTableWidgetItem(value))`
- `setSpan(row, col, rowspan, colspan)` برای ادغام سلول‌ها
- `cellClicked` سیگنال برای واکنش به کلیک روی سلول Qt Documentation

مثال کاربردی

```
import sys
from PyQt6.QtWidgets import QApplication, QWidget, QTableWidget, QTableWi

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('QTableWidget Example')
window.resize(450, 300)

table = QTableWidget(window)
table.setRowCount(3)
table.setColumnCount(3)
table.setHorizontalHeaderLabels(['A', 'B', 'C'])  # set headers

# fill table with data
for i in range(3):
    for j in range(3):
        item = QTableWidgetItem(f'{i},{j}')
        table.setItem(i, j, item)

# span a cell (make it cover 2 columns)
table.setSpan(2, 0, 1, 2)  # row=2, col=0, rowspan=1, colspan=2

table.resize(430, 280)
table.move(10, 10)

def on_cell_click(row, col):
    print(f'Cell clicked at ({row}, {col})')
```

```
28
29 table.cellClicked.connect(on_cell_click)
30
31 window.show()
32 sys.exit(app.exec())
```

تمرین‌های پیشنهادی

۱. `QListWidget` : یک دکمه اضافه کن که آیتم انتخاب‌شده را با `takeItem` حذف کند.

۲. `QTabWidget` : امکان افزودن تب پویا با کلیک یک دکمه بساز (use `addTab`).

۳. `QTreeWidget` : عمق درخت را تا سه سطح افزایش بده و برای هر گره آیکن اختصاص بده.

۴. `QTableWidget` : سلول‌های یک ستون را قابل ویرایش (`setFlags`) کن و داده‌ها را در فایل CSV ذخیره نما.