Third Part

مکانیزم سیگنال‌ها و اسلات‌ها

سیگنال‌ها پیام‌هایی هستند که هنگام وقوع رویدادی در یک ویجت (مثل کلیک دکمه) منتشر می‌شوند و اسلات‌ها توابع یا متدهایی هستند که به این سیگنال‌ها واکنش می‌دهند. با استفاده از `connect()` و `disconnect()` می‌توان سیگنال‌ها را به اسلات‌ها متصل یا از هم جدا کرد. همچنین امکان تعریف سیگنال‌های سفارشی با `pyqtSignal` و اسلات‌های تایپ‌شده با `pyqtSlot` وجود دارد. این مکانیزم کاملاً type-safe است و حتی در محیط‌های چندنخی نیز قابل‌اعتماد عمل می‌کند.

۱. مفهوم سیگنال و اسلات

- در یک برنامه‌ی GUI، **سیگنال** زمانی صادر (emit) می‌شود که رویدادی رخ دهد (مثلاً کاربر دکمه را کلیک کند) ZetCode.
- **اسلات** هر تابع یا متدی است که می‌تواند به یک سیگنال متصل شده و هنگام انتشار آن اجرا شود ZetCode.
- این مکانیزم، ارتباط میان اجزای مختلف را بدون وابستگیِ مستقیم آن‌ها ممکن می‌سازد (loosely coupled) Qt Documentation.

۲. اتصال و انتشار سیگنال‌های داخلی

۲.۱ اتصال سیگنال به اسلات

برای متصل کردن سیگنال یک ویجت به تابع دلخواه از متد `connect` استفاده می‌کنیم:

```
button.clicked.connect(on_click)    # when button is clicked, on_click()
```

هر بار که کاربر دکمه را کلیک کند، سیگنال `clicked` منتشر و تابع `on_click` اجرا می‌شود ZetCode.

۲.۲ قطع ارتباط

برای قطع اتصال می‌توان از `disconnect()` بهره برد:

```
button.clicked.disconnect(on_click)  # stop calling on_click when clicked
```

این کار باعث می‌شود اسلات دیگر به انتشار سیگنال واکنش ندهد Qt Documentation.

۳. سیگنال‌های سفارشی

۳.۱ تعریف با `pyqtSignal`

می‌توان در کلاس‌های مشتق از `QObject` سیگنال‌های جدید تعریف کرد:

```
from PyQt6.QtCore import QObject, pyqtSignal

class Emitter(QObject):
    data_ready = pyqtSignal(str)     # define a signal with a string para
```

این سیگنال می‌تواند حاوی داده‌های دلخواه باشد و هنگام انتشار، آنها را به اسلات‌ها بفرستد Medium.

۳.۲ انتشار سیگنال

برای ارسال سیگنال از متد `emit` استفاده کنید:

```
emitter = Emitter()
emitter.data_ready.connect(handle_data)
emitter.data_ready.emit('Hello')  # publish the signal with argument 'Hel
```

تابع `handle_data` این مقدار را دریافت خواهد کرد w3resource.

۴. اسلات‌های تایپ‌شده با `pyqtSlot`

با دکوراتور `@pyqtSlot` می‌توان اسلات‌ها را نوع‌بندی کرد که به بهینه‌سازی و خوانایی کمک می‌کند:

```
from PyQt6.QtCore import pyqtSlot

@pyqtSlot(str)
def handle_data(text):
    print('Received:', text)
```

این اعلان صریح نوع پارامتر را مشخص می‌کند و از overloading سیگنال/اسلات پشتیبانی می‌کند Stack Overflow.

۵. منتقل کردن داده میان ویجت‌ها

می‌توان سیگنال‌های ویجت‌ها را به اسلات‌های ویجت یا توابع دیگر متصل کرد تا داده انتقال یابد:

```
line_edit.textChanged.connect(label.setText)  # update label as user type
```

هر بار متن تغییر کند، label با آن مقدار به‌روزرسانی می‌شود Python GUIs.

۶. ایمنی در چندنخی (Thread Safety)

سیگنال‌ها و اسلات‌های PyQt6 به صورت thread-safe پیاده‌سازی شده‌اند، بنابراین می‌توان از آن‌ها برای ارتباط میان Threadها استفاده کرد بدون نگرانی از شرایط رقابتی Python Tutorials – Real Python.

۷. مثال جامع

```
import sys
from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QLabel, Q
from PyQt6.QtCore import QObject, pyqtSignal, pyqtSlot

class Worker(QObject):
    progress = pyqtSignal(int)  # custom signal

    def run(self):
        for i in range(5):
            # Simulate work...
```

```
            self.progress.emit(i)

class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Signals & Slots Demo')

        self.label = QLabel('Progress: 0', self)
        self.button = QPushButton('Start', self)
        self.button.clicked.connect(self.start_work)

        layout = QVBoxLayout(self)
        layout.addWidget(self.label)
        layout.addWidget(self.button)

        self.worker = Worker()
        self.worker.progress.connect(self.update_label)

    def start_work(self):
        self.worker.run()

    @pyqtSlot(int)
    def update_label(self, value):
        self.label.setText(f'Progress: {value}')

app = QApplication(sys.argv)
window = MainWindow()
window.show()
sys.exit(app.exec())
```

- سیگنال `clicked` دکمه به متد `start_work` متصل است.
- سیگنال سفارشی `progress` با اسلات `update_label` ارتباط دارد.
- `@pyqtSlot(int)` تضمین می‌کند که اسلات فقط عدد صحیح دریافت کند.

تمرین‌ها

۱. یک سیگنال سفارشی با دو پارامتر (`str` و `int`) تعریف و منتشر کن.

۲. از `()disconnect` برای غیرفعال کردن موقت اسلات استفاده کن.

۳. سیگنال یک Thread واقعی (`QThread`) را به یک اسلات در `MainWindow` متصل کن تا پیشرفت را در رابط نمایش دهی.