Qt

Bartosz Szreder

Co to jest?

- Zorientowana obiektowo biblioteka C++.
- Działa na Linuksie, OS X, Windows oraz niektórych systemach embedded i mobilnych.
- LGPL 2.1

Główne moduły

- Core, Network, SQL, Test
- GUI (Qt5), Widgets, Multimedia, WebKit
- Quick (Qt5), QML (Qt5)
- Dodatkowo: Concurrent, D-Bus, OpenGL, Script, SVG, XML, JSON.
- IDE: QtDesigner, QtCreator

Architektura

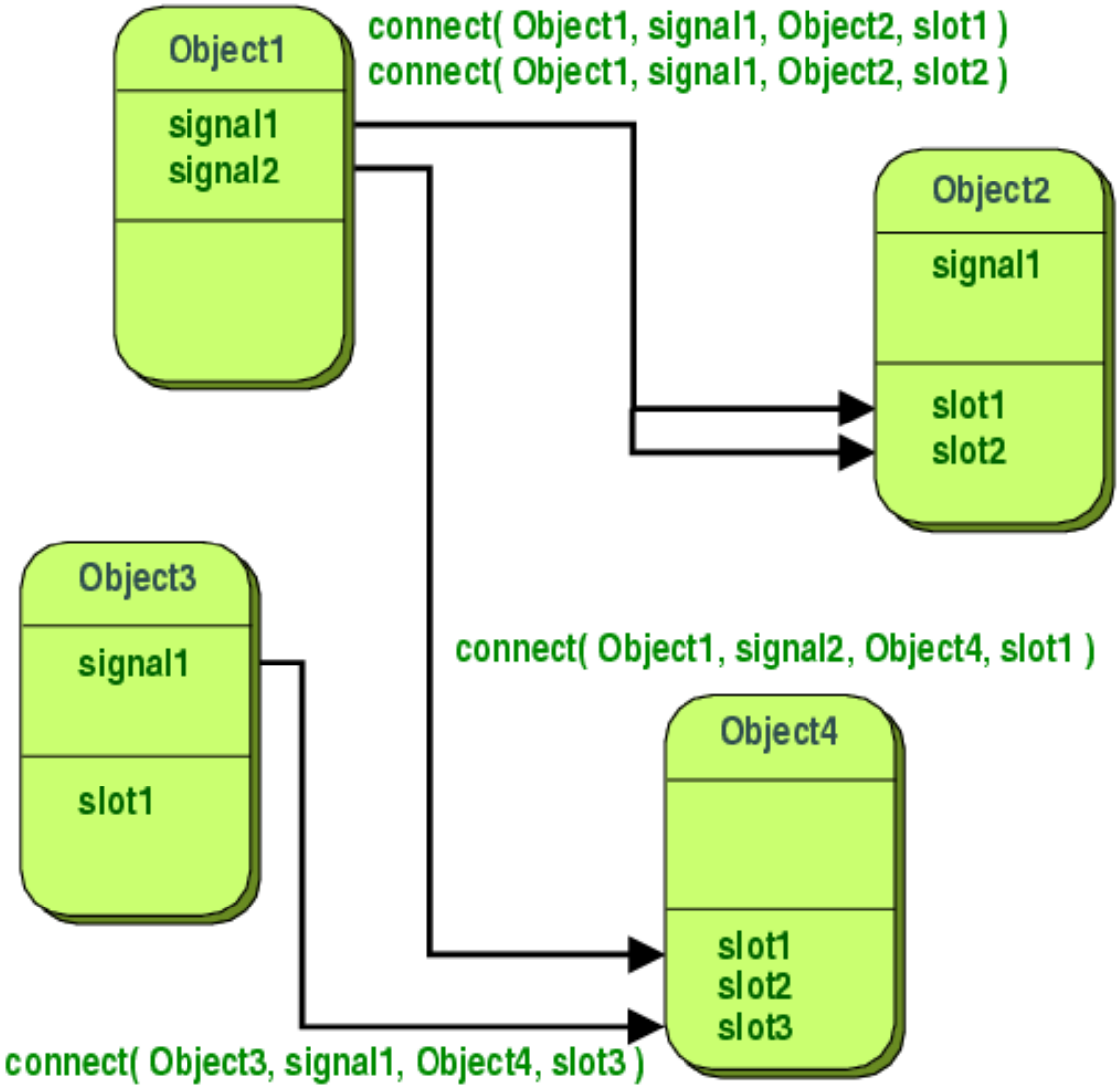
- Podstawowa klasa: `QObject`. Obiekty można łączyć w drzewa.
- Podstawowa klasa „graficzna”: `QWidget`.
- Event-driven programming (ale można też pracować bez event loop).
- Rozszerzenie języka C++ o system własności (properties) oraz sygnały i sloty.

Hello world

```
1
2   #include <QtGui/QApplication>
3   #include <QtGui/QPushButton>
4
5   int main(int argc, char *argv[])
6   {
7       QApplication application(argc, argv);
8       QPushButton button("Hello world");
9       button.show();
10      return application.exec();
11  }
12
13
```

- `QApplication.exec()` odpala event loop.
- Jeśli piszemy program konsolowy, to wystarczy `QCoreApplication`.
- W Qt5 jest jeszcze coś pomiędzy: `QGuiApplication` – chcemy część podsystemu graficznego, ale bez QWidgetów (albo z ograniczeniami).

Signals and slots



```
#include <QObject>

class Counter : public QObject
{
    Q_OBJECT

public:
    Counter() { m_value = 0; }

    int value() const { return m_value; }

public slots:
    void setValue(int value);

signals:
    void valueChanged(int newValue);

private:
    int m_value;
};
```

```
void Counter::setValue(int value)
{
    if (value != m_value) {
        m_value = value;
        emit valueChanged(value);
    }
}
```

Łączenie sygnałów i slotów dawniej:

```
QObject::connect(&a, SIGNAL(valueChanged(int)),
    &b, SLOT(setValue(int)));
```

...i w Qt5:

```
QObject::connect(&a, &Counter::valueChanged,
    &b, &Counter::setValue);
```

Czasami możemy chcieć używać starej składni łączenia, w szczególności gdy mamy przeładowane sygnały lub sloty i musielibyśmy użyć jawnego rzutowania:

```
QSignalMapper mapper;
QObject::connect(&mapper,
    static_cast<void (QSignalMapper::*)(int)>
    (&QSignalMapper::mapped), ...)
```

- Można też przyłączać sygnały do sygnałów, nie tylko do slotów...
- ...a nawet do funktorów i lambda-funkcji.
- Połączenia są jeden-do-wielu, uruchamiane w kolejności przyłączania.

Postulat: sygnały i sloty są The Right Way™ w kategorii oddzielania warstw i modułów w kodzie.

- Bardziej eleganckie od callbacków (niestety nieco droższe).
- Warstwy „niższe” nie muszą wiedzieć absolutnie niczego o reszcie świata.
- Połączenia tworzymy top-down: w warstwach wyższych nasłuchujemy sygnałów od warstw niższych.
- Połączenia mogą być wykonywane natychmiastowo lub być opóźnione do momentu powrotu do event loop.
- Działają między wątkami.