

Qt4 - podstawy

13.11.2008

- Qt jest multiplatformowym zestawem bibliotek i narzędzi, dostępnych na: Linux/X11, Embedded Linux, S60 (w najbliższym czasie), Mac OS, Windows, Windows CE,
- Początkowo tworzona przez firmę Trolltech, aktualnie przejęta przez firmę Qt Software (Nokia).
- Pierwsi twórcy Qt to: Haavard Nord i Eirik Chambe-Eng, pierwsze linie kodu napisane w 1992 roku.
- Nazwali bibliotekę Qt, ponieważ literka 'Q' ładnie wygląda w czcionce Haavarda pod Emacssem, a 't' zaporzyczyli od Xt (X toolkit). (*Wikipedia ang.*)

- Qt Commercial License
- Qt Open Source License
- Qt Educational License
- Qt Academic License

Projekty oparte o Qt

- KDE
- Opera
- Gadu-Gadu (wer. 8)
- Skype
- Google Earth
- Picasa
- Adobe Photoshop Album
- Photoshop Elements
- VLC Media Player
- ...

Wspierane języki programowania

- C++
- Java (*Qt Jambi*)
- C# (*Qyoto*)
- Python (*PyQt*)
- Perl (*PerlQt*)
- Ruby (*RubyQt*)
- PHP (*Qt-PHP*)
- Ada (*QtAda*)
- Pascal

- Graficzne
 - Qt Designer
 - Qt Assistant
 - Qt Linguist (lupdate, lrelease)
 - Qt Creator
- Prekompilacji
 - qmake (qmake-qt4)
 - Meta-Object Compiler (moc)
 - User Interface Compiler (uic)
 - Resource Compiler (rcc)

Podział na Moduły

QtCore	QtGui	QtUiTools
QtNetwork	QtOpenGL	QtTest
QtScript	QtSql	QtHelp
QtSvg	QtWebKit	QtDesigner
QtXml	QtXmlPatterns	QtAssistant
Phonon	Qt3Support	QAxContainer
		QAxServer
		QtDBus

QTL (Tulip) vs. STL

Kontenery sekwencyjne:

QList

QLinkedList

QVector

Kolejki:

QStack

QQueue

Inne:

QVarLengthArray

QCache

QPair

Kontenery asocjacyjne:

QMap

QMultiMap

QHash

QMultiHash

QSet

'Nie wzorcowe':

QBitArray

QByteArray

QString

QStringList

STL-style vs. JAVA-style

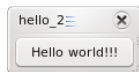
```
// styl STL

QList<int>::iterator i;
i = list.begin();
while( i != list.end() ) {
    if( *i == 0 )
        i = list.erase(i);
    else if( *i < 0 ) {
        *i = -*i;
        ++i;
    }
}
```

```
// styl Javy

QMutableListIterator<int> i(list);
while( i.hasNext() ) {
    int val = i.next();
    if( val == 0 )
        i.remove();
    else if( val < 0 )
        i.setValue( - val );
}
```

Nieśmiertelne Hello world!!!



```
#include <QApplication>
#include <QPushButton>
#include <QObject>

int main( int argc, char * argv [])
{
    QApplication app( argc, argv );
    QPushButton button( "Hello world!!!" );
    QObject::connect( & button, SIGNAL( clicked() ),
                    qApp,      SLOT( quit() ));
    button.show();
    return app.exec();
}
```

*Tworzenie plików *.pro:*

```
qmake -project QT+=opengl
```

*Tworzenie plików Makefile na podst. plików *.pro:*

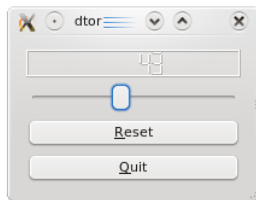
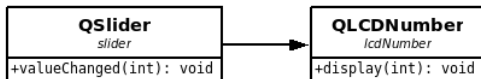
```
qmake
```

Kompilacja:

```
make
```

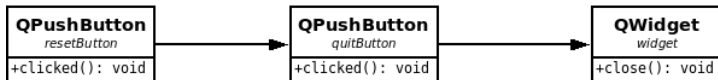
Sygnaly i sloty - większy przykład

```
connect(  
    slider, SIGNAL( valueChanged(int) ),  
    lcdNumber, SLOT( setValue(int) ) );
```



```
connect(  
    resetButton, SIGNAL( clicked() ),  
    quitButton, SIGNAL( clicked() ) );
```

```
connect(  
    quitButton, SIGNAL( clicked() ),  
    this, SIGNAL( close() ) );
```



Klasa z obsługą sygnałów

```
#include <QObject>

class Object : public QObject
{
    Q_OBJECT
public:
    Object(QObject * parent = 0) : QObject(parent) {}

signals:
    void textChanged(const QString &);

public slots:
    void setText(const QString & newText) {
        if( text == newText ) return;
        text = newText;
        emit textChanged( text );
    }

private:
    QString text;
};
```