Dzisiejszy wykład

- System okienkowy interfejsu użytkownika QT

QT

- Wieloplatformowa biblioteka *open source* do budowy interfejsów użytkownika
- Dostępna pod adresem http://www.trolltech.com
- Pojedyncze API (Application Programming Interface) używane w wielu środowiskach (Unix, Windows, Mac)
  - Dostosowanie aplikacji do innego środowiska wymaga jedynie rekompilacji
- Kompilacja w wielu środowiskach ułatwia wykrycie błędów programisty nie ujawniających się przy pracy z konkretnym kompilatorem pod konkretnym systemem operacyjnym
- Wygląd aplikacji zgodny z innymi programami pracującymi w danym środowisku

Pojedynczy przycisk

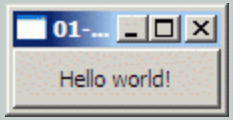
Pojedynczy przycisk:

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

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QPushButton hello("Hello world!");
    hello.resize(100, 30);

    hello.show();
    return app.exec();
}
```

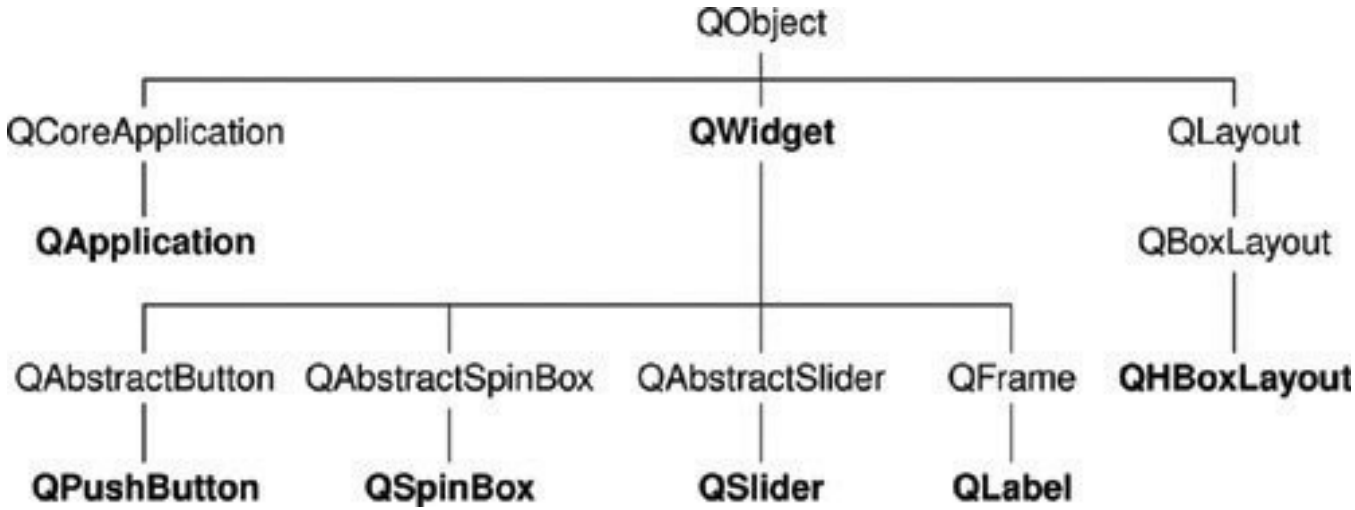


Kompilacja:

```
qmake -project
qmake
make
```

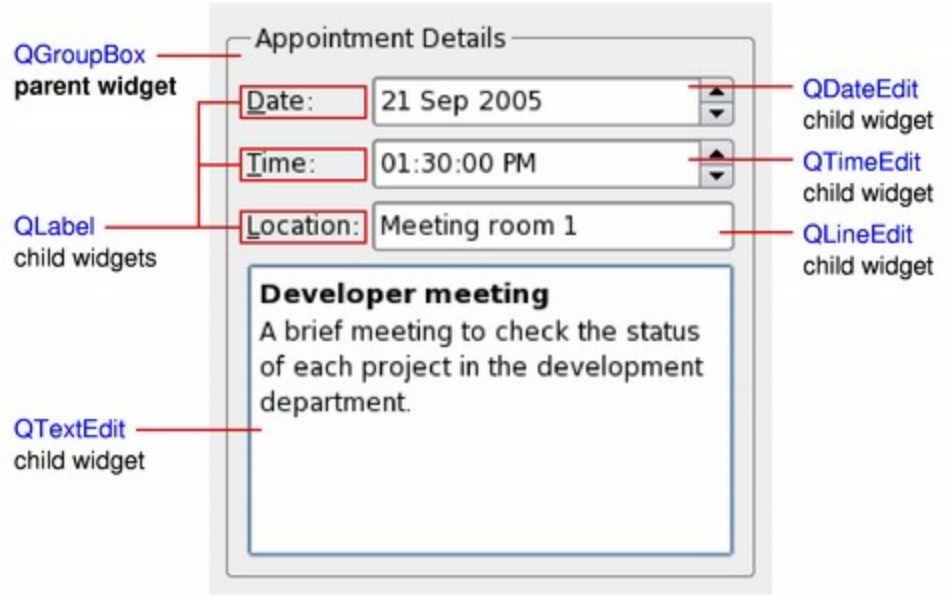
Widget

- Widoczny element interfejsu użytkownika
- Nazwa to skrót od *window gadget*
- Przyciski, menu, ramki, suwaki to wszystko widgety
- Widget może zawierać wewnątrz inne widgety



Widgety potomne i rodzicielskie

- Widget bez widgetu rodzicielskiego jest zawsze osobnym oknem
- Pozostałe widgety są wyświetlane wewnątrz widgetu rodzicielskiego
- Widget rodzicielski zwolni pamięć zaalokowaną na widget wstawiony do niego



Internationalization

- Zaznaczyć teksty do przetłumaczenia:

```
QPushButton(tr("Quit"));
```

- Zainstalować translator w programie:

```
#include <QtGui>
...
int main(int argc, char *argv[])
{
  QApplication app(argc, argv);
  QString locale = QLocale::system().name();
  QTranslator translator;
  translator.load(QString("app_") + locale);
  app.installTranslator(&translator);
```

- Zmodyfikować plik projektu: `TRANSLATIONS=app_pl.ts`
- Wyekstrahować łańcuchy: lupdate
- Wpisać tłumaczenia: linguist
- Skompilować bazę danych: lrelease

Sygnały i sloty

```
#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;
};
```

```
#include "sigslots.h"
#include <iostream>
using namespace std;
void Counter::setValue(int value)
{
     if (value != m_value) {
          m_value = value;
          emit valueChanged(value);
     }
}
int main()
{
 Counter a, b;
 QObject::connect(&a, SIGNAL(valueChanged(int)),
                  &b, SLOT(setValue(int)));
 a.setValue(12);
 cout << "a="<<a.value()<<",b="<<b.value()<<endl;
 b.setValue(48);
 cout << "a="<<a.value()<<",b="<<b.value()<<endl;
}
```