

# Lecture Material

# QT graphical user interface framework

# QT

- # Multiplatform GUI (Graphical User Interface) library
- # Available at <http://www.trolltech.com>
- # Single API (Application Programming Interface) used in multiple environments (Unix, Windows, Mac)
  - Adaptation to a different environment requires only a recompilation
  - Compilation in multiple environment makes it easier to detect errors not surfacing when working with a given compiler under a given operating system
- # Look of the application consistent with other programs working in given environment

# Simple Button

## # Simple button:

```
#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();
}
```

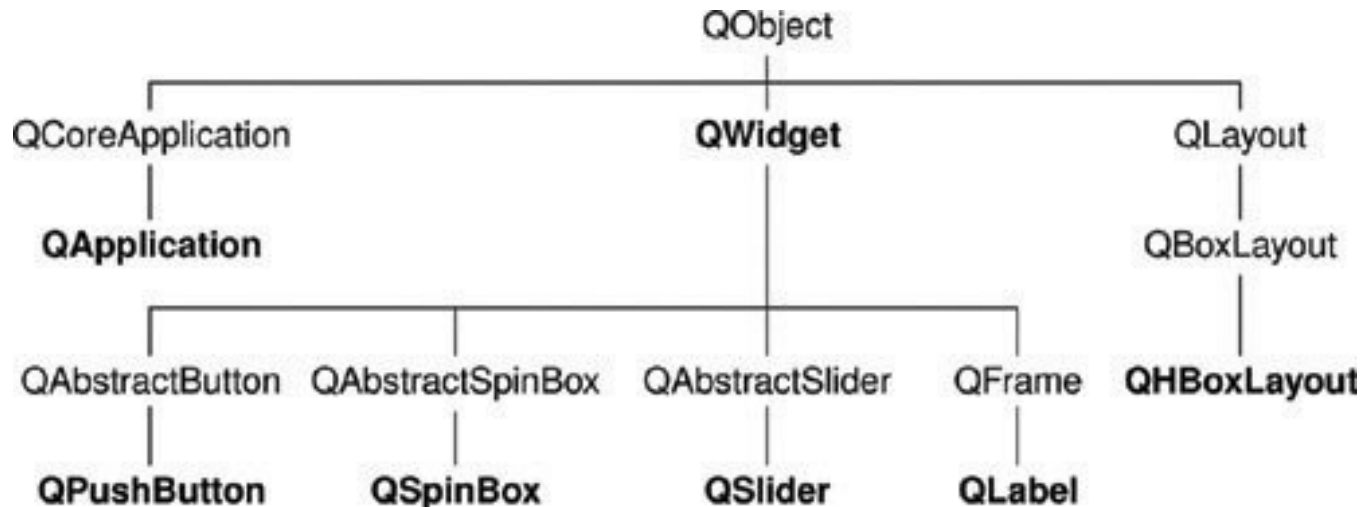


## # Compilation:

```
qmake -project
qmake
make
```

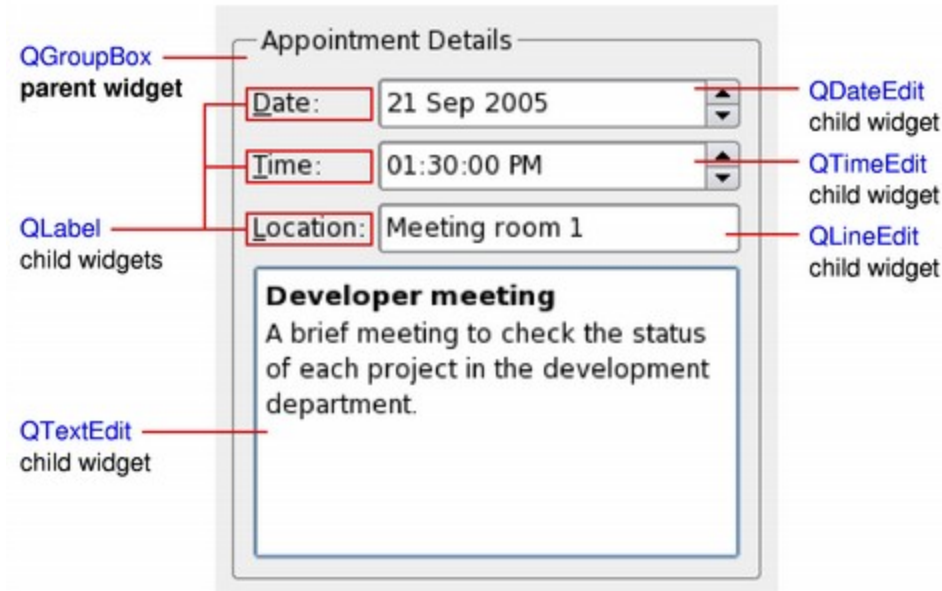
# Widget

- # Visible element of the user interface
- # Shorthand for *window gadget*
- # Buttons, menus, frames, sliders are all widgets
- # A widget can contain other widgets



# Parent and child widgets

- # A widget without the parent widget is always a separate window
- # The remaining widgets are displayed inside the parent widget
- # The parent widget frees the memory allocated for widgets inserted into it

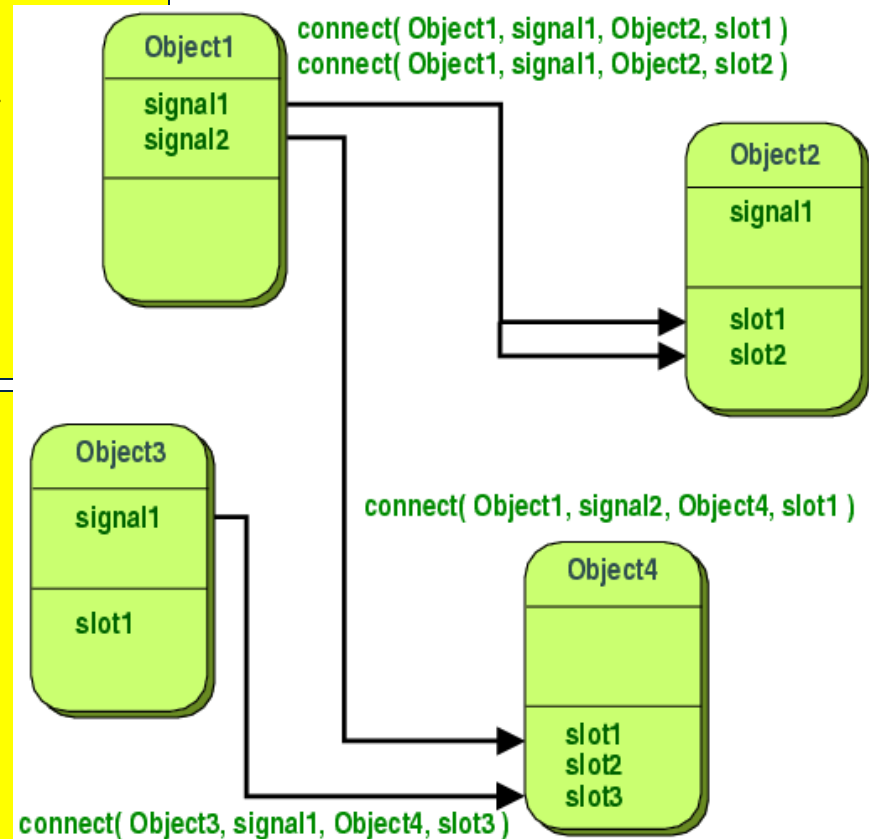


# Signals and Slots

```
#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;
};
```

```
#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;
}
```



# Internationalization

## # Mark strings for translation:

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

## # Install the translator in the application:

```
#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);
}
```

## # Modify the project file: TRANSLATIONS=app\_pl.ts

## # Extract the strings: lupdate

## # Edit the strings: linguist

## # Compile the database: lrelease