Lecture Material

QT graphical user interface framework

QT

Multiplatform GUI (Graphical User Interface) library

Available at https://www.qt.io/

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();
}
```
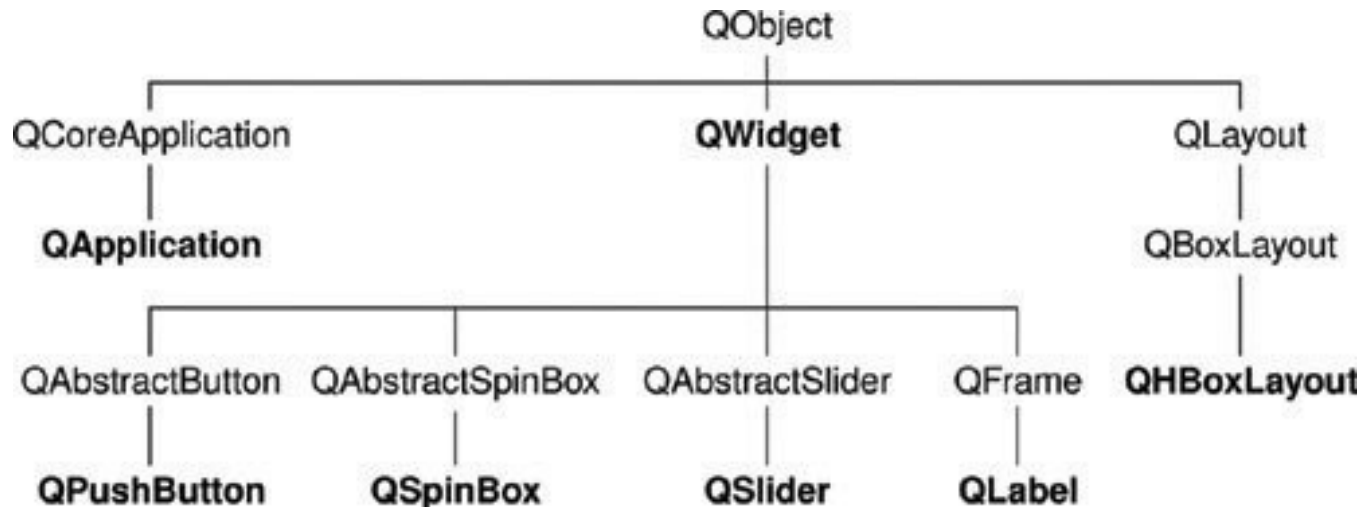
Compilation:

```
qmake -project
qmake QT+=widgets
make
```
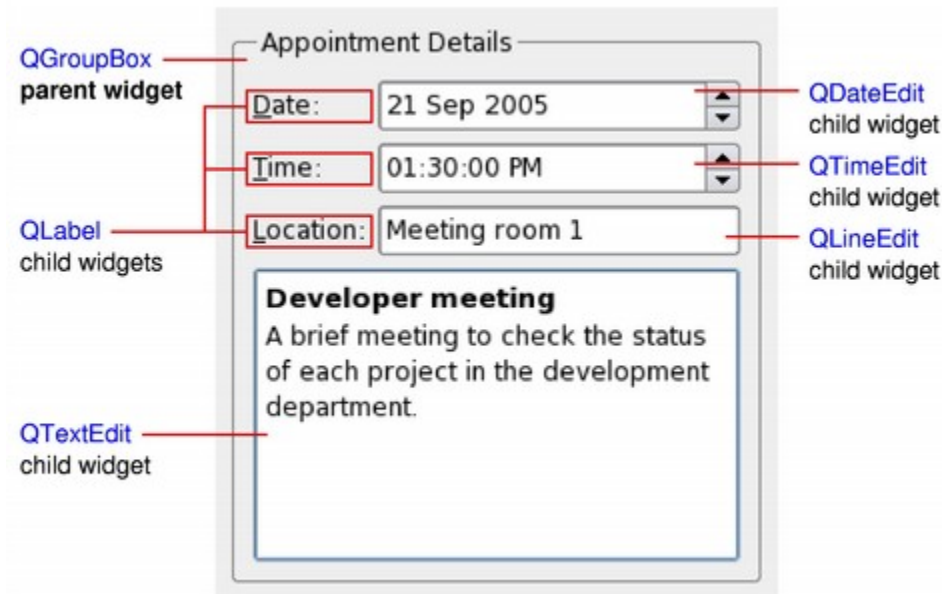
Widget

- Visible user interface component
- Shorthand for *window gadget*
- Buttons, menus, frames, sliders are all widgets
- Widget can contain other widgets

```
                              QObject
                                 |
        ┌────────────────────────┼────────────────────────┐
  QCoreApplication            QWidget                   QLayout
        |                        |                         |
   QApplication                  |                     QBoxLayout
                                 |                         |
        ┌─────────────┬──────────┴──────┬──────────┐       |
  QAbstractButton  QAbstractSpinBox  QAbstractSlider  QFrame  QHBoxLayout
        |                |               |            |
   QPushButton       QSpinBox        QSlider       QLabel
```

Parent and Child Widgets

- Widget without a parent is always a separate window
- The remaining widgets are displayed inside the parent widget
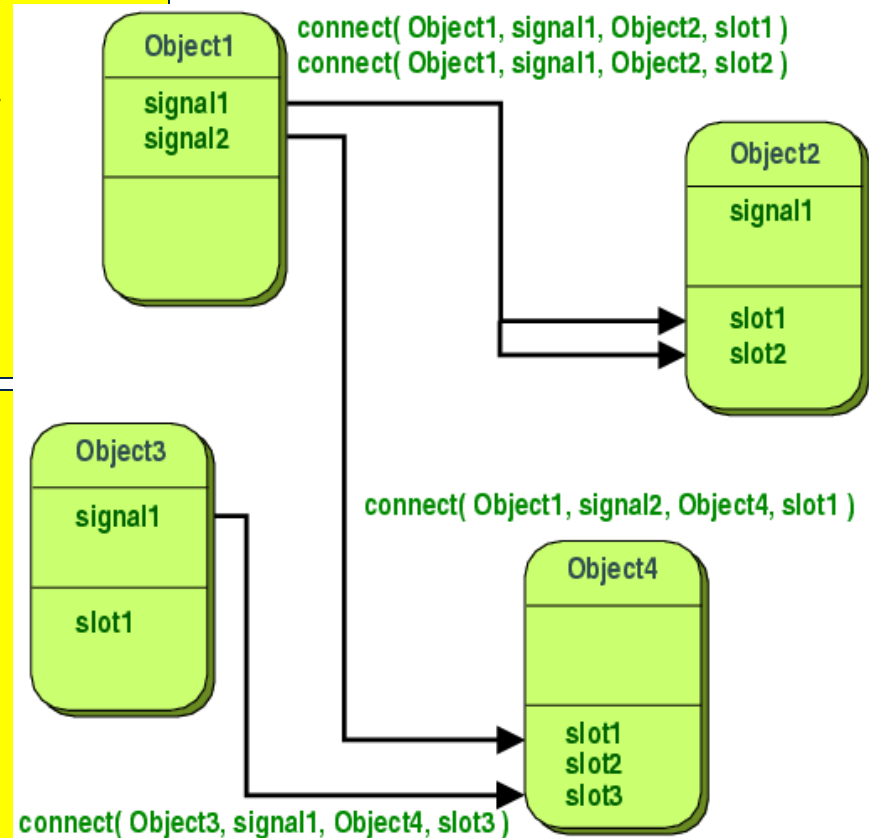- The parent widget will free the resources allocated by its child widgets

QGroupBox parent widget — Appointment Details

QLabel child widgets — Date:, Time:, Location:

QDateEdit child widget — 21 Sep 2005

QTimeEdit child widget — 01:30:00 PM

QLineEdit child widget — Meeting room 1

QTextEdit child widget — Developer meeting. A brief meeting to check the status of each project in the development department.

Signals and Slots

```cpp
#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;
};
```

```cpp
#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;
}
```


Object1
signal1
signal2

connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)

Object2
signal1
slot1
slot2

Object3
signal1
slot1

connect(Object1, signal2, Object4, slot1)

Object4
slot1
slot2
slot3

connect(Object3, signal1, Object4, slot3)

6

Internationalization

⌗ Mark texts for translation:

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

⌗ Install the translator in the program:

```
#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 strings: lupdate

⌗ Provide translations: linguist

⌗ Compile the database: lrelease