Основы разработки на Qt
(день первый)

4th Nokia week, 7th FRUCT
April 2010, S.Petersburg

приветствие

День первый

- Что такое Qt, краткая история, архитектура.
- Обзор основных пакетов и инструментов;
- Алгоритм создания приложения, система сборки qmake;
- Архитектура метаобъектной системы Qt;
- Сигналы и слоты;
- Способы обмена сообщениями между объектами;
- Обзор стандартных диалоговых окон, Базовые классы QWidget и QDialog;
- Менеджеры компоновки;
- Соглашения об выделении и использовании памяти.

Источники информации

- Qt Reference Documentation
- A. Ezust, P. Ezust Introduction to Design Patterns in C++ with Qt 4
- J. Blanchette, M. Summerfield C++ GUI Programming with Qt 4, Second Edition
- Creating Qt Applications for Maemo / Forum Nokia
- T. Torp Essentials of Qt programming / 6th FRUCT
- A. Jakl Qt for Symbian / Forum Nokia

История Qt

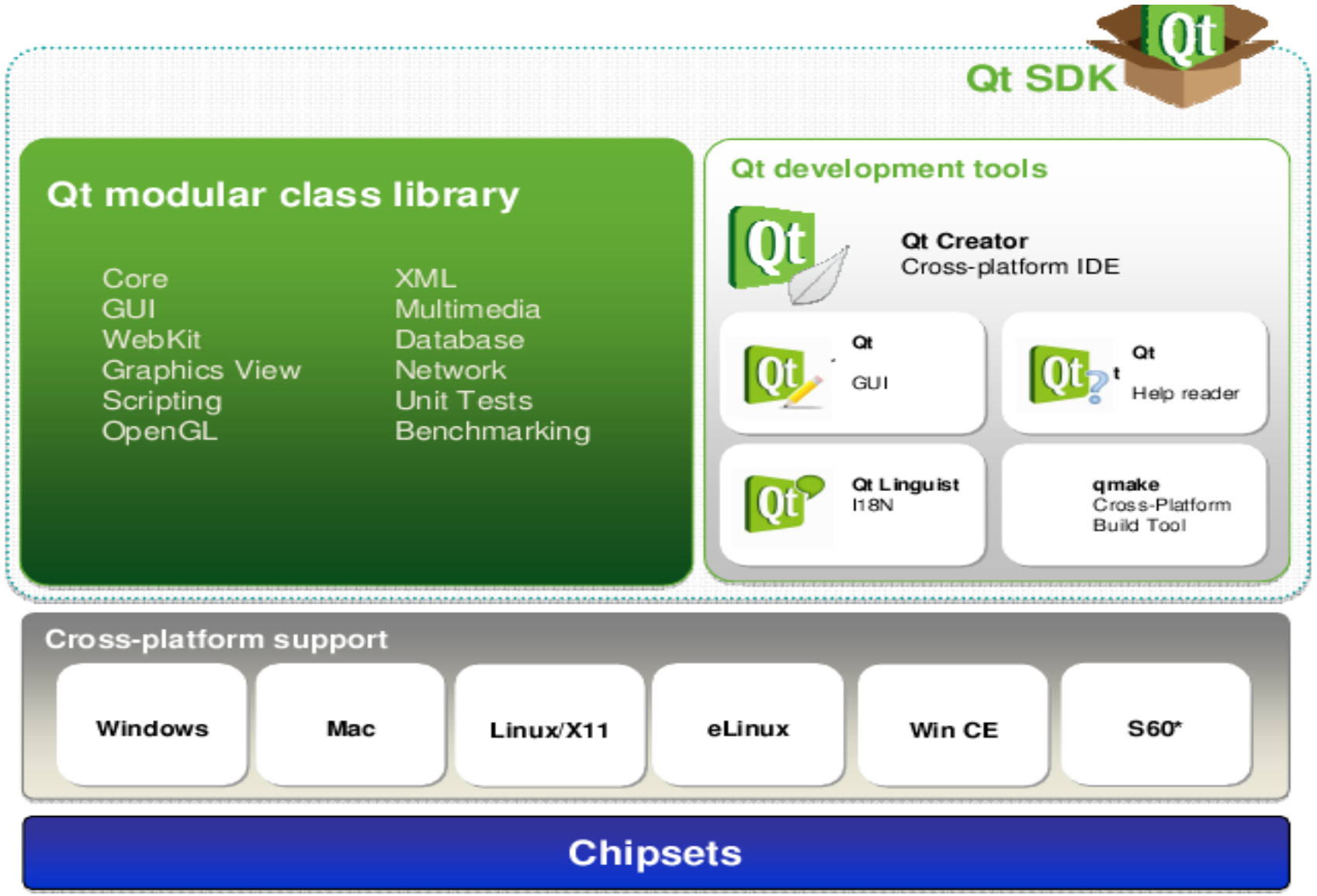
- **1991:** Development started
  - Cross platform GUI toolkit was needed
- **1994:** Decided to **go into business**
  - "Q" looked beautiful in Emacs font. "t" for toolkit
  - **Company name:** Quasar Technologies (later: Trolltech, now: Qt Software / Nokia)
- **1995: First public release** through newsgroups (Qt 0.90)
  - **Dual licensing:** commercial & free for open source
- **2000: Qtopia** – platform for mobile phones & PDAs
- **2005: Qt 4.0** – compatibility break, leads to new KDE 4 desktop
- **2008:** Nokia purchases Qt, name changes first to Qt Software, then to Qt Development Frameworks
- **2009 / 2010:** Focus on mobile platforms (Symbian and Maemo), dedicated mobility APIs

Eirik Chambe-Eng Nord

Haavard

Что такое Qt?

Компоненты библиотеки



Основные отличительные черты

- Библиотека C++ классов
- Переносимость между desktop/embedded системами
- Look and Feel оригинальных OS
- Кроссплатформенные средства разработки
- Высокая производительность на встраиваемых системах

Поддерживаемые платформы

- Windows
- Mac OS X
- Linux/X11 (KDE)
- Windows CE
- Embedded Linux
- Symbian S60

Языки программирования

- Python
- Ada
- Pascal
- Perl
- PHP
- Ruby
- Java (Qt Jambi)

Модули Qt

QtCore	Core non-graphical classes used by other modules
QtGui	Graphical user interface (GUI) components
QtNetwork	Classes for network programming
QtOpenGL	OpenGL support classes
QtScript	Classes for evaluating Qt Scripts
QtScriptTools	Additional Qt Script components
QtSql	Classes for database integration using SQL
QtSvg	Classes for displaying the contents of SVG files
QtWebKit	Classes for displaying and editing Web content
QtXml	Classes for handling XML
QtXmlPatterns	An XQuery & XPath engine for XML and custom data models
Phonon	Multimedia framework classes
Qt3Support	Qt 3 compatibility classes

Модули Qt - (прод)

- Инструменты

QtDesigner	Classes for extending *Qt Designer*
QtUiTools	Classes for handling *Qt Designer* forms in applications
QtHelp	Classes for online help
QtAssistant	Support for online help
QtTest	Tool classes for unit testing

- Поддержка ActiveX

QAxContainer	Extension for accessing ActiveX controls
QAxServer	Extension for writing ActiveX servers

- Поддержка Dbus

QtDBus	Classes for Inter-Process Communication using the D-Bus

Qt vs GTK

- Both Qt and GTK+ were developed from the ground up with Object Oriented Programming in mind
- Qt is said to be clearer and better documented
- GTK uses less memory
- GTK uses standard C++
- Qt extends C++ (MOC needed)
- Qt has a better cross-platform support
- Qt is a complete framework (networking, threading, I/O)
- GTK interoperates well with other librares like GLib, GNet..
- http://www.wikivs.com/wiki/Qt_vs_GTK

Qt vs Java

- Java
  - Написано один раз – исполняется везде
- Qt
  - Написано один раз – компилируется везде

Инструменты.
Алгоритмы создания приложений.

Что нужно знать...

- Основы C++
- Объекты и классы
  - Объявление, наследование, вызов методов
- Полиморфизм, виртуальные функции
- Перегрузка операторов
- Шаблоны только для контейнеров
- Не использовать RTTI

Инструменты

- Qmake
- Moc
- Linguist
- Designer
- Assistant
- IDE integration: Visual Studio,Eclipse

Методы создания приложения

- Qt Creator
- Qmake +QtDesigner
- Вручную (Qmake+ текстовый редактор)

Hello world!

Объявления Классов Qt

```
#include <QApplication>
#include <QVBoxLayout>
#include <QDialog>
#include <QLabel>
```

```
int main(int c, char **v)
{
  QApplication a(c,v);
  QDialog d(NULL);

  d.setLayout(new QVBoxLayout());
  d.layout()->addWidget(new QLabel("Hello world!", &d));

  d.show();
  return a.exec();
}
```

Создание компоновки

Исполнение диалога и приложения

Алгоритм создания приложения

- $qmake -project
  - создание файла .pro
- $qmake
  - подключает специфичные для платформы заголовочные файлы и библиотеки
  - вызывает компилятор метаобъектной системы
  - создает makefile
- $make
  - Компилиует проект стандартными средствами gmake/gcc

Qmake: управление проектом

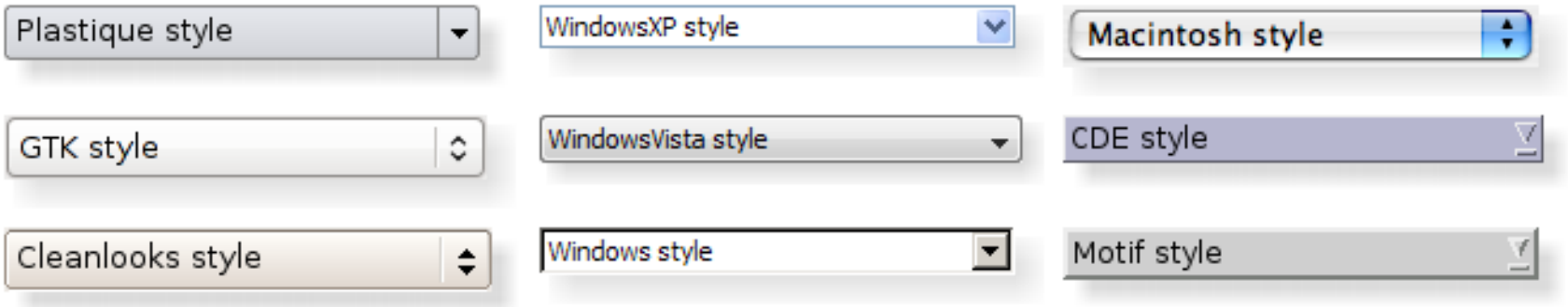
- Исходные файлы
  - hello.cpp
  - hello.h
  - main.cpp
- Создание проекта (*.pro)
- Добавление исходных текстов
  - SOURCES = hello.cpp main.cpp
  - HEADERS = hello.h

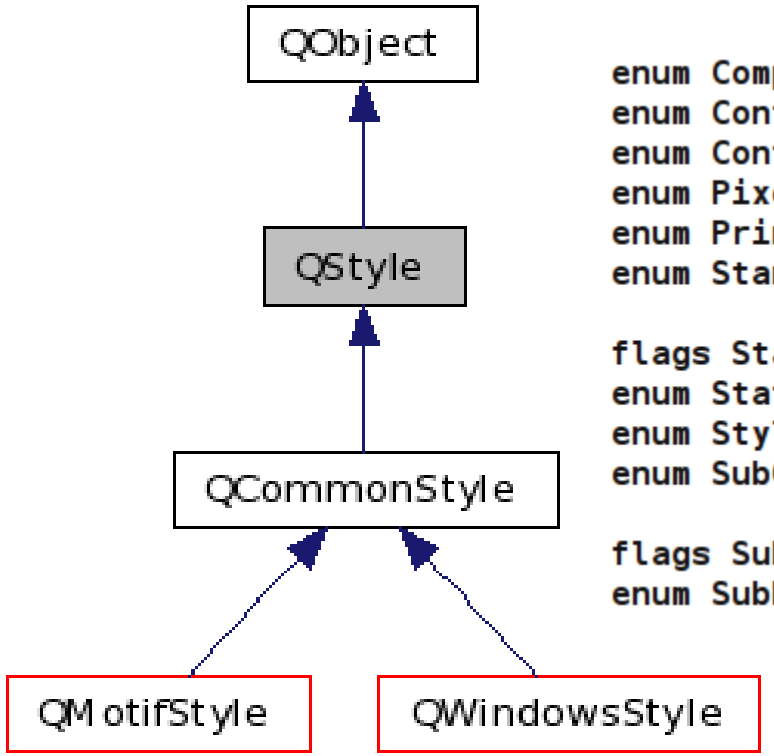
Qmake: управление проектом (пр.)

- TARGET = helloworld
- CONFIG += qt

- Компиляцияи сборка
  - $qmake [-o Makefile hello.pro]
  - $make all
- Платформо-зависимые файлы
    - win32 {
    - SOURCES += hellowin.cpp
    - }
- Проверка существования
  - !exists(main.cpp) {
  - error("No main.cpp file found")
  - }

Внешний вид приложений






```
enum ComplexControl    { CC_SpinBox, ...}
enum ContentsType      { CT_CheckBox, ... }
enum ControlElement    { CE_PushButton, ... }
enum PixelMetric       { PM_ButtonMargin, ... }
enum PrimitiveElement  { PE_FrameStatusBar, ... }
enum StandardPixmap    { SP_TitleBarMinButton, ... }

flags State
enum StateFlag         { State_None, State_Active, ... }
enum StyleHint         { SH_EtchDisabledText, ... }
enum SubControl        { SC_None, SC_ScrollBarAddLine, ... }

flags SubControls
enum SubElement        { SE_PushButtonContents, ... }
```

Внешний вид приложений

```
#include <QApplication>
#include <QVBoxLayout>
#include <QDialog>
#include <QLabel>
#include <QPushButton>

int main(int c, char **v)
{
  QApplication a(c,v);
  QDialog d(NULL);

  d.setLayout(new QVBoxLayout());
  d.layout()->addWidget(new QLabel("Hello world!", &d));
  d.layout()->addWidget(new QPushButton("Press me",&d));

  d.show();
  a.exec();
}
```

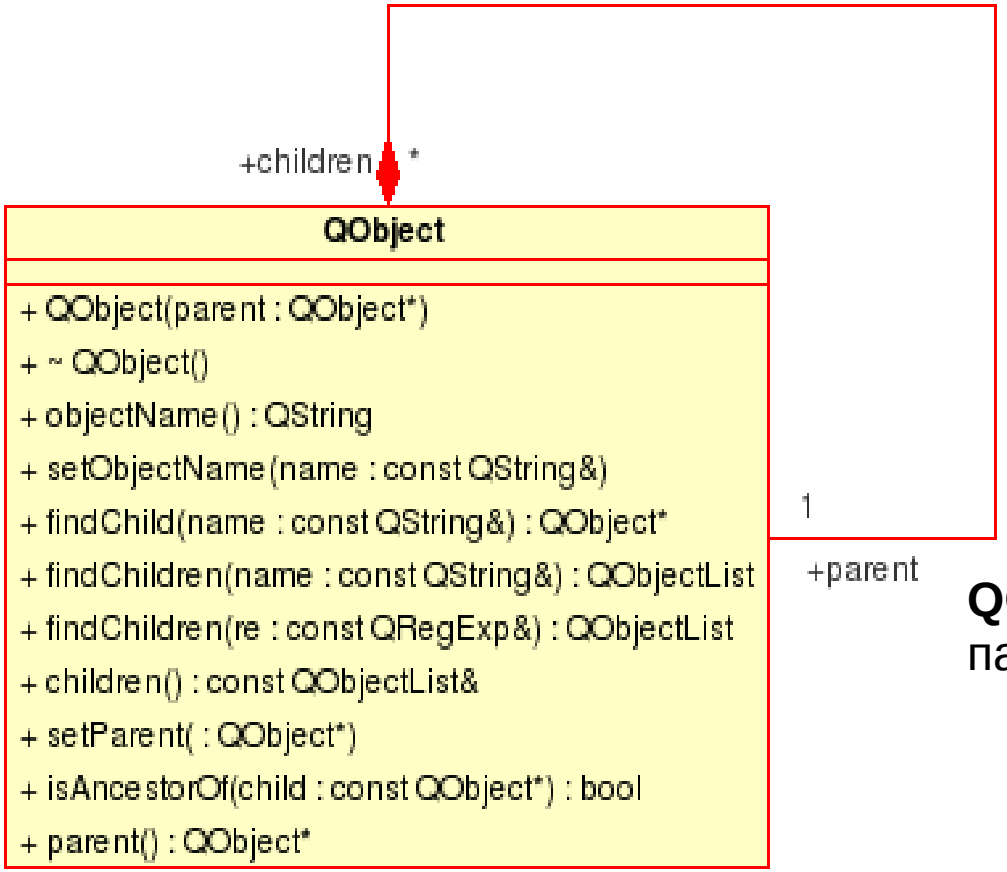
$./hello

$./hello -style windows

$./hello -style motif

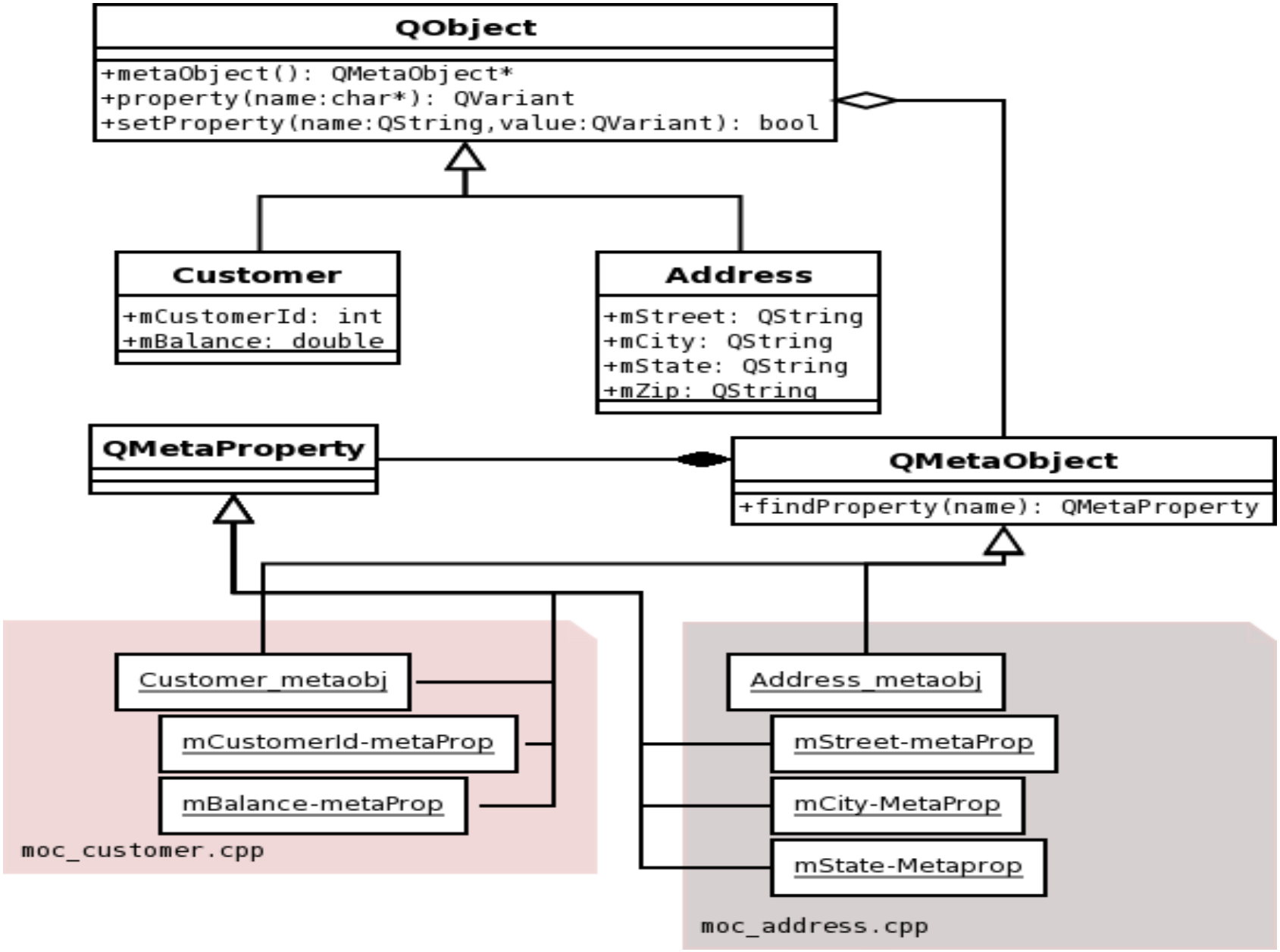
Метаобъектная система

Класс QObject



QObject реализует архитектурный паттерн “Композиция”

Метаобъектная система Qt



Информация об объектах

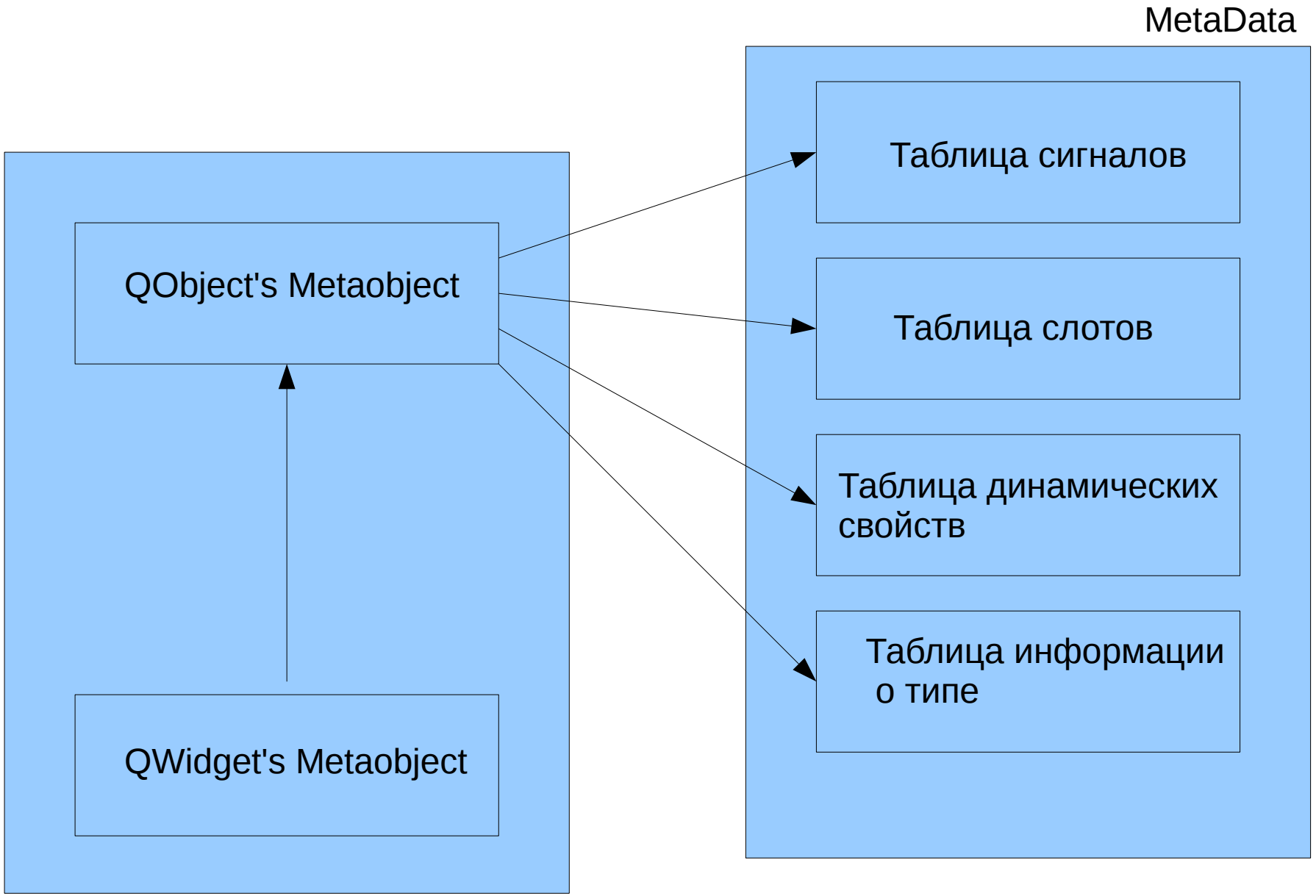
- C++ стиль

```
if (widget->inherits("QAbstractButton"))
{
    QAbstractButton *button = static_cast<QAbstractButton *>(widget);
    button->toggle();
}
```

- Qt стиль

```
if (QAbstractButton *button = qobject_cast<QAbstractButton *>(widget))
{
        button->toggle();
}
```

QMetaObject



Meta-object compiler

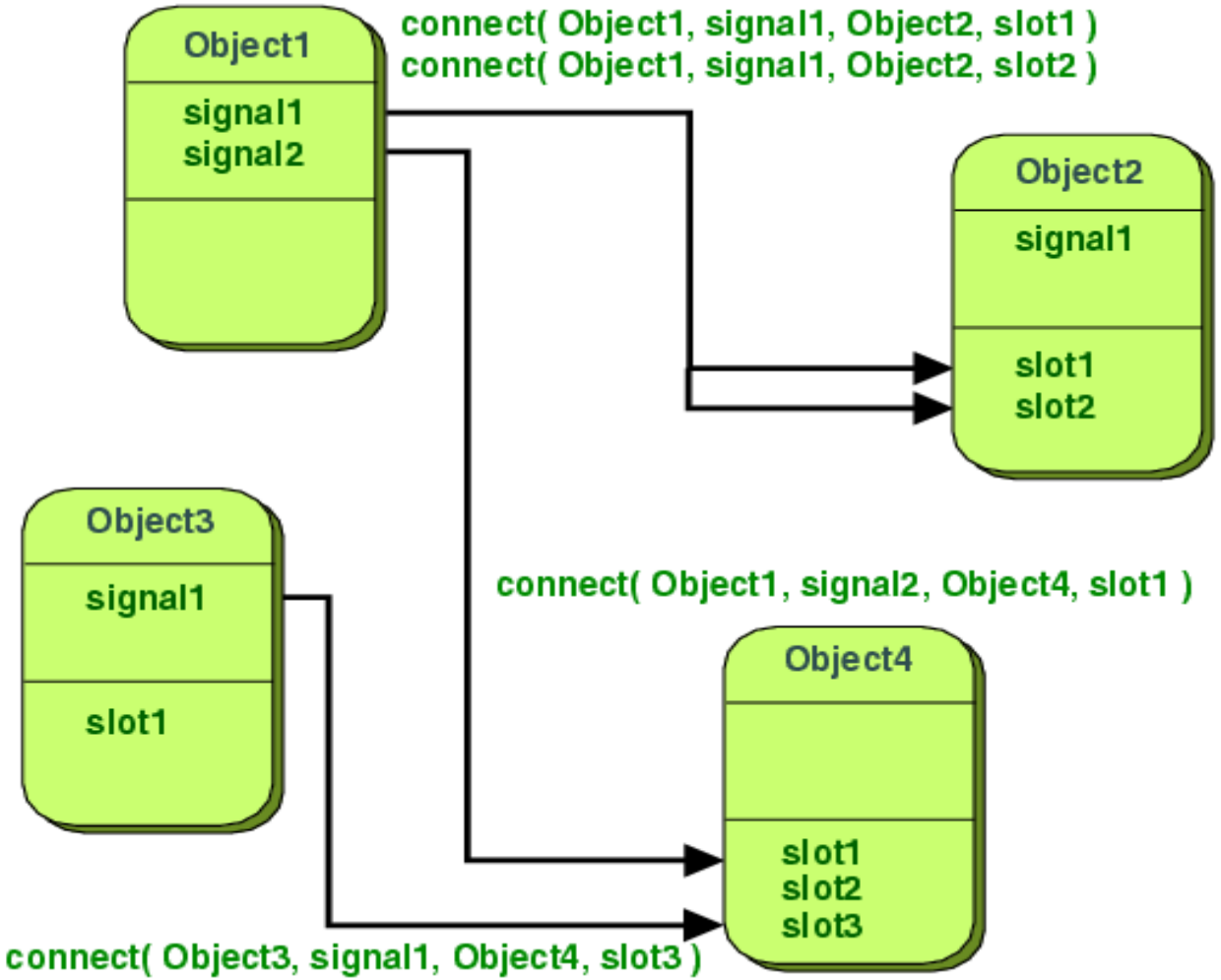
- Qt не стандартный C++
- Обрабатывает расширения C++
- Ищет определения Q_OBJECT и другие; генерирует C++ код
- Требуется для:
  - Сигналов и слотов
  - Получения run-time информации
  - Поддержки динамических свойств

moc: контрольный список

- Объявление ***каждого*** класса помещено в отдельный .h файл
- Имплементация ***каждого*** класса помещена в соответсвующий .cpp файл
- ***Каждый*** заголовочный файл защищен #ifndef
- ***Каждый*** .cpp файл перечислен в SOURCES
- ***Каждый*** .h файл перечислен в HEADERS
- Q_OBJECT макро присутствует в определении ***каждого*** класса

Сигналы и слоты.
Обмен сообщениями между объектами

Сигналы и слоты кратко



Основные примитивы

- QObject::connect (

  sender, SIGNAL(signal),

  receiver, SLOT(slot));

QObject::disconnect (

  sender, SIGNAL(signal),

  receiver, SLOT(slot));

- emit (signal(parameters));
- Макросы (CONFIG += no_keywords):
  - Q_SIGNAL, Q_SLOT, Q_EMIT;

Пример соединения

```
 1 #include <QApplication>
 2 #include <QPushButton>

 3 int main(int argc, char *argv[])
 4 {
 5     QApplication app(argc, argv);
 6     QPushButton *button = new QPushButton("Quit");
 7     QObject::connect(button, SIGNAL(clicked()),
 8                      &app, SLOT(quit()));
 9     button->show();
10    return app.exec();
11 }
```

Соединяет сигнал кнопки button
со слотом quit()приложения

Сигналы и слоты (пример 1/2)

```
#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;
};
```

Сигналы и слоты (пример 2/2)

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

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

a.setValue(12);     // a.value() == 12, b.value() == 12
b.setValue(48);     // a.value() == 12, b.value() == 48
```

Сигналы и слоты (пример 2/3)

```
#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;
};
```

Сигналы и слоты (пример 3/3)

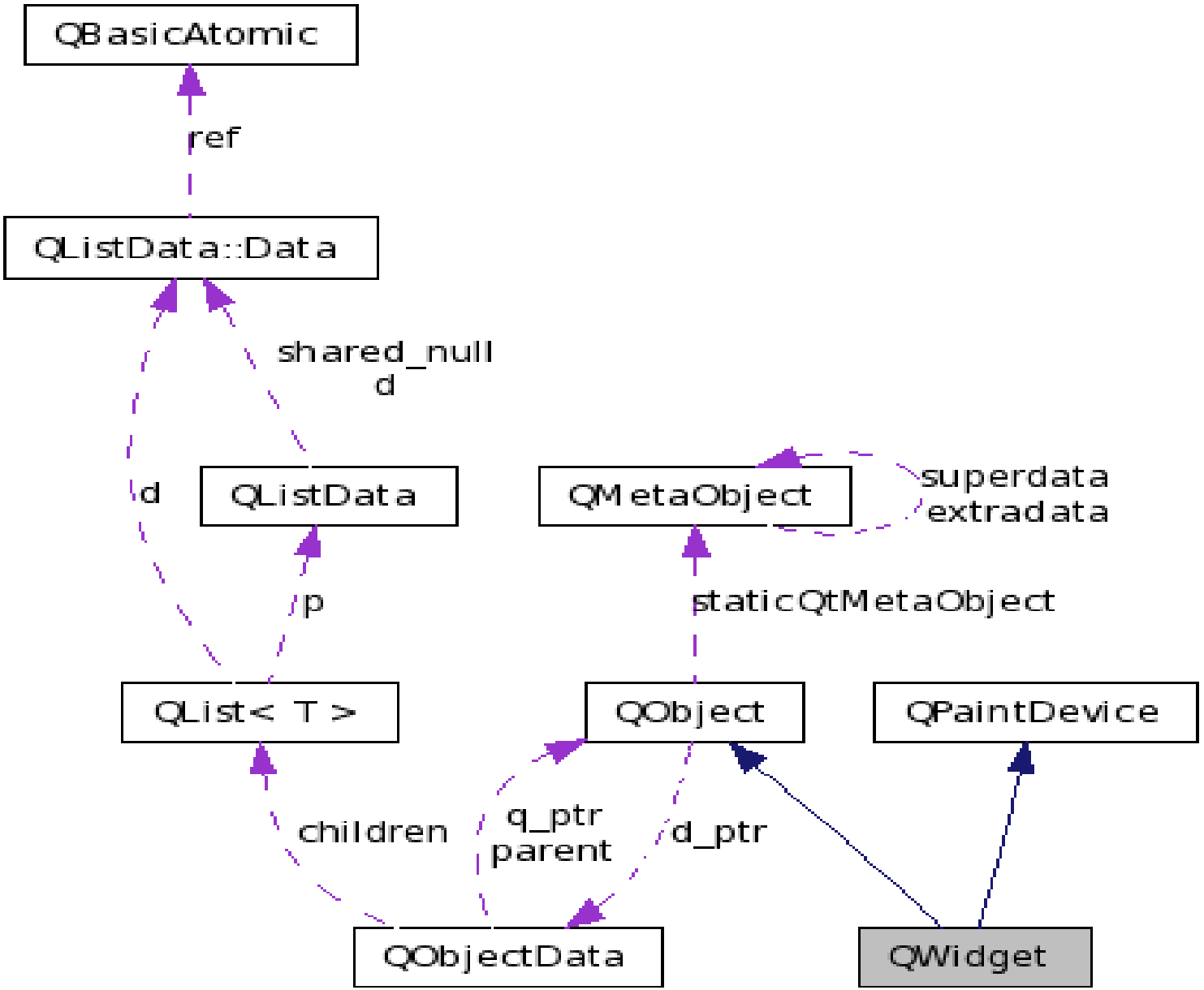
```
void Counter::setValue(int value)
{
    if (value != m_value)
    {
        m_value = value;
        emit valueChanged(value);
    }
}
```

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

a.setValue(12);     // a.value() == 12, b.value() == 12
b.setValue(48);     // a.value() == 12, b.value() == 48
```

QtGUI. Окна и диалоги.

Класс QWidget



Основные положения

- QWidget родитель всех классов GUI
- Конструктор любого виджета принимает:
  - QWidget *parent [= 0]
  - Qt::WindowFlags f [= 0]
- QWidget верхнего уровня ***должен*** иметь
  - parent = NULL
- QWidgetы потомки должны иметь
  - Parent != NULL

Класс QDialog

Public Functions

- **QDialog** (QWidget * *parent* = 0, Qt::WindowFlags *f* = 0)
- **~QDialog** ()
- bool **isSizeGripEnabled** () const
- int **result** () const
- void **setModal** (bool *modal*)
- void **setResult** (int *i*)
- void **setSizeGripEnabled** (bool)

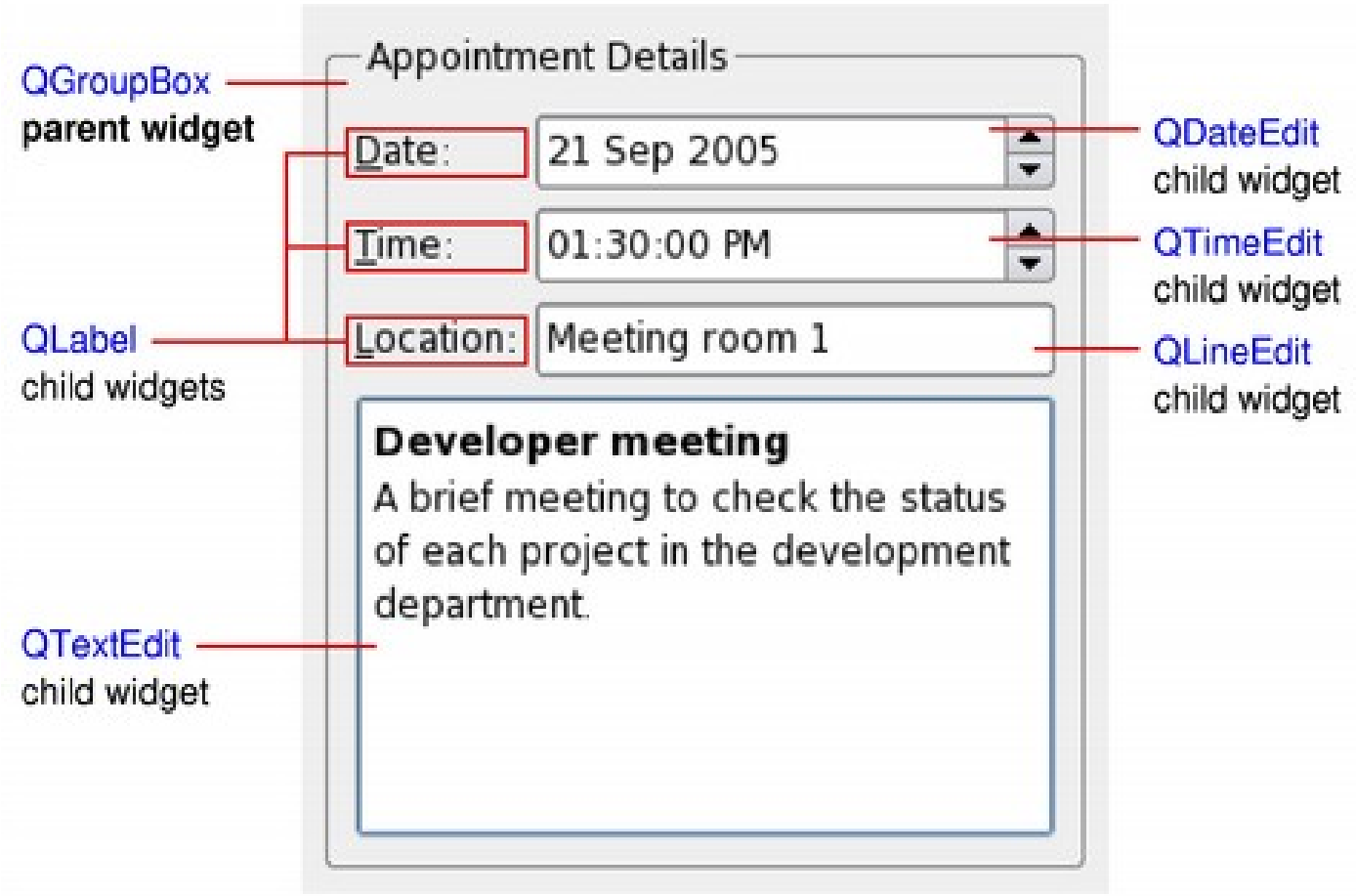
Public Slots

- virtual void **accept** ()
- virtual void **done** (int *r*)
- int **exec** ()
- void **open** ()
- virtual void **reject** ()

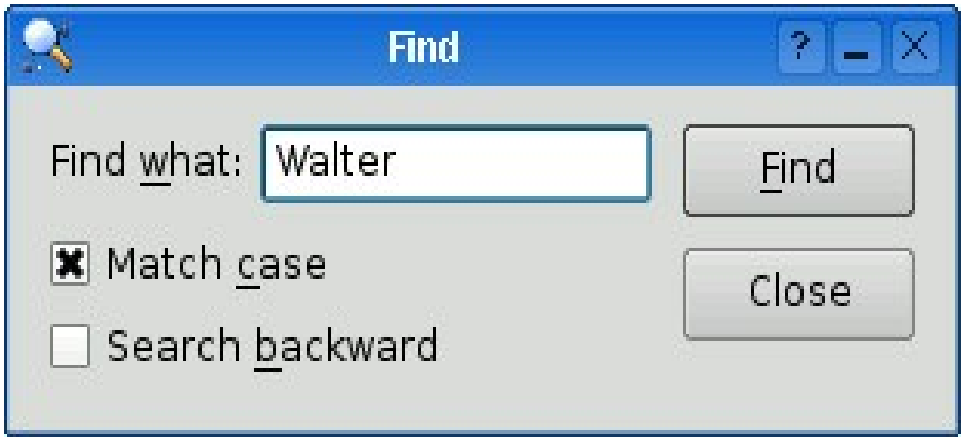
Signals

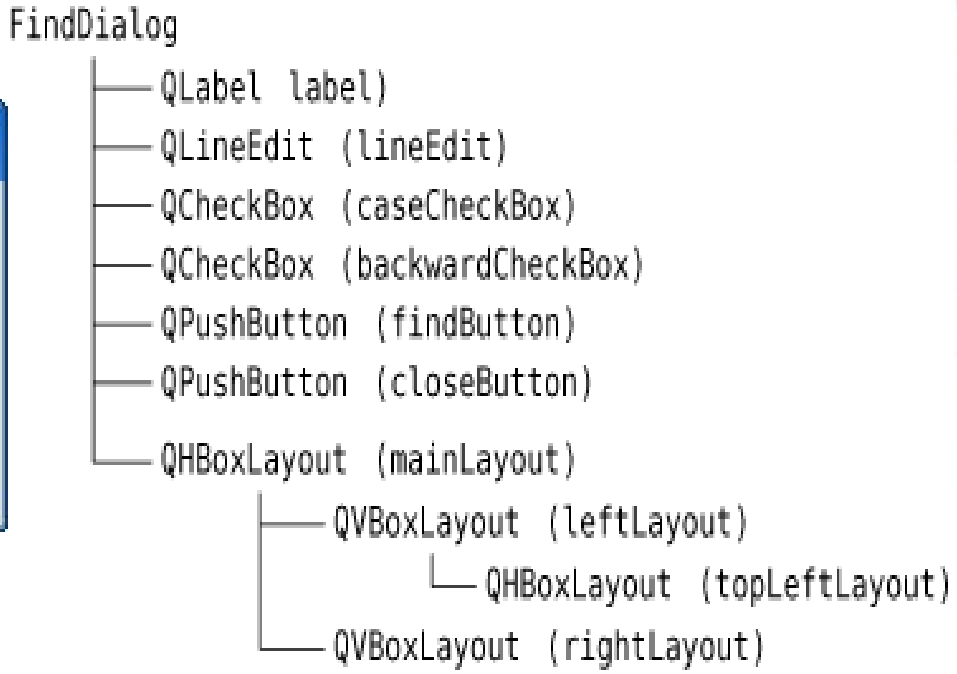
- void **accepted** ()
- void **finished** (int *result*)
- void **rejected** ()

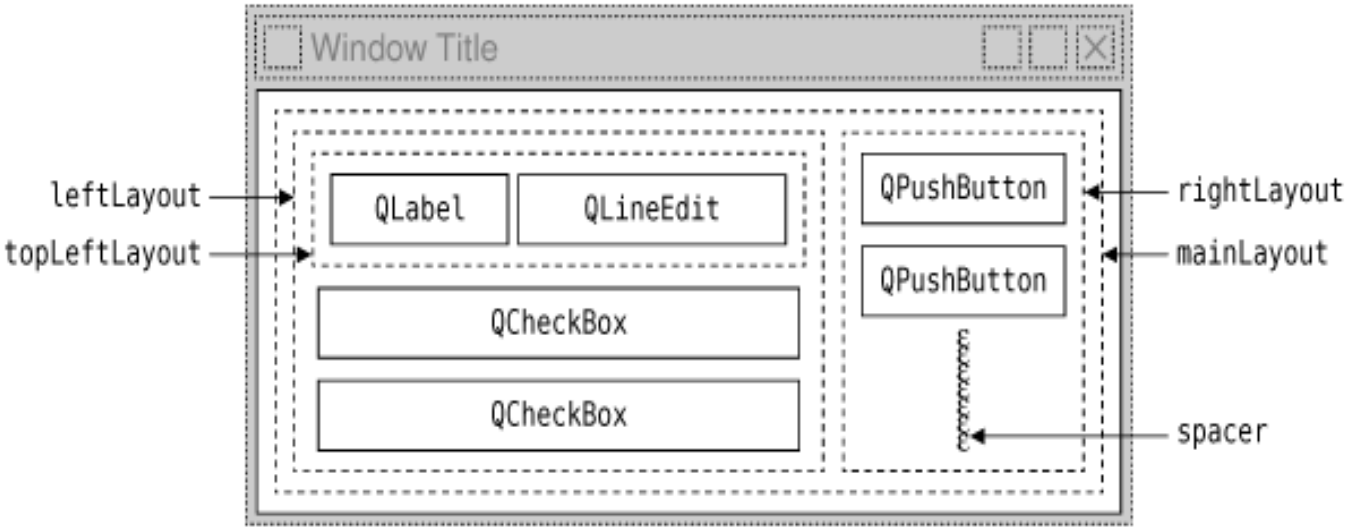
Отношение Parent-Child



Пример


Find
Find what:
Walter
Find
Match case
Close
Search backward


FindDialog
QLabel label)
QLineEdit (lineEdit)
QCheckBox (caseCheckBox)
QCheckBox (backwardCheckBox)
QPushButton (findButton)
QPushButton (closeButton)
QHBoxLayout (mainLayout)
QVBoxLayout (leftLayout)
QHBoxLayout (topLeftLayout)
QVBoxLayout (rightLayout)


Window Title
leftLayout
topLeftLayout
QLabel
QLineEdit
QCheckBox
QCheckBox
QPushButton
QPushButton
rightLayout
mainLayout
spacer

QDialog: использование

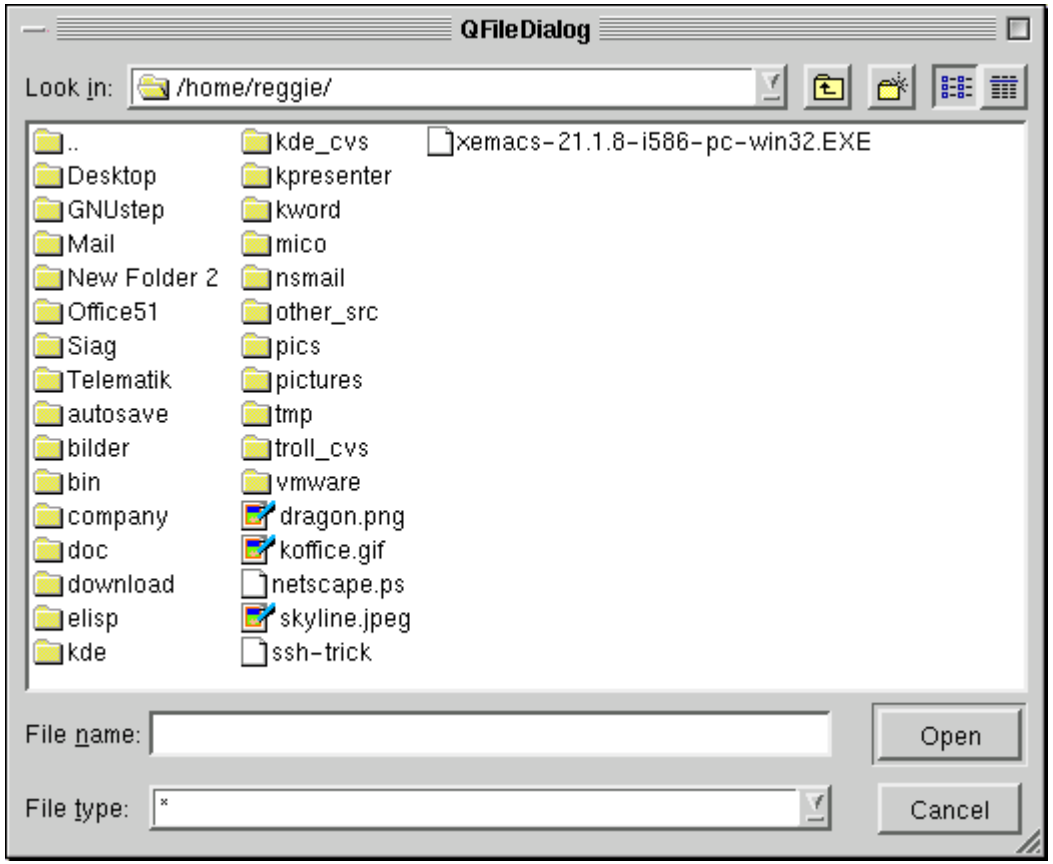
```
///Модальный диалог
 void EditorWindow::countWords()
 {
     WordCountDialog dialog(this);
     dialog.setWordCount(document().wordCount());
     dialog.exec();
 }

///Немодальный диалог
 void EditorWindow::find()
 {
     if (!findDialog) {
         findDialog = new FindDialog(this);
         connect(findDialog, SIGNAL(findNext()), this, SLOT(findNext()));
     }
     findDialog->show();
     findDialog->raise();
     findDialog->activateWindow();
 }
```

Стандартные диалоговые окна

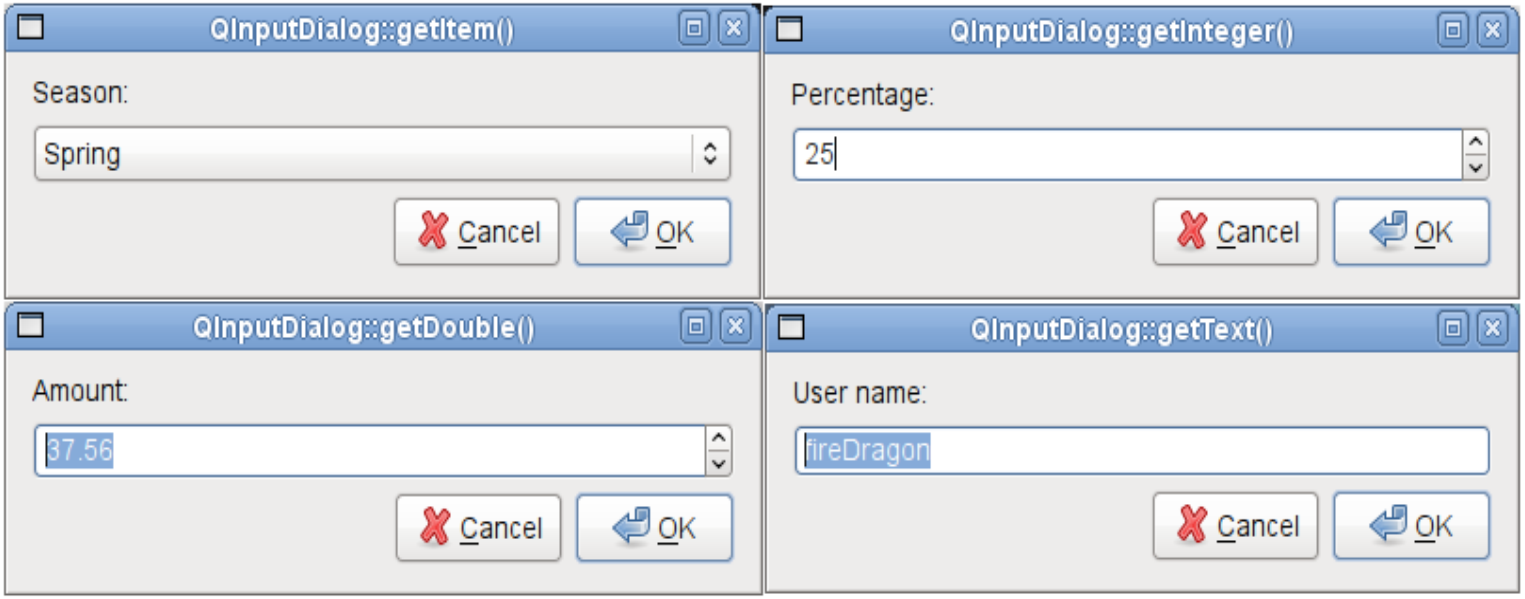
- QFileDialog
- QInputDialog
- QMessageBox
- QErrorMessage
- QColorDialog
- QFontDialog
- QPringDialog
- QProgressDialog

QFileDialog



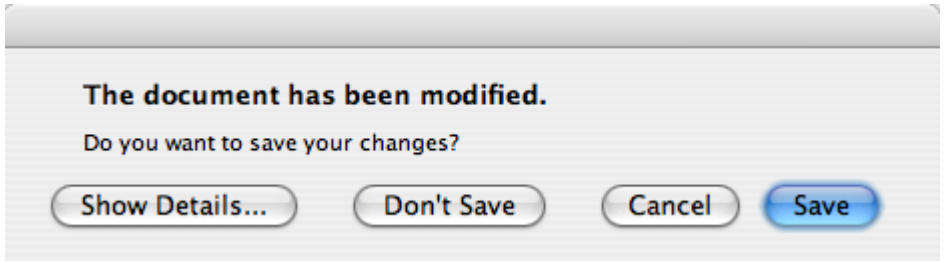
- acceptMode : AcceptMode
- defaultSuffix : QString
- fileMode : FileMode

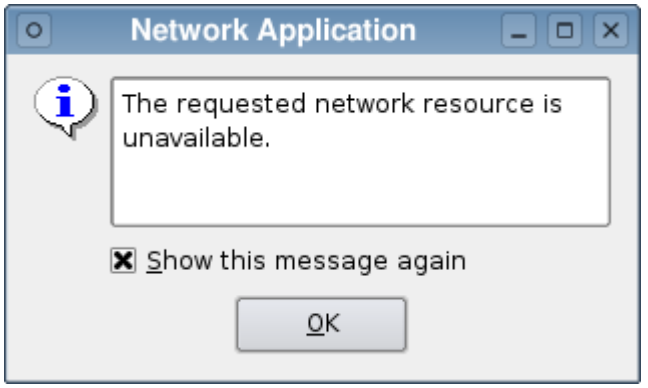
QInputDialog

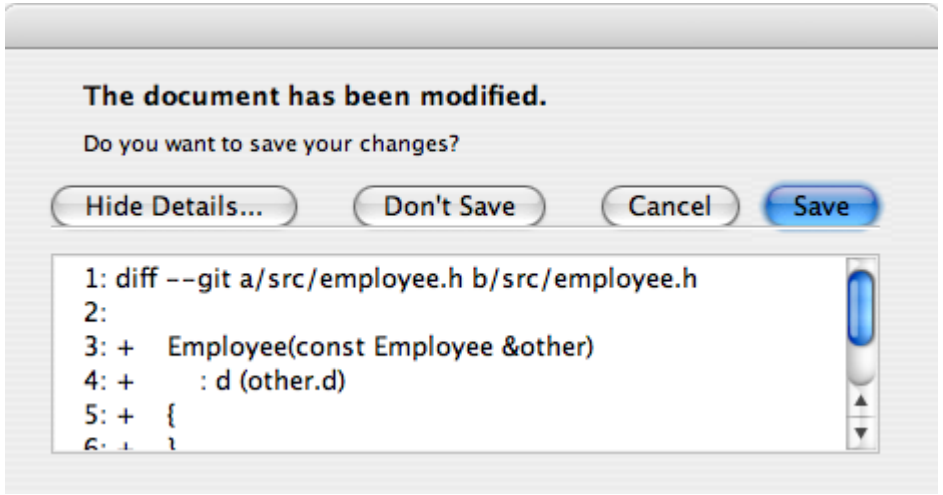


- **cancelButtonText** : QString
- **comboBoxEditable** : bool
- **comboBoxItems** : QStringList
- **doubleDecimals** : int
- **doubleMaximum** : double
- **doubleMinimum** : double
- **doubleValue** : int
- **inputMode** : InputMode
- **intMaximum** : int
- **intMinimum** : int
- **intStep** : int
- **intValue** : int
- **labelText** : QString
- **okButtonText** : QString
- **options** : InputDialogOptions
- **textEchoMode** : QLineEdit::EchoMode
- **textValue** : QString

QMessageBox, QErrorMessage

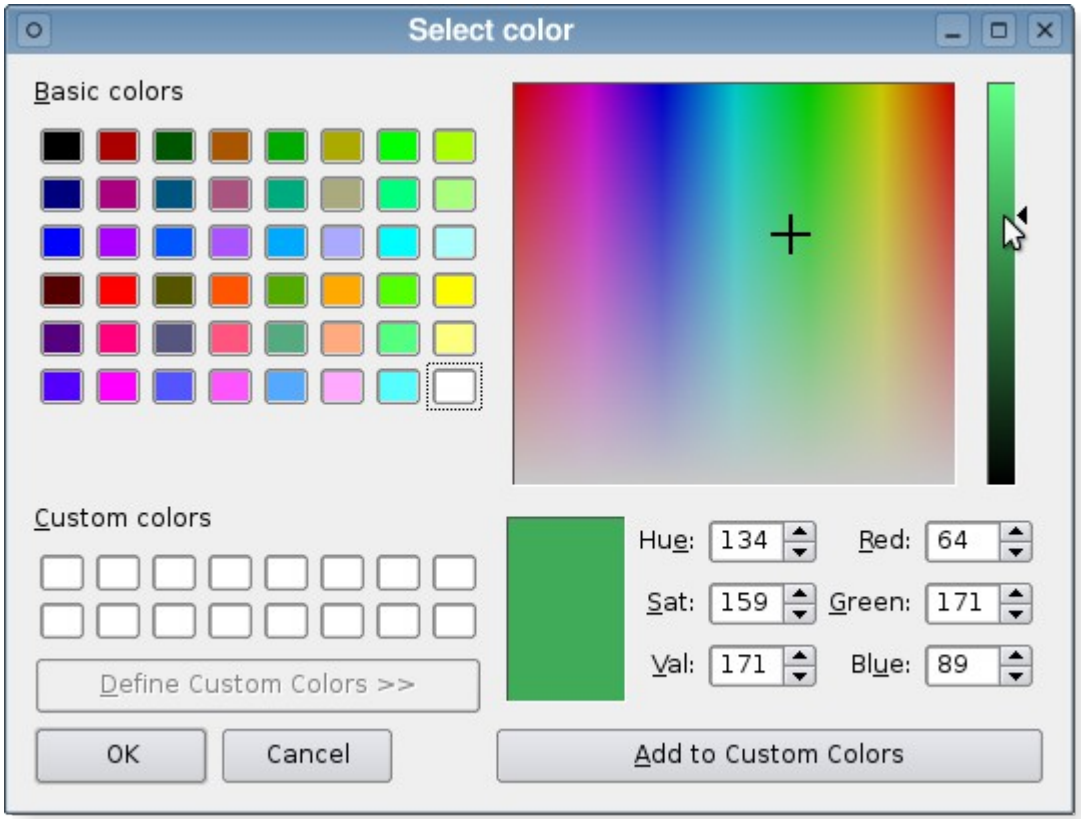






- **detailedText** : QString
- **icon** : Icon
- **iconPixmap** : QPixmap
- **informativeText** : QString
- **standardButtons** : StandardButtons
- **text** : QString
- **textFormat** : Qt::TextFormat

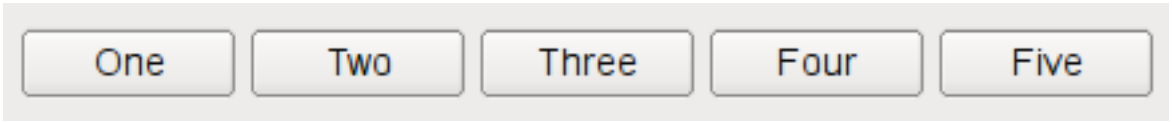
QColorDialog



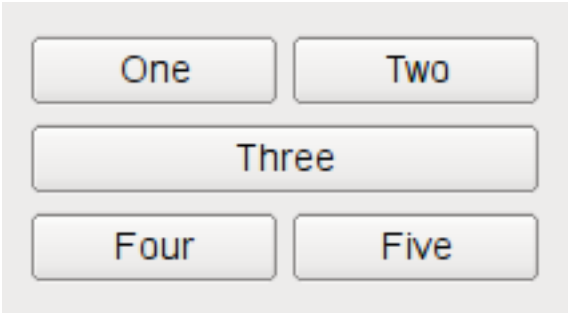
- **currentColor** : QColor

Менеджеры компоновки.

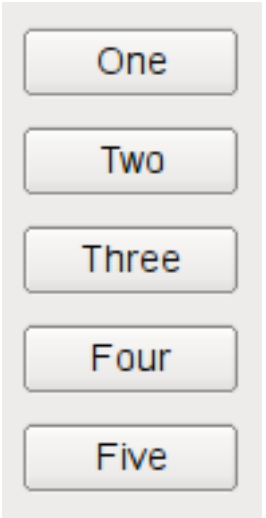
Layouts: менеджеры компоновки



QHBoxLayout



QGridLayout



QVBoxLayout

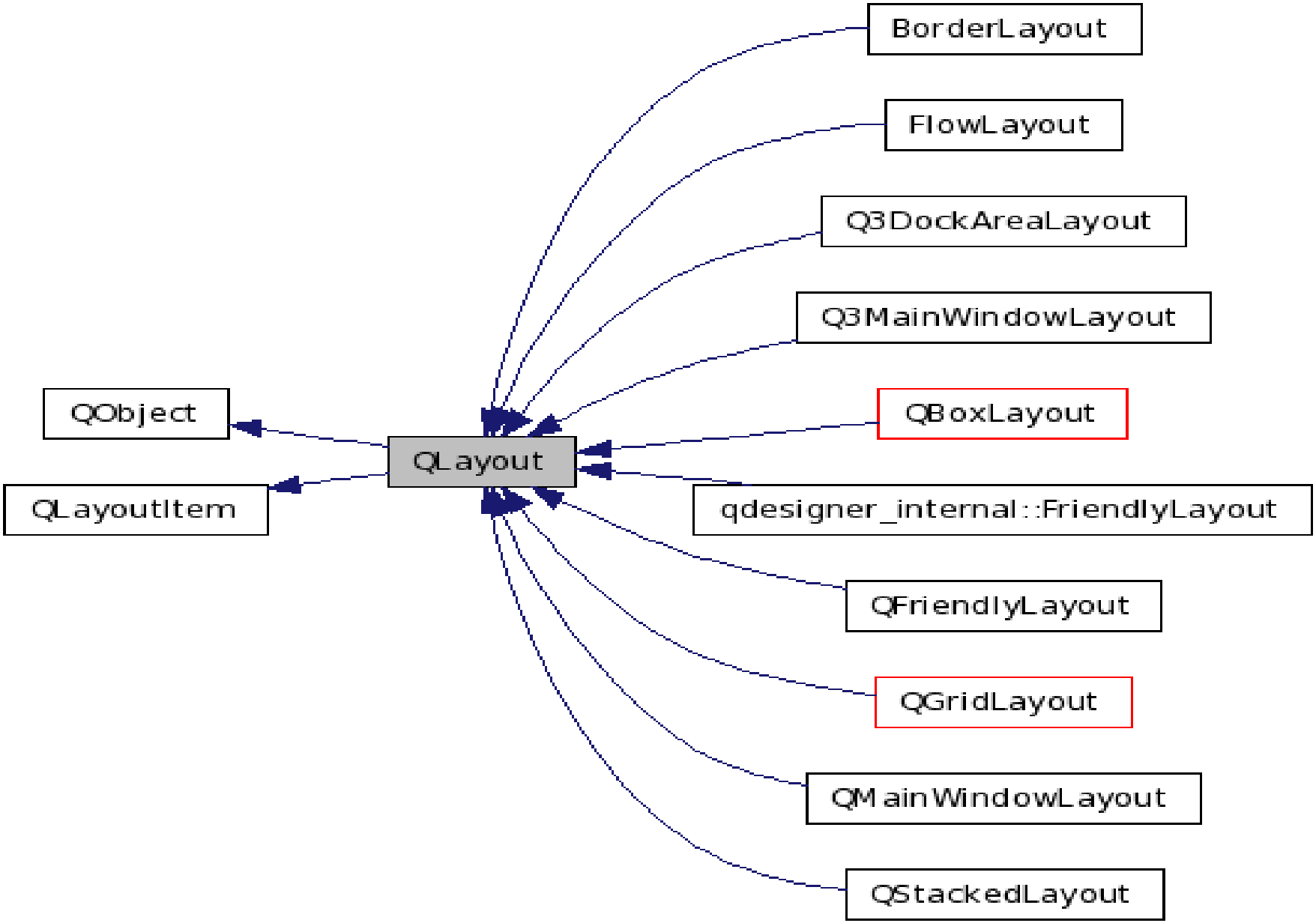
QFormLayout

QLayout

- virtual void addItem (QLayoutItem * item)
- void addWidget (QWidget * w)

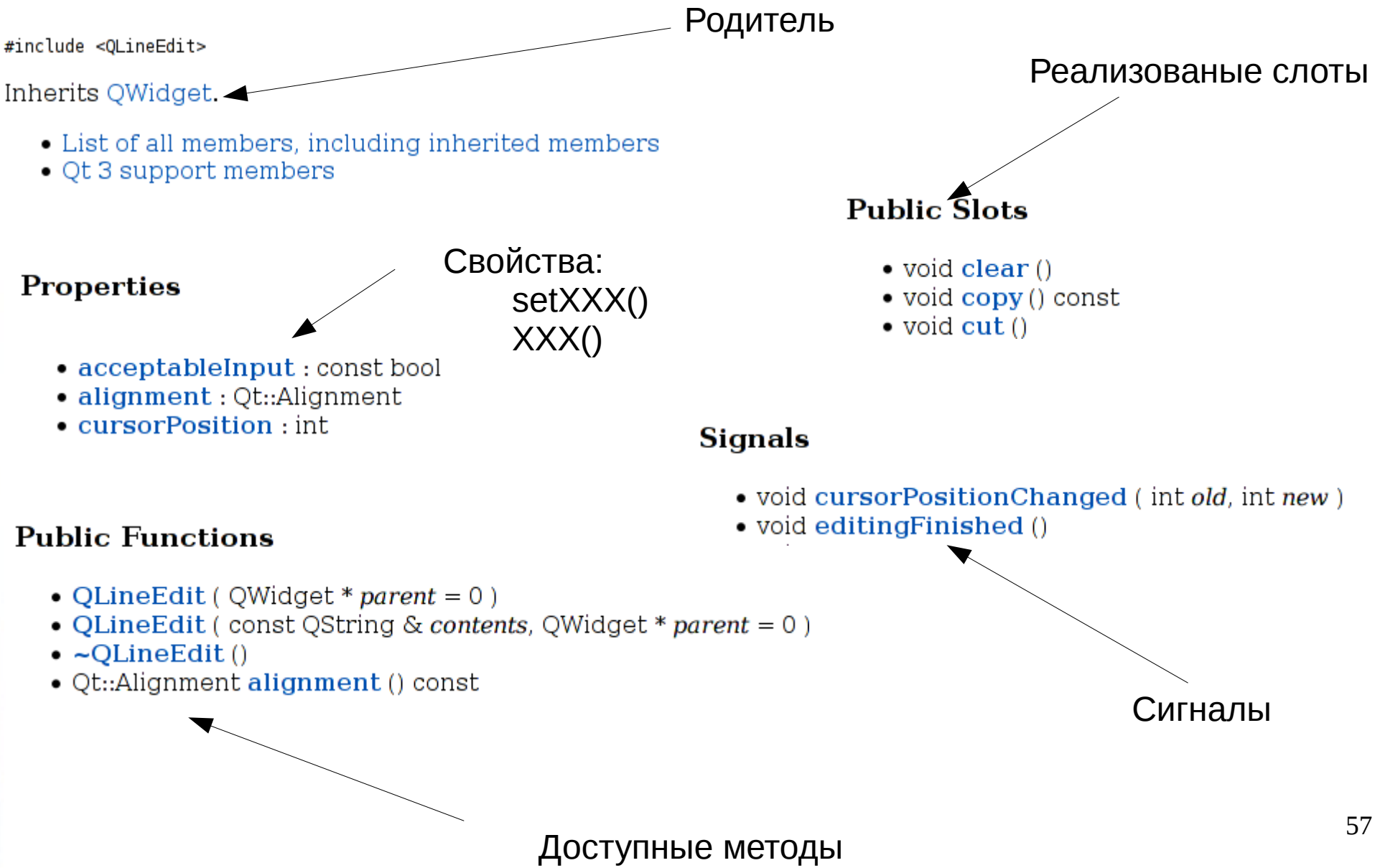
- void removeItem (QLayoutItem * item)
- void removeWidget (QWidget * widget)

QLayout (детали)



Проектирование GUI.

Как читать документацию



Спасибо...

Кирилл Кринкин
Open Source & Linux Lab, FRUCT
osll@fruct.org, http://osll.fruct.org