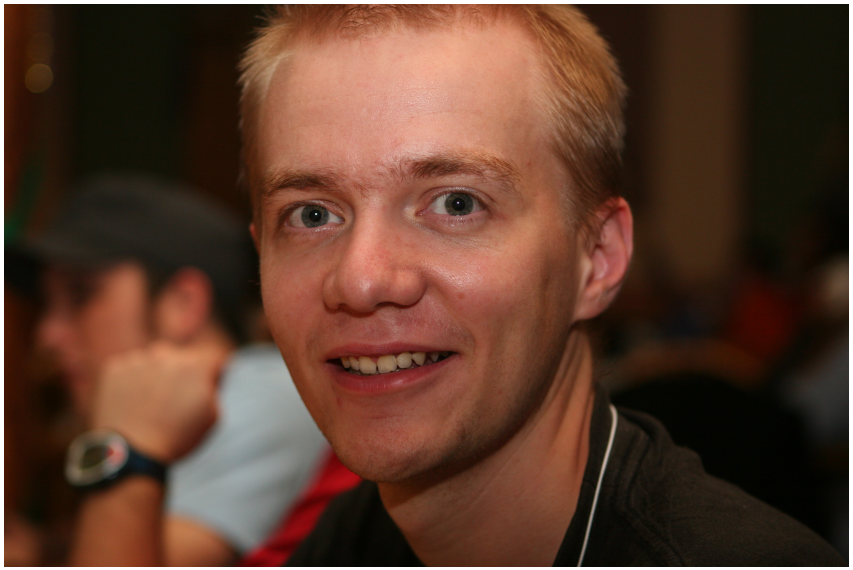
Introduction au développement avec Qt

Kévin Ottens

KDAB

Le maître de séance

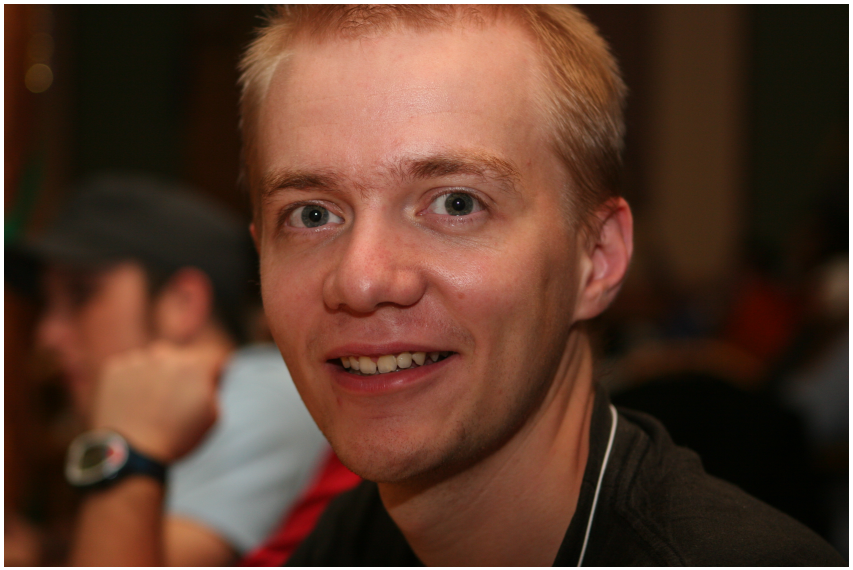
Le petit moment de mégalomanie



- ervin@kde.org
- ottens@irit.fr
- ottens@ups-tlse.fr? et pourtant non...
- kevin@kdab.com

Le maître de séance

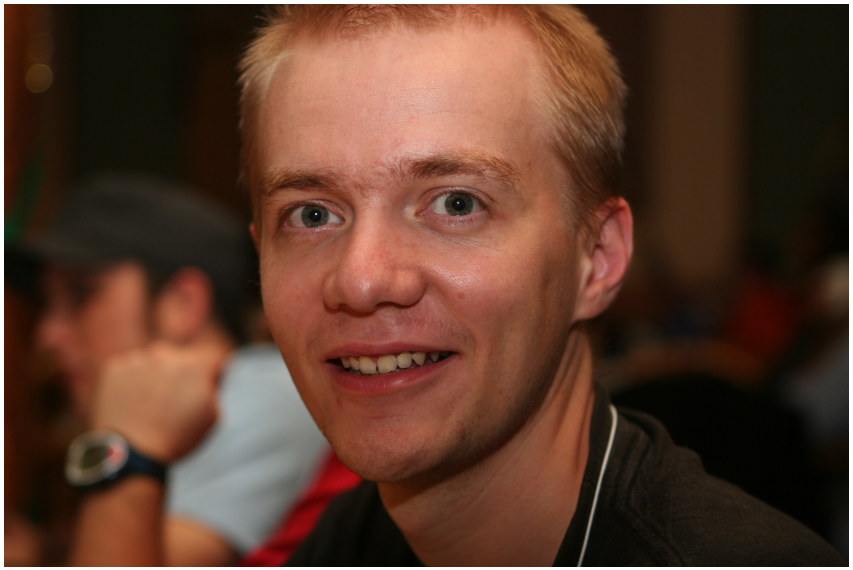
Le petit moment de mégalomanie



- ervin@kde.org
- ottens@irit.fr
- ottens@ups-tlse.fr? et pourtant non...
- kevin@kdab.com

Le maître de séance

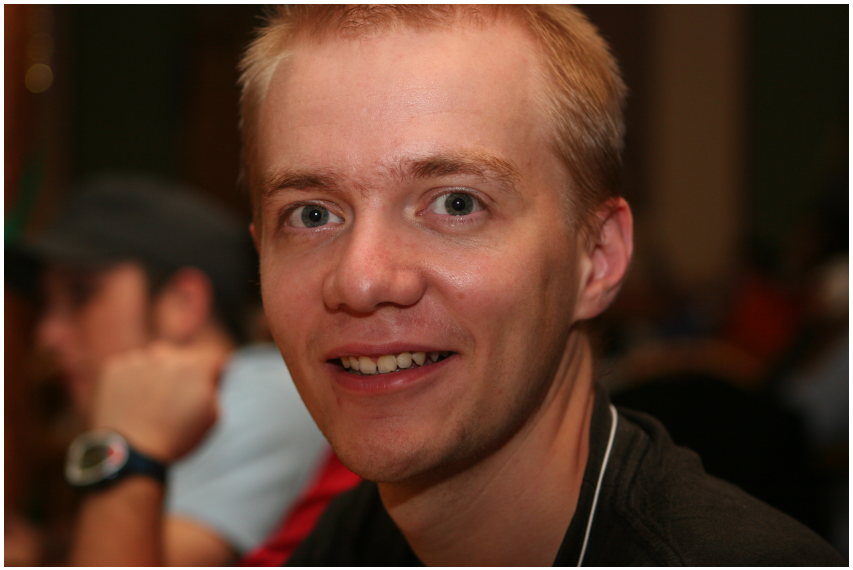
Le petit moment de mégalomanie



- ervin@kde.org
- ~~ottens@irit.fr~~
- ottens@ups-tlse.fr? et pourtant non...
- kevin@kdab.com

Le maître de séance

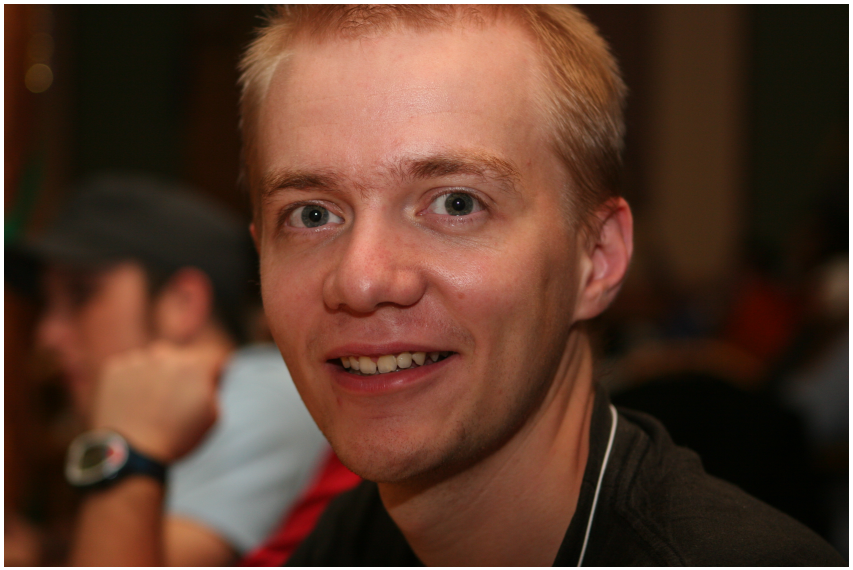
Le petit moment de mégalomanie



- ervin@kde.org
- ~~ottens@irit.fr~~
- ottens@ups-tlse.fr? et pourtant non...
- kevin@kdab.com

Le maître de séance

Le petit moment de mégalomanie



- ervin@kde.org
- ~~ottens@irit.fr~~
- ottens@ups-tlse.fr? et pourtant non...
- kevin@kdab.com

Plan

1 Introduction

2 L’essentiel de Qt

Plan

1 Introduction

2 L'essentiel de Qt

Prérequis et Buts

La connaissance est un mur de briques monté ligne par ligne

C++

- Utilisation des concepts objet
- Templates (au moins un peu, mais rien de poussé)

Système d'exploitation

- Ici *nix
- Savoir ce qu'est un shell

Buts

- Donner un aperçu de ce que l'on peut faire avec Qt
- Donner les moyens d'aller plus loin

Historique (1/2)

Nos expériences passées déterminent ce que nous sommes

Trolltech

- 1994 : Création de Trolltech à Oslo, Norvège
- 1996 : Première vente de Qt ! (ESA)
- 1996 : Qt utilisé par la communauté KDE naissante
- 1999 : Qt2, bureaux en Australie
- 2000 : Qt/Embedded, bureaux aux Etats-Unis
- 2000 : Qt/X11 disponible sous GPL !
- 2001 : Sharp utilise Qtopia dans ses produits
- 2001 : Qt3 !
- 2003 : Qt/Mac disponible sous GPL !
- 2005 : Qt4 !! bureaux en Chine
- 2008 : Nokia rachète Trolltech

Historique (2/2)

Nos expériences passées déterminent ce que nous sommes

Qt Development Frameworks @ Nokia

- Objectifs:
  - Qt Everywhere
  - Multiplier la base de développeurs par 10
- 2008 : Sortie d'une pre-release de Qt/S60
- 2009 : Sortie d'une pre-release de Qt/Maemo
- 2009 : Sortie de Qt 4.6 (environ 20 plateformes supportées)

Qt, un aperçu

De loin tout paraît plus simple

D'après les auteurs

- "Qt is a cross-platform application and UI framework. [...]"
- "Code Less. Create More. Deploy everywhere."

Ce qu'offre Qt

- Du C++ "modifié"
- Des outils et une API multi-plateforme pour...
  - ... développer des interfaces graphiques
  - ... faire des applications multi-thread
  - ... communiquer sur le réseau et entre applications
  - ... accéder à des bases de données, traiter du XML
  - ... interagir avec le web
  - ... interagir avec OpenGL
  - ... et bien plus!

Bibliothèques multi-plateformes (1/3)

Tous les chemins ne mènent pas forcément à Rome

Couche d'API

- Une couche d'API par dessus l'API native
- Exemple: wxWidget
- Avantages: intégration visuelle
- Désavantages:
  - Performance
  - Expose le plus petit dénominateur commun
  - Héritage et spécialisation des widgets difficiles voire impossible

Bibliothèques multi-plateformes (2/3)

Tous les chemins ne mènent pas forcément à Rome

Emulation d'API

- Emule une API native sur les systèmes non natifs
- Exemple: Winelib émule l'API Win32 sur les systèmes UNIX
- Avantage: l'API native est utilisable sur un système
- Désavantages:
  - Performances sur la plateforme non native
  - Maintenance pour les développeurs de la bibliothèque
  - Suit uniquement les fonctionnalités documentées, pas les bugs ou fonctionnalités non documentées et pourtant exploités

Bibliothèques multi-plateformes (3/3)

Tous les chemins ne mènent pas forcément à Rome

Emulation d'UI

- Emule le style graphique en utilisant uniquement les fonctions de tracé bas niveau du système
- Exemple: Qt émule les styles Win32, Motif, etc. sur Win32, Unix, Mac OS, etc.
- Avantages:
  - Performances
  - Les styles ne sont pas liés au système
  - Héritage et spécialisation des widgets aisé
- Désavantage: les développeurs de la bibliothèque doivent suivre les modifications de style

Plan

1 Introduction

2 L’essentiel de Qt

Plan

2 L’essentiel de Qt
- Concepts clefs
- Widgets
- Outils pour le développement

QObject

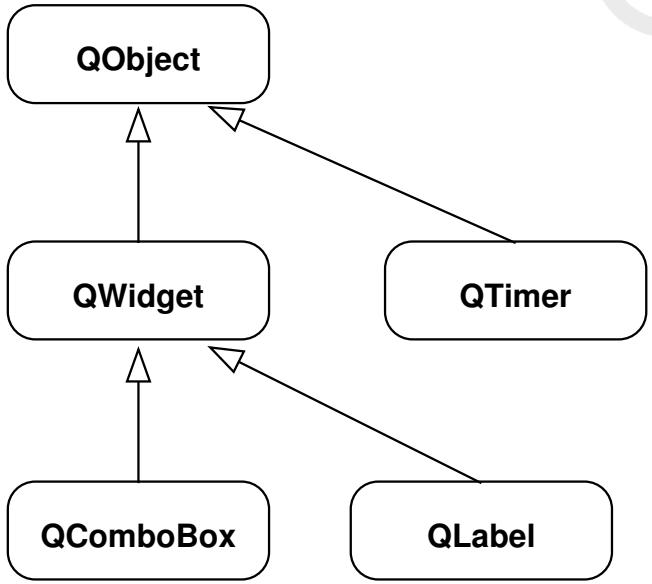
Le père de toute chose... ou presque

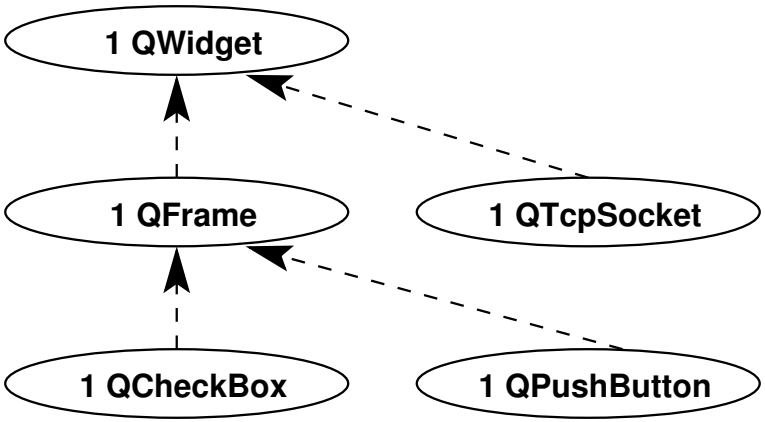
Fonctionnalités

- Relation parent/enfant, utile pour
  - Gestion mémoire
  - Visibilité et état des widgets
- Signaux et slots
- Introspection
- Système de propriétés

Inconvénients

- Difficultés pour l'héritage multiple





QObject

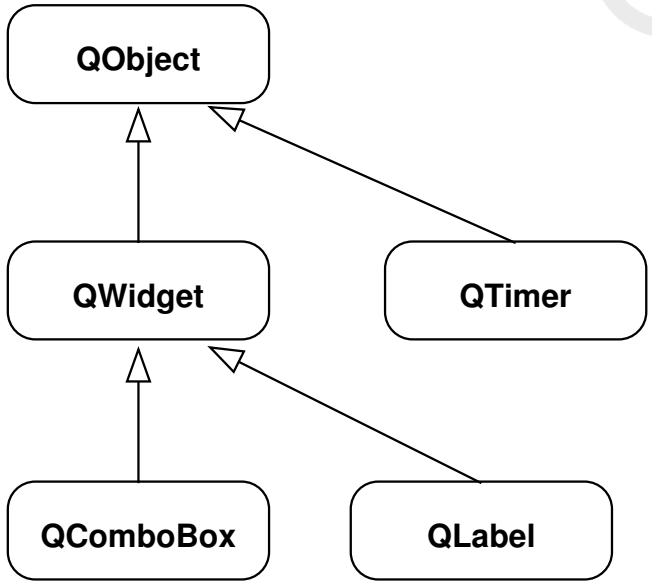
Le père de toute chose... ou presque

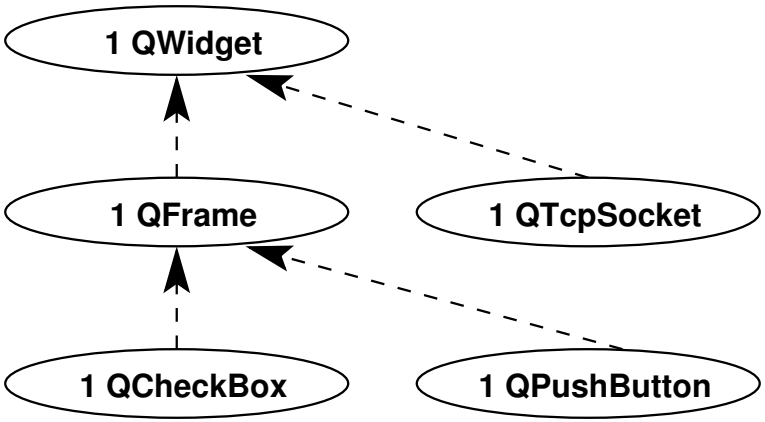
Fonctionnalités

- Relation parent/enfant, utile pour
  - Gestion mémoire
  - Visibilité et état des widgets
- Signaux et slots
- Introspection
- Système de propriétés

Inconvénients

- Difficultés pour l'héritage multiple





QObject

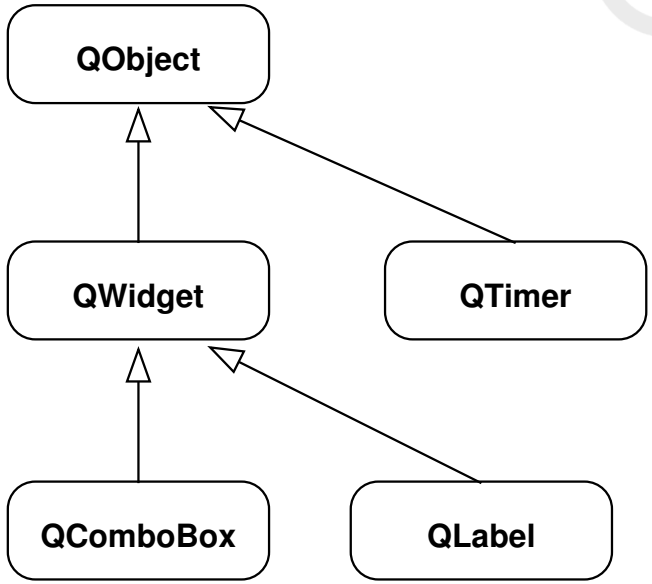
Le père de toute chose... ou presque

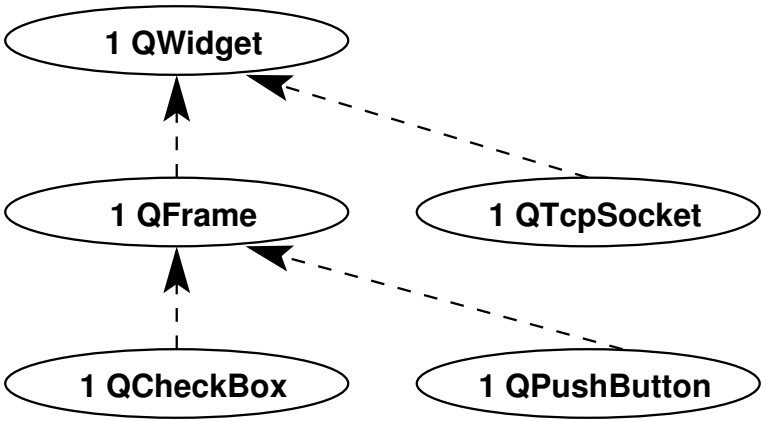
Fonctionnalités

- Relation parent/enfant, utile pour
  - Gestion mémoire
  - Visibilité et état des widgets
- Signaux et slots
- Introspection
- Système de propriétés

Inconvénients

- Difficultés pour l'héritage multiple





Signaux et Slots (1/5)

Le Morse c'est dépassé

Métaphore

- Un objet "émet" (emit) un signal pour signifier quelque chose potentiellement intéressant pour l'extérieur
- Un ou plusieurs objets recoivent le signal par une méthode de signature compatible

Primitives

- `connect()` / `disconnect()`
- `emit`

Avantages

- Couplage faible
- Programmation événementielle facile

Signaux et Slots (2/5)

Le Morse c'est dépassé

Phare

```
#include <QtCore/QObject>
#include <QtCore/QPoint>

class Beacon : public QObject
{
    Q_OBJECT

signals:
    void beamOfLight(QPoint pos, int degree);
};
```

Signaux et Slots (3/5)

Le Morse c'est dépassé

Bateau

```
#include <QtCore/QObject>
#include <QtCore/QPoint>

class Boat : public QObject
{
    Q_OBJECT

public slots:
    void lightSpotted(QPoint pos, int degree);
};
```

Signaux et Slots (4/5)

Le Morse c'est dépassé

Connection

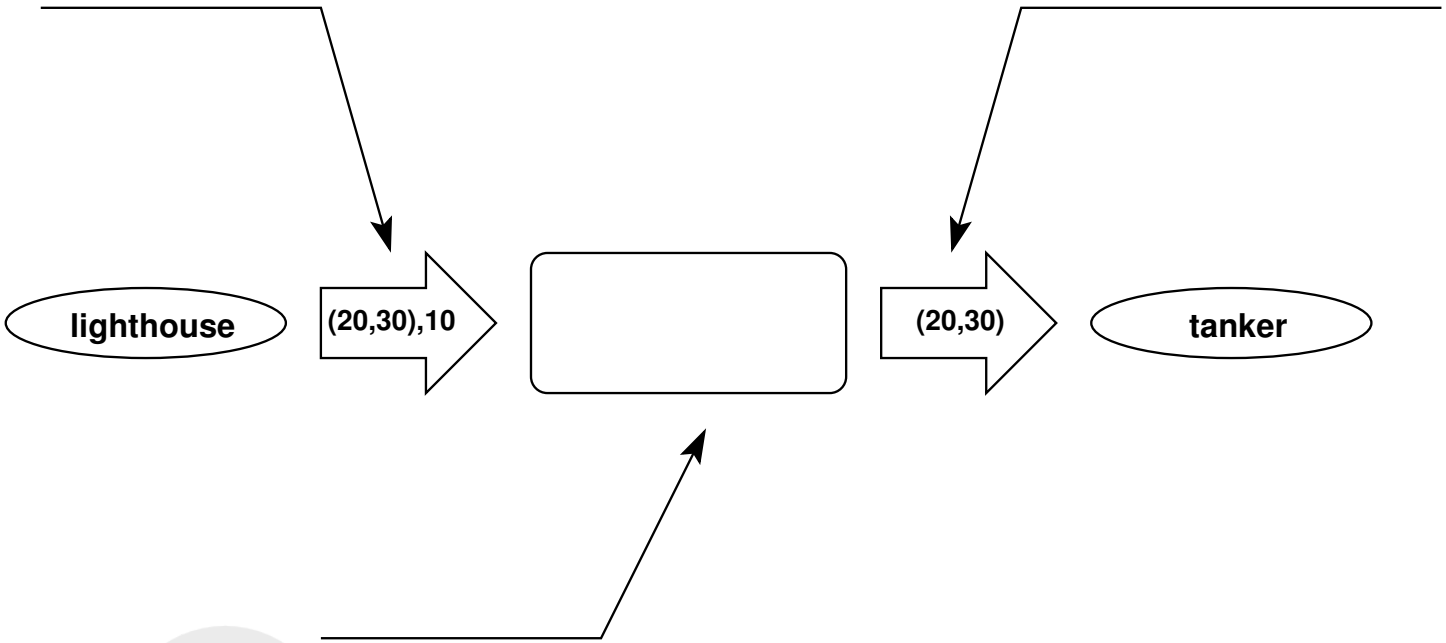
```
Beacon *lighthouse;
Boat *tanker;

[...]

connect(lighthouse,
        SIGNAL(beamOfLight(QPoint, int)),
        tanker,
        SLOT(lightSpotted(QPoint, int)));
```

Signaux et Slots (5/5)

Le Morse c'est dépassé

```
void Beacon::rotate(int angle)
{
    ...
    mAngle+= angle;
    emit beamOfLight(mPosition, mAngle);
    ...
}
```

```
void Boat::lightSpotted(const QPoint &pos)
{
    ...
    mLastLightPosition = pos;
    ...
}
```




```
connect(lighthouse, SIGNAL(beamOfLight(QPoint, int)),
        tanker, SLOT(lightSpotted(QPoint)));
```

Partage implicite

Ce qui est à toi est à moi... ou presque

"Copy on write"

- Utilisable par valeur
- `operator=()` duplique uniquement un pointeur
- Duplication des données sur écriture

Partage implicite

Ce qui est à toi est à moi... ou presque

"Copy on write"

- Utilisable par valeur
- `operator=()` duplique uniquement un pointeur
- Duplication des données sur écriture

QString s1 = "foo";

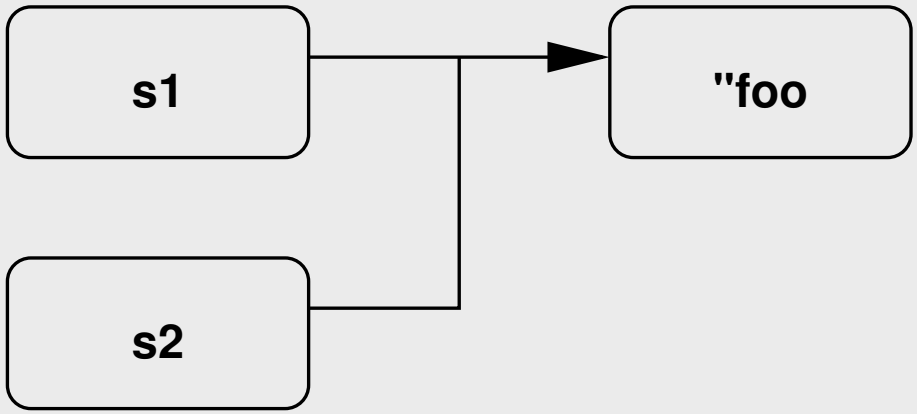
Partage implicite

Ce qui est à toi est à moi... ou presque

"Copy on write"

- Utilisable par valeur
- `operator=()` duplique uniquement un pointeur
- Duplication des données sur écriture

QString s2 = s1;



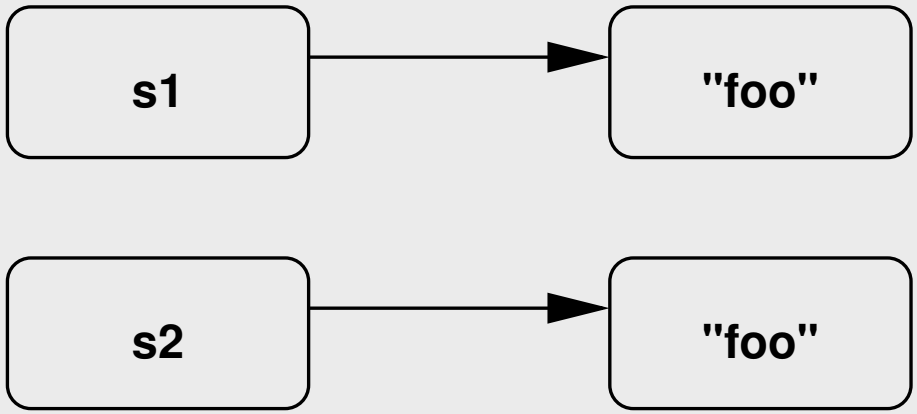
Partage implicite

Ce qui est à toi est à moi... ou presque

"Copy on write"

- Utilisable par valeur
- `operator=()` duplique uniquement un pointeur
- Duplication des données sur écriture

s1.append("/bar");



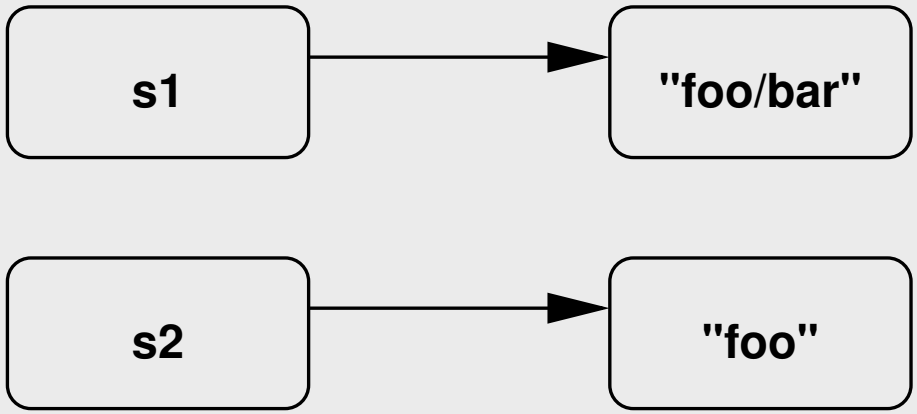
Partage implicite

Ce qui est à toi est à moi... ou presque

"Copy on write"

- Utilisable par valeur
- `operator=()` duplique uniquement un pointeur
- Duplication des données sur écriture

s1.append("/bar");



Partage implicite

Ce qui est à toi est à moi... ou presque

"Copy on write"

- Utilisable par valeur
- `operator=()` duplique uniquement un pointeur
- Duplication des données sur écriture

Classes couvertes

Images, polygones, chaînes, url, variants, collections...

Partage implicite

Ce qui est à toi est à moi... ou presque

"Copy on write"

- Utilisable par valeur
- `operator=()` duplique uniquement un pointeur
- Duplication des données sur écriture

(A propos des collections)

- Tout le nécessaire : `QList`, `QMap`, `QSet`, `QMultiMap`...
- Trois styles possibles pour itérer :
  - Itérateurs STL
  - Itérateurs Java
  - `foreach()`

Plan

2 L’essentiel de Qt
- Concepts clefs
- Widgets
- Outils pour le développement

Petit catalogue de widgets (1/3)

Ooooh, c'est beau ! Et il y en a beaucoup...

Boutons

- QCheckBox
- QRadioButton
- QPushButton
- QToolButton
- QButtonGroup (pas un widget!)

Saisie

- QLineEdit
- QTextEdit
- QComboBox
- QCompleter (pas un widget!)
- QValidator (pas un widget!)

Petit catalogue de widgets (2/3)

Ooooh, c'est beau ! Et il y en a beaucoup...

Affichage

- QLabel
- QLCDNumber
- QTextBrowser

Intervals

- QSlider
- QDial
- QScrollBar
- Q(Double)SpinBox

Petit catalogue de widgets (3/42)

Ooooh, c'est beau ! Et il y en a beaucoup...

Et aussi...

- Gestion des dates et du temps
- Arranger d'autres widgets (splitter, scroll areas...)
- Constituants de fenêtres principales (menus, tool bars, onglets, docks...)
- Boites de dialogues (fichiers, couleurs, polices...)

Interview (1/3)

Modèle/Vue enfin abordable

A propos du MVC

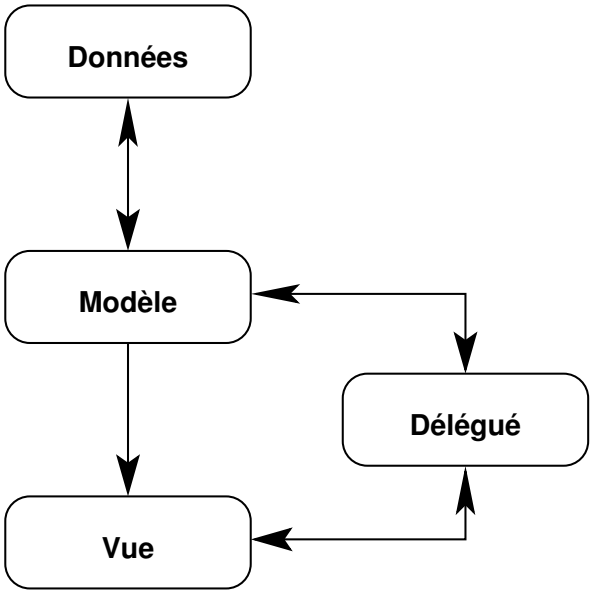
- Pattern d'**Architecture**
- Assez simple à comprendre... généralement difficile à mettre en œuvre
- Applicable pour des widgets complexes, ou des boîtes de dialogue

Originalités d'Interview

- Mise en place d'une architecture MVC pour les widgets complexes
- Interface de modèles "générique" (table/liste/arbre)
- Communication par signaux et slots entre les objets intervenants

Interview (2/3)

Modèle/Vue enfin abordable



Modèle/Vue/Délégué

- Le modèle interface la source de données avec les autres composants
- La vue demande au modèle les données à afficher
- Le délégué effectue le rendu conjointement avec la vue, et lors d'une édition indique les changements au modèle
- Toutes les références aux données passent par des instances de `QModelIndex`

Interview (3/3)

Modèle/Vue enfin abordable

Classes disponibles

- `QAbstractItemModel` : Interface de base pour les modèles
- `QAbstractListModel` : Modèles orientés liste
- `QAbstractTableModel` : Modèles orientés table
- `QListView` : Vue en liste
- `QTableView` : Vue en table
- `QTreeView` : Vue en arbre
- `QAbstractProxyModel` : Interface pour "modèles filtrants"

QGraphicsView

Séparation modèle/vue pour le canvas

Modèle

- `QGraphicsScene`
  - Permet la gestion des objets dans la scène
  - Distribue les événements aux objets et gère leur état
- `QGraphicsItem`, objets affichables
  - Texte, lignes, polygônes, pixmap, SVG, etc.
  - Détection de collisions
  - Drag'n'Drop, événements clavier et souris, etc.

Vue

- Plusieurs vues sur la même scène
- Chaque vue peut avoir des options de rendu, ou des transformations différentes

Mais aussi...

En bref !

Scribe : Gestion de texte riche

- `QTextDocument` : contenu d'un texte en lecture seule
  - Structure hiérarchique
  - `QTextBlock`, `QTextFrame`, `QTextTable`, `QTextList`
- `QTextCursor` permet de manipuler un texte en utilisant la métaphore du curseur
- `QTextEdit` : widget d'édition de texte

Mainwindow : flexibilité dans la fenêtre principale

- `QMainWindow` : Fenêtre principale disposant d'un menu
- `QToolBar` : Barres d'outils en bords de fenêtres
- `QDockWidget` : Panneaux déplaçables liés à la fenêtre

Plan

2 L’essentiel de Qt
- Concepts clefs
- Widgets
- Outils pour le développement

QMake (1/2)

Construire c'est ma passion... eeeuuh, en fait non

Faciliter la portabilité

- Description des règles de construction d'une application
- Génération de fichiers Makefile ou Visual Studio
- Possibilité d'ajouter des réglages particuliers pour une plateforme donnée

Variables importantes

- `TEMPLATE`: type de projet (app, lib, subdirs)
- `HEADERS`, `SOURCES` et `FORMS`: fichiers utilisés pour la construction
- `TARGET`: nom du produit construit
- `CONFIG`: options de construction (debug, multi-thread...)

QMake (2/2)

Construire c'est ma passion... eeeuuh, en fait non

Un exemple rapide

```
TEMPLATE = app
TARGET   = seashore_simulator

CONFIG   += qt warn_on release

HEADERS  = beacon.h boat.h
SOURCES  = main.cpp beacon.cpp boat.cpp
FORMS    = seashoreview.ui
```

Qt Designer (1/2)

Dessiner c'est gagner

Constats

- Ecrire l'IHM à la main est fastidieux
- Code résultant souvent lourd à maintenir

designer + uic

- Designer : éditeur graphique d'IHM
- Résultat : fichiers ".ui"
- uic : tranforme les ".ui" en classes C++ (public:)

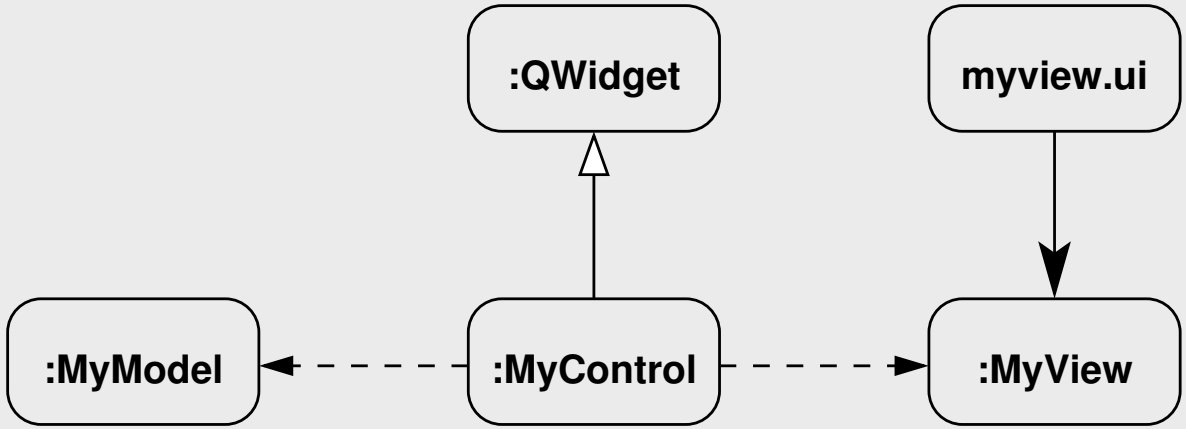
Approche de travail

- On ne conserve **que** les ".ui"
- Classes générées uniquement pour la construction
- Incite au MVC

Qt Designer (2/2)

Dessiner c'est gagner

MVC avec Designer

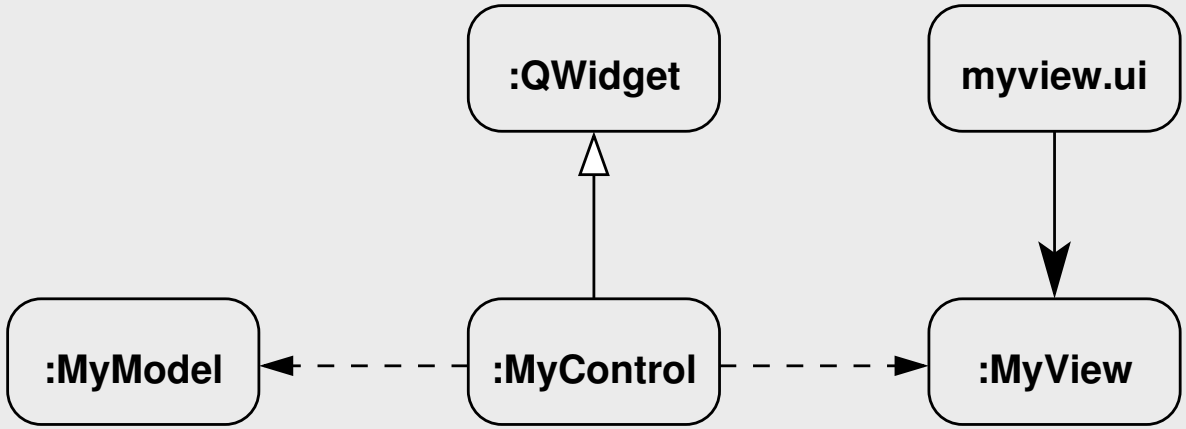


- `MyView` est générée par uic à partir de `myview.ui`
- `MyModel` est une classe domaine à afficher ou éditer
- `MyControl` écoute les signaux de `MyModel` et des widgets de `MyView`
- Hériter d'un `QWidget` permet en plus d'utiliser le contrôle comme élément dans une interface plus complexe

Qt Designer (2/2)

Dessiner c'est gagner

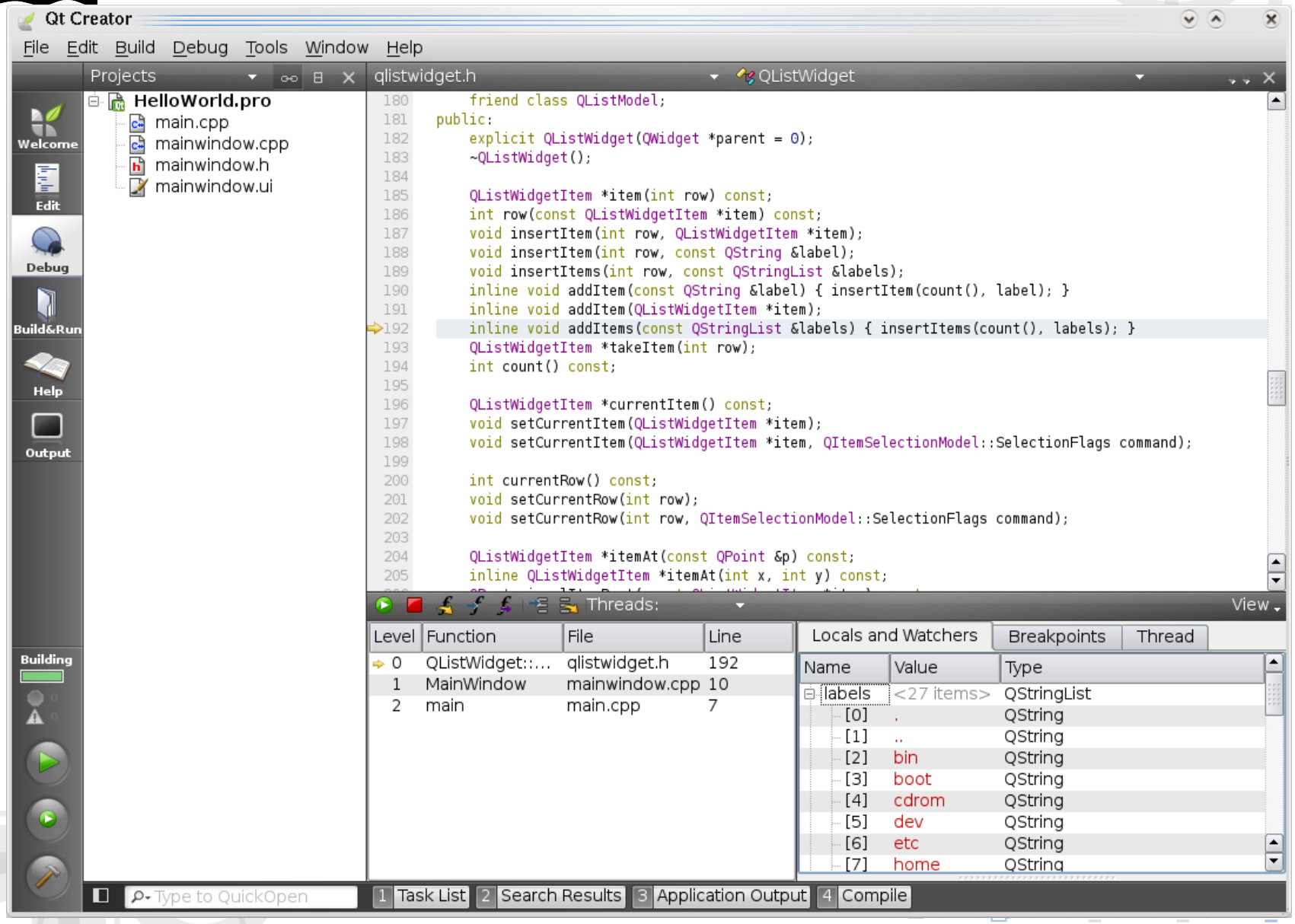
MVC avec Designer



- `MyView` est générée par uic à partir de `myview.ui`
- `MyModel` est une classe domaine à afficher ou éditer
- `MyControl` écoute les signaux de `MyModel` et des widgets de `MyView`
- Hériter d'un `QWidget` permet en plus d'utiliser le contrôle comme élément dans une interface plus complexe

Qt Creator

Le tout intégré



Sites Web (Qt)

Pour les geeks qui ne lachent pas le clavier

Qt Development Frameworks @ Nokia

`http://qt.nokia.com`

Qt Online Reference Documentation

`http://qt.nokia.com/doc`

Qt fr: Le site Francophone

`http://www.qtfr.org`

QtCentre: The Ultimate Qt Community site

`http://www.qtcentre.org`

Qt Labs' blogs

`http://labs.qt.nokia.com/blogs`

Ateliers KDE

Page de publicité

Un samedi par mois...

Ateliers KDE

Page de publicité

Venez participer aux projets de la communauté KDE!

Ateliers KDE

Page de publicité

Toutes les contributions sont les bienvenues (pas uniquement sous forme de code)

Ateliers KDE

Page de publicité

Pour chaque Atelier KDE au moins une présentation technique offerte

Ateliers KDE

Page de publicité

Les Ateliers KDE de Toulibre rendent la bonne humeur et les cheveux soyeux

Questions ?

Kévin Ottens
ervin@kde.org