

Interaction entre boost.signals et les signaux/slots Qt

par Denys Bulant ([Tutoriels Qt](#))

Date de publication : 19 mai 2008

Dernière mise à jour :

Cet article est destiné à vous guider pour mettre en place une interaction entre boost.signals et le mécanisme de signaux/slots Qt.

I - Pré-requis.....	3
II - Introduction.....	3
III - Précautions et conflits.....	3
IV - Utilisation d'un boost.signal connecté à un slot Qt.....	3
V - Utilisation d'un boost.signal connecté à un signal Qt.....	4
VI - Utilisation d'un signal Qt connecté à une (ou des) fonction(s) C++ standard par le biais de boost.signals.....	5
VII - Code récapitulatif.....	7
VIII - Remerciements.....	9

I - Pré-requis

- Une installation de Qt4 fonctionnelle
- une connaissance minimale du **principe des signaux/slots avec Qt**
- une connaissance minimale de **boost.signals** et **boost.functional** (fonction de binding)

II - Introduction

Il est intéressant de savoir faire ceci dans la mesure où vous désireriez utiliser des classes écrites en C++ pur avec les mécanismes Qt. Par exemple, une couche métier utilisant le mécanisme de boost.signals comme implémentation du pattern observateur, que vous désirez connecter à la couche IHM écrite à l'aide de Qt. Toutefois, cet article n'est ni une introduction à Qt, ni à boost.signals.

III - Précautions et conflits

Premièrement, il y a un conflit entre boost et Qt au niveau du terme **signals**. En effet, Qt déclare celui-ci comme `#define signals protected`, tandis qu'il s'agit d'un type dans la bibliothèque boost. Il y a plusieurs façons de résoudre ce conflit:

- Inclure les en-têtes boost en premier
- Placer boost.signals dans son propre namespace (pour plus d'infos se référer à [la Q/R 3 de la FAQ boost.signals](#))
- Depuis Qt 4.1, Trolltech a ajouté le paramètre `no_keywords` à la liste des valeurs potentielles pour CONFIG. Ajouter `CONFIG += no_keywords` dans votre fichier pro vous permettra donc d'utiliser sans soucis boost.signal. Le bémol de cette solution est que les "mots clés" suivants deviennent donc:

mot clé classique	remplacement si no_keywords est spécifié
signals	Q_SIGNALS
slots	Q_SLOTS
emit	Q_EMIT
foreach	Q_FOREACH
forever	Q_FOREVER

Les slots de Qt sont des fonctions membres standard. Il y a cependant une fonctionnalité liée qui est la possibilité pour le slot de connaître son émetteur (à la condition que son appel soit lié à l'émission d'un signal Qt). L'usage de cette fonction est grandement déconseillé en temps normal (**QSignalMapper** devrait amplement suffire pour s'en passer). Mais dans certains rares cas, vous pouvez avoir absolument besoin de cette info; au quel cas pensez bien à vérifier que le `QObject*` retourné n'est pas null. Dans ce dernier cas, c'est que le slot a été invoqué de façon normale (c'est ce que l'on fait en connectant un slot Qt à un signal boost).

IV - Utilisation d'un boost.signal connecté à un slot Qt

Un slot au sens Qt est en réalité une fonction standard. Par conséquent, il est trivial d'en connecter un à un signal boost.

Cependant, la seule chose différente entre un slot et une fonction membre standard est la possibilité de récupérer des infos sur l'objet appelant (par le biais de `QObject::sender()`). Je tiens à souligner à nouveau le "danger" posé par cette fonction membre: si le slot n'est pas appelé par un signal, le `QObject*` renvoyé sera null. Par conséquent, réduisez au minimum son utilisation, et vérifiez toujours la validité de l'objet renvoyé!

```

class QtClass : public QObject
{
    Q_OBJECT
public slots:
    void qtSlot(float x)
    {
        QObject *s = sender();
        if(s == this)
            std::cout << "From internal connection:" << std::endl;
        else if(!s)
            std::cout << "Unknown sender:" << std::endl;
        std::cout << "In QtClass::qtSlot(), x=" << x << std::endl;
    }
};

class NonQtClass
{
public:
    void boostSignalToQtSlot(QtClass *qc)
    {
        boost::signal<void (float)> sig;
        sig.connect(boost::bind(&QtClass::qtSlot, qc, _1));
        sig(3.14f);
    }
};

int main(int argc, char **argv)
{
    NonQtClass nqc;
    QtClass qc;

    // Déclenche un boost::signal attaché à un slot Qt:
    nqc.boostSignalToQtSlot(&qc);

    return 0;
}
    
```

Sortie obtenue:

```

Unknown sender:
In QtClass::qtSlot(), x=3.14
    
```

V - Utilisation d'un boost.signal connecté à un signal Qt

Un signal au sens Qt n'est qu'une fonction membre standard, le mot clé **signals** étant redéfini par Qt comme **protected**. Il est donc nécessaire de fournir une méthode permettant de déclencher ce signal avec le(s) paramètre(s) voulu(s). Il y a 2 approches possibles; la première consiste à ajouter une fonction à votre classe Qt, la seconde à utiliser une classe de pont.

Bien que plus verbeuse, la seconde méthode permet de découpler réellement la couche utilisant Qt de la couche sans Qt en "intercalant" une couche servant d'adaptateur. Ici, ce sera la première méthode qui sera illustrée; la seconde le sera dans le 3ème et dernier scénario.

```

class QtClass : public QObject
{
    Q_OBJECT

public:
    QtClass()
    {
        connect(this, SIGNAL(qtSignal(float)), this, SLOT(qtSlot(float)));

        // OBLIGATOIRE si on veut connecter un signal boost à un signal Qt: les signaux sont **protégés**
    }
}
    
```

```

void triggerQtSignalEmission(float x)
{
    emit qtSignal(x);
}

public slots:
void qtSlot(float x)
{
    QObject *s = sender();
    if(s == this)
        std::cout << "From internal connection:" << std::endl;
    else if(!s)
        std::cout << "Unknown sender:" << std::endl;
    std::cout << "In QtClass::qtSlot(), x=" << x << std::endl;
}

signals:
void qtSignal(float);
};

class NonQtClass
{
public:
void boostSignalToQtSignal(QtClass *qc)
{
    boost::signal<void (float)> sig;
    sig.connect(boost::bind(&QtClass::triggerQtSignalEmission, qc, _1));
    sig(42.0f);
}
};

int main(int argc, char **argv)
{
    NonQtClass nqc;
    QtClass qc;

    /* Déclenche un boost::signal lié à l'émission d'un signal Qt
    (on triche par l'utilisation d'une fonction membre intermédiaire obligatoire) */
    nqc.boostSignalToQtSignal(&qc);

    return 0;
}
    
```

Sortie obtenue:

```

From internal connection:
In QtClass::qtSlot(), x=42
    
```

VI - Utilisation d'un signal Qt connecté à une (ou des) fonction(s) C++ standard par le biais de boost.signals

Ici, ça se complique un peu (vraiment un peu, ne vous inquiétez pas). En effet, nous savons:

- qu'un signal Qt est une fonction membre protégée
- que le corps d'un signal Qt va associer les informations sur l'objet émetteur (dans le but de permettre l'utilisation de la fonction `QObject::sender()` dans le slot)
- que nous ne pouvons pas connecter de fonction standard avec un signal Qt

Nous allons donc mettre en place ici une méthode impliquant la création d'une classe servant d'adaptateur. Cette classe connectera les signaux d'intérêt de la classe Qt à des slots privés, et utilisera un `boost.signal` pour faire le pont vers les fonctions standard.

```

class QtClass : public QObject
{
    Q_OBJECT

public:
    // OBLIGATOIRE si on veut connecter un signal boost à un signal Qt: les signaux sont **protégés**
    void triggerQtSignalEmission(float x)
    {
        emit qtSignal(x);
    }

signals:
    void qtSignal(float);
};

class QtSignalToStandardCPPBridge : public QObject
{
    Q_OBJECT

public:
    QtSignalToStandardCPPBridge(QtClass *qc)
    {
        connect(qc, SIGNAL(qtSignal(float)), this, SLOT(onQtSignalEmitted(float)));
    }

    template<typename Signature>
    void addStandardSlot(Signature fun)
    {
        sig.connect(fun);
    }

private slots:
    void onQtSignalEmitted(float x)
    {
        sig(x);
    }

private:
    boost::signal<void (float)>sig;
};

class NonQtClass
{
public:
    // fonction normale que l'on appellera depuis un signal Qt en passant par un pont
    void normalFunction(float x)
    {
        std::cout << "In NonQtClass::normalFunction(), x=" << x << std::endl;
    }

    // Et une autre pour le fun
    void displaySquareRoot(float x)
    {
        std::cout << "In NonQtClass::displaySquareRoot(), sqrt(x)=" << std::sqrt(x) << std::endl;
    }

public:
    void QtSignalToStandardFunction(QtClass *qc)
    {
        QtSignalToStandardCPPBridge adapter(qc);
        adapter.addStandardSlot(boost::bind(&NonQtClass::normalFunction, this, _1));
        adapter.addStandardSlot(boost::bind(&NonQtClass::displaySquareRoot, this, _1));
        // On déclenche ici manuellement l'émission du signal Qt
        qc->triggerQtSignalEmission(121.0f);
    }
};

int main(int argc, char **argv)
{
    NonQtClass nqc;
    QtClass qc;

```

```
nqc.QtSignalToStandardFunction(&qc);

return 0;
}
```

Sortie obtenue:

```
In NonQtClass::normalFunction(), x=121
In NonQtClass::displaySquareRoot(), sqrt(x)=11
```

VII - Code récapitulatif

N'oubliez pas d'ajouter la directive "CONFIG += console" à votre fichier .pro pour voir les sorties, ainsi que de régler les valeurs de LIBS et INCLUDEPATH correspondant à votre installation. Ce code est présent dans **l'archive téléchargeable ici (mirroir)**.

```
#include <boost/signal.hpp>
#include <boost/bind.hpp>

#include <QtCore>

#include <iostream>
#include <cmath>

//-----
// Partie Qt
//-----

class QtClass : public QObject
{
    Q_OBJECT

public:
    QtClass()
    {
        connect(this, SIGNAL(qtSignal(float)), this, SLOT(qtSlot(float)));
    }

    // OBLIGATOIRE si on veut connecter un signal boost à un signal Qt: les signaux sont **protégés**
    void triggerQtSignalEmission(float x)
    {
        emit qtSignal(x);
    }

public slots:
    void qtSlot(float x)
    {
        QObject *s = sender();
        if(s == this)
            std::cout << "From internal connection:" << std::endl;
        else if(!s)
            std::cout << "Unknown sender:" << std::endl;
        std::cout << "In QtClass::qtSlot(), x=" << x << std::endl;
    }

signals:
    void qtSignal(float);
};

//-----
// Adapteur Qt
//-----

class QtSignalToStandardCPPBridge : public QObject
{
    Q_OBJECT
```

```

public:
    QtSignalToStandardCPPBridge(QtClass *qc)
    {
        connect(qc, SIGNAL(qtSignal(float)), this, SLOT(onQtSignalEmitted(float)));
    }

    template<typename Signature>
    void addStandardSlot(Signature fun)
    {
        sig.connect(fun);
    }

private slots:
    void onQtSignalEmitted(float x)
    {
        sig(x);
    }

private:
    boost::signal<void (float)>sig;
};

// Nécessaire dû à l'utilisation des macros Q_OBJECT dans QtClass et QtToBoostBridge
#include "main.moc"

//-----
// Partie non Qt
//-----

class NonQtClass
{
public:
    // fonction normale que l'on appellera depuis un signal Qt en passant par un pont
    void normalFunction(float x)
    {
        std::cout << "In NonQtClass::normalFunction(), x=" << x << std::endl;
    }

    // Et une autre pour le fun
    void displaySquareRoot(float x)
    {
        std::cout << "In NonQtClass::displaySquareRoot(), sqrt(x)=" << std::sqrt(x) << std::endl;
    }

public:
    void boostSignalToQtSlot(QtClass *qc)
    {
        boost::signal<void (float)> sig;
        sig.connect(boost::bind(&QtClass::qtSlot, qc, _1));
        sig(3.14f);
    }

    void boostSignalToQtSignal(QtClass *qc)
    {
        boost::signal<void (float)> sig;
        sig.connect(boost::bind(&QtClass::triggerQtSignalEmission, qc, _1));
        sig(42.0f);
    }

    void QtSignalToStandardFunction(QtClass *qc)
    {
        QtSignalToStandardCPPBridge adapter(qc);
        adapter.addStandardSlot(boost::bind(&NonQtClass::normalFunction, this, _1));
        adapter.addStandardSlot(boost::bind(&NonQtClass::displaySquareRoot, this, _1));
        // On déclenche ici manuellement l'émission du signal Qt
        qc->triggerQtSignalEmission(121.0f);
    }
};

int main(int argc, char **argv)
{

```

```
NonQtClass nqc;
QtClass      qc;

// Déclenche un boost::signal attaché à un slot Qt:
nqc.boostSignalToQtSlot(&qc);

// Déclenche un boost::signal lié à l'émission d'un signal Qt
// (on triche par l'utilisation d'une fonction membre intermédiaire obligatoire)
nqc.boostSignalToQtSignal(&qc);

nqc.QtSignalToStandardFunction(&qc);

return 0;
}
```

Sortie obtenue:

```
Unknown sender:
In QtClass::qtSlot(), x=3.14
From internal connection:
In QtClass::qtSlot(), x=42
From internal connection:
In QtClass::qtSlot(), x=121
In NonQtClass::normalFunction(), x=121
In NonQtClass::displaySquareRoot(), sqrt(x)=11
```

(Notez que pour le dernier scénario, la connexion au slot Qt étant toujours active, le slot reste appelé par l'émission de QtClass::qtSignal(float).)

VIII - Remerciements

Trolltech et les développeurs de Boost en premier lieu :) Puis Aurélien Régat-Barrel, Mongaulois, Alp et JauB pour leur relectures et critiques