

Ouverture d'une boîte de dialogue avec Qt 4

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

Date de publication :

Dernière mise à jour :

Ce tutoriel est destiné aux nouveaux venus à Qt qui cherchent à savoir comment ouvrir une fenêtre à partir de leur fenêtre principale. Nous verrons 2 façons de faire : utilisation d'une fenêtre principale avec menus pour ouvrir une boîte de dialogue et une boîte "A propos..." ET ouverture d'une fenêtre de dialogue par le biais d'un bouton

J'en profiterais pour aborder très brièvement 2 autres points : l'utilisation de **QAction** et les 2 modes d'affichage des boîtes de dialogue.

- I - Pré-requis
- II - Une introduction aux boîtes de dialogue
- III - A propos de ce tutoriel
- IV - Lectrice/Lecteur, voici QDialog; QDialog, voici ton futur maître ;-)
 - IV-1 - De la modalité et différences entre les fonctions d'affichage
 - IV-2 - Valeurs de retours
 - IV-3 - Une boîte de dialogue...
- V - Ouverture de fenêtre avec des boutons
 - V-1 - Classes abordées
 - V-2 - Description de la solution
 - V-3 - Exemple
- VI - Utilisation d'une fenêtre principale avec menus pour ouvrir une boîte de dialogue et une boîte "A propos..."
 - VI-1 - Classes abordées
 - VI-2 - Description de la solution
 - VI-3 - Exemple
- VII - Archive
- VIII - Conclusion

I - Pré-requis

- Une installation de Qt4 fonctionnelle
- une connaissance minimale du principe des signaux/slots avec Qt

II - Une introduction aux boîtes de dialogue

Les boîtes de dialogue sont des composants essentiels à la très grosse majorité des applications. Qu'il s'agisse de boîte de configuration ou de fournir une interactivité à un niveau plus ou moins bas avec le document ouvert (par exemple, la recherche dans un traitement de texte ou un tableur), vous aurez souvent l'occasion d'en utiliser.

On distingue 2 types d'utilisation des boîtes de dialogue:

- non modale: l'utilisateur peut continuer d'interagir avec le reste de l'application
- modale: l'utilisateur est obligé de fermer la boîte de dialogue affichée avant de pouvoir à nouveau se servir de l'application

Chaque usage à son but; par exemple, une boîte de dialogue permettant une recherche dans un document aura tout intérêt à être non modale par souci d'ergonomie. Par contre, une fenêtre de configuration devrait plutôt être modale, due à une éventuelle dépendance du résultat sur le comportement de l'application.

III - A propos de ce tutoriel

Ce tuto vous permettra, je l'espère de vous familiariser avec **QDialog** principalement, et plus globalement l'ouverture de fenêtre à partir d'une autre. Je fourni ici 2 approches:


- la première est l'ouverture d'une fenêtre en appuyant sur un bouton: elle est la méthode la plus simple, mais pas forcément la plus commune
- la seconde fait utilisation d'un menu, et vous montre comment afficher une boîte de type "A propos...". Il y est fait appel à 2 concepts supplémentaires: les menus et les actions. Le code est loin d'être compliqué, mais assurez-vous d'avoir compris la première partie pour lire la seconde.

Comme dit dans l'introduction, une version pyQt a été réalisée par [alteo_gange](#) . Elle est disponible sur le [forum Qtfr](#) .

IV - Lectrice/Lecteur, voici QDialog; QDialog, voici ton futur maître ;-)

La documentation de la classe QDialog se trouve  [ici](#) .

Avant de sauter dans le vif du sujet, faisons plus ample connaissance avec la classe **QDialog** . Vous vous en doutez probablement, elle réalise ce qu'une boîte de dialogue est sensée faire, et permet quelques raccourcis que vous n'auriez pas si vous dériviez d'un **QWidget** .

Cette section sert de très brève introduction aux fonctions les plus basiques de **QDialog** . Je vous invite à aller lire  [la description détaillée](#) sur la page de documentation de Trolltech. Ils en parlent bien mieux que moi.

IV-1 - De la modalité et différences entre les fonctions d'affichage

Afficher une instance de **QDialog** peut se faire de plusieurs façons. En voici un aperçu:

- `QDialog::show()` : utilisée pour un affichage de la boîte de dialogue; par défaut l'affichage est non modal; peut servir à afficher un **QDialog** de façon modale si `setModal(true)` à été appelée précédemment.
- `QDialog::hide()` : permet de cacher une boîte de dialogue
- `QWidget::setVisible()` et `QWidget::isVisible()` : **QDialog** dérive de **QWidget** , et fournit donc cette alternative à `show()/hide()` . L'utilité de ces fonctions se trouve principalement dans la possibilité de changer la visibilité d'une fenêtre en une ligne de code.
- `QDialog::exec()` : sert à afficher une boîte de dialogue modale et éventuellement récupérer le code de retour

IV-2 - Valeurs de retours

Il peut être parfois utile de savoir si l'interaction demandée à l'utilisateur par le biais de votre boîte de dialogue à réussi ou échoué (dans le cas d'une boîte modale). `QDialog::exec()` renvoie un code d'erreur qui est soit `QDialog::Accepted` , soit `QDialog::Rejected` (ou encore une de vos valeurs propres dans des cas peu courants). La première valeur signifie un succès, tandis que la seconde indique soit un échec, soit un refus ou une annulation.

Du côté de votre boîte de dialogue, vous pouvez renvoyer ces valeurs de diverses façons :

- `QDialog::accept()` : `QDialog::Accepted` sera la valeur renvoyée par `exec()`
- `QDialog::reject()` : `QDialog::Rejected` sera renvoyée
- `QDialog::done(int)` : vous permet de spécifier l'une des 2 valeurs, mais aussi de spécifier la votre si vous en avez un réel besoin (utilisation assez rare tout de même).

IV-3 - Une boîte de dialogue...



La classe définie ici sera celle que nous allons afficher dans les 2 exemples suivant. Elle n'est composée que d'un bouton "Fermer".

```
class Dialog : public QDialog
{
public:
    Dialog(QWidget *parent=0)
        :QDialog(parent)
    {
        // Elle est simplement composé d'un bouton "Fermer"
```

```
QPushButton *closeBtn = new QPushButton("Fermer", this);  
  
// lequel ferme la fenetre lorsqu'on clic dessus  
connect(closeBtn, SIGNAL(clicked()), this, SLOT(accept()));  
}  
};
```

V - Ouverture de fenêtre avec des boutons

V-1 - Classes abordées

-  **QDialog** : sert de classe de base aux boîtes de dialogue
-  **QPushButton** : déclenche l'affichage des boîtes de dialogue, et quitte l'application

V-2 - Description de la solution

Bien que rarement utilisée/utilisable à cause du "clicodrome" qui risque d'être engendré (affichage d'une boîte de dialogue qui en appelle une autre qui... :)), cette méthode permet d'illustrer simplement l'ouverture d'une boîte de dialogue. Ceci dit, il peut toujours y avoir besoin d'une telle utilisation, ne serait-ce que parce que votre widget principal ne possède pas de menu.

V-3 - Exemple

Voici la classe principale. C'est à partir de celle-ci que nous allons demander l'ouverture d'une instance de *Dialog* .

```
class MainDialog : public QDialog
{
    Q_OBJECT

public:
    MainDialog(QWidget *parent=0)
        :QDialog(parent), dlg(0)
    {
        dlg = new Dialog(this);

        // Ce bouton nous permettra d'afficher notre boîte de dialogue
        QPushButton *modalDlgBtn = new QPushButton(" Open Modal dialog", this);

        // J'agrandis la fenetre pour que dlg soit visuellement différenciable de celle-ci
        resize(250,50);





        // On demande à Qt d'exécuter showDialogModal() lorsque le bouton est cliqué
        connect(modalDlgBtn, SIGNAL(clicked()), this, SLOT(showDialogModal()));
    }

private slots:
    void showDialogModal()
    {
        // on affiche la boîte de dialogue de façon modale
        dlg->setModal(true);
        dlg->show();
    }

private:
    Dialog *dlg;
};
```


VI - Utilisation d'une fenêtre principale avec menus pour ouvrir une boîte de dialogue et une boîte "A propos..."

VI-1 - Classes abordées

-  **QMainWindow** : nous en dériverons pour créer une fenêtre principale avec barre de menu
-  **QDialog** : sert de classe de base à notre boîte de dialogue
-  **QMessageBox** : affichage d'un message "A propos..."
-  **QAction** : permet de centraliser et coordonner le comportement et l'apparence entre des choix de menus ou des boutons sur une barre d'outils (chose que nous ne verrons pas ici)

VI-2 - Description de la solution

Nous allons utiliser ici un menu pour permettre d'ouvrir une boîte de dialogue de façon modale, non modale, et enfin une description de ce petit code. La classe **QAction** permet d'ajouter des fonctionnalités à un menu, ainsi qu'à d'éventuelles barre d'outils.

VI-3 - Exemple

C'est à partir de l'interface décrite ci-dessous que nous allons demander l'ouverture d'une instance de **Dialog**, que ce soit de façon modale ou non.

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent=0)
    : QMainWindow(parent), dlg(0)
    {
        // Nous creons l'instance de dialogue que nous voulons afficher
        dlg = new Dialog(this);

        // Les actions correspondront aux items de menus
        QAction *modelessAct = new QAction("Dialogue non modale", this);
        QAction *modalAct = new QAction("Afficher dialogue modale", this);
        QAction *aboutAct = new QAction("A propos...", this);
        QAction *closeAct = new QAction("Quitter", this);

        // Nous créons le menu proprement dit. Nous y ajoutons les actions précédemment créées
        QMenu *menu = menuBar()->addMenu("Divers...");
        menu->addAction(modelessAct);
        menu->addAction(modalAct);
        menu->addAction(aboutAct);
        menu->addAction(closeAct);

        // Définition des comportements des actions
        connect(modelessAct, SIGNAL(triggered()), this, SLOT(showDialogModeless()));
        connect(modalAct, SIGNAL(triggered()), this, SLOT(showDialogModal()));
        connect(aboutAct, SIGNAL(triggered()), this, SLOT(about()));
        connect(closeAct, SIGNAL(triggered()), qApp, SLOT(quit()));
    }

private slots:
    // Affichée de façon modeless, l'utilisateur peut continuer d'interagir avec le reste de
    // l'application
```



```
void swapShowDialogModeless()  
{  
    // Si elle est déjà affichée, on cache la boîte de dialogue sinon on l'affiche  
    dlg->setVisible(!dlg->isVisible());  
}  
  
// En version modale, une boîte de dialogue empêche l'utilisateur d'intervenir sur une autre  
partie  
// de l'interface que la boîte de dialogue  
void showDialogModal()  
{  
    // On s'assure que notre boîte de dialogue n'est pas déjà affichée de façon non modale  
    if(dlg->isVisible())  
    {  
        QMessageBox::critical(this, "Erreur", "La boîte de dialogue est déjà ouverte. Veuillez la fermer  
pour l'ouvrir à nouveau.");  
    }  
    else  
    {  
        dlg->exec();  
    }  
}  
  
// Affichage d'information à propos de ce "logiciel"  
void about()  
{  
    QMessageBox::about(this, "Tuto: Menus et dialogues",  
        "Cette application est destinée à illustrer:\n" \  
        "> l'ouverture de boîte de dialogue\n" \  
        "> l'ouverture d'une fenêtre \"A propos...\"\n" \  
        "> et une rapide introduction à QAction");  
}  
  
private:  
    Dialog *dlg;  
};
```

VII - Archive

Vous trouverez **dans cette archive** 2 sous répertoires :

- bouton : exemple de la première partie
- menu : exemple de la seconde

VIII - Conclusion

L'article original se situe  [ici](#) . Pour toute question, n'hésitez pas à vous rendre sur  [le forum dédié à Qt sur Developpez](#) .

