

HLIN408 : Qt - Gui en C++

Félix Costa et Julie Cailler

Université de Montpellier

20 Janvier 2017



Introduction

- Qt est un framework (ensemble de bibliothèques) permettant, notamment, de créer des interfaces graphiques (GUI) en C++.
- Son utilisation ne se limite pas à cela, mais seule cette application sera abordée ici.
- Qt est multi-plateforme et sous licence LGPL et est utilisée par Google, la NASA, ...



Figure – Logo de Qt

Code minimal

Le code minimal pour créer une fenêtre avec Qt est le suivant :
Il faut créer un objet de type QApplication. La méthode exec() lancera l'interface graphique jusqu'à sa fermeture.

```
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    return app.exec();
}
```

Table des matières

1 Widgets



Table des matières

- 1 Widgets
- 2 Signaux et slots



Table des matières

- ❶ Widgets
- ❷ Signaux et slots
- ❸ **Layouts**



Table des matières

- ❶ Widgets
- ❷ Signaux et slots
- ❸ Layouts
- ❹ Pour aller plus loin



Table des matières

- ❶ Widgets
- ❷ Signaux et slots
- ❸ Layouts
- ❹ Pour aller plus loin
- ❺ Conclusion



Table des matières

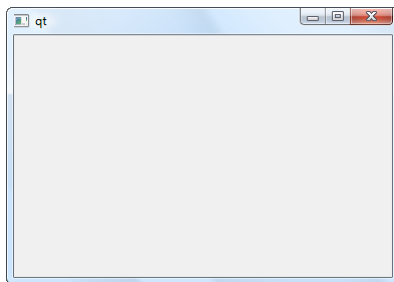
- 1 Widgets
 - Présentation
 - Exemple
 - Exemple - suite
 - Widgets et héritage
 - Illustration
- 2 Signaux et slots
- 3 Layouts
- 4 Pour aller plus loin
- 5 Conclusion

Un Widget, qu'est-ce que c'est ?

Le widget est l'élément de base composant l'interface.

Ils peuvent s'emboîter les uns dans les autres.

Un widget qui n'a pas de parent est appelé une fenêtre.



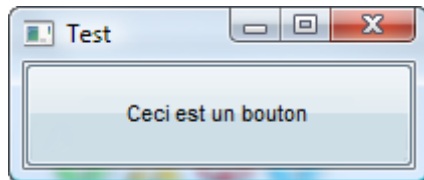
Exemple :

Par exemple, pour créer des boutons, il faut `#include <QPushButton>`.

```
1  #include <QApplication>
2  #include <QPushButton>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication app(argc, argv);
7
8      QPushButton bouton("Ceci est un bouton ");
9      bouton.show();
10
11     return app.exec();
12 }
```

Exemple :

Par exemple, pour créer des boutons, il faut `#include <QPushButton>`.



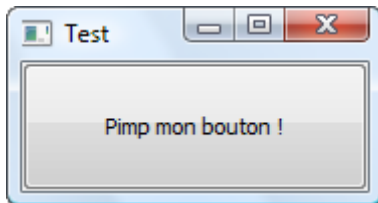
Exemple :

On peut modifier le texte grâce à la méthode bouton.setText()

```
1 #include <QApplication>
2 #include <QPushButton>
3
4
5 int main(int argc, char *argv[])
6 {
7     QApplication app(argc, argv);
8
9     QPushButton bouton("Ceci est un bouton ");
10    bouton.setText("Pimp mon bouton !");
11
12    bouton.show();
13
14    return app.exec();
15 }
```

Exemple :

On peut modifier le texte grâce à la méthode `bouton.setText()`





Un peu plus loin avec les Widgets...

Les widgets sont tous des classes qui héritent de classes-mères.

La classe-parente est QObject.

Dans une interface graphique, tout est considéré comme un widget, ce qui implique que tous les widgets héritent de QWidget, qui est une classe-fille de QObject.

Pour inclure tous les headers nécessaires aux GUI, il faut faire `#include <QWidgets>`.

Graphique

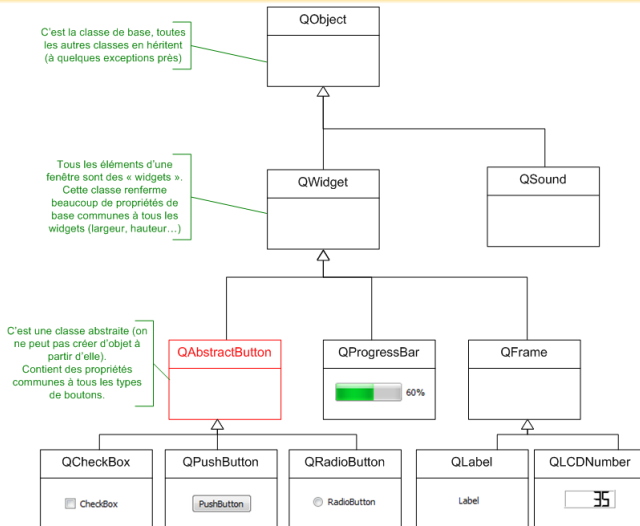




Table des matières

1 Widgets

2 Signaux et slots

- Signaux et slots I
- Exemple
- Signaux et slots II
- Exemple
- Méthode connect()
- Exemple

3 Layouts

4 Pour aller plus loin

5 Conclusion

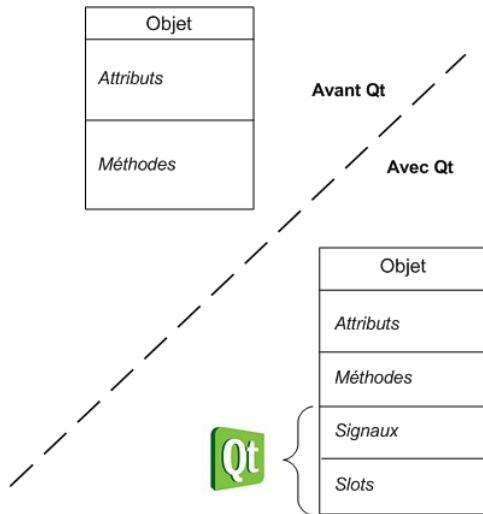


Signaux et slots I

Les signaux et les slots sont un concept développé par Qt afin de dynamiser l'interactivité avec la fenêtre.

- Les signaux : ce sont des messages envoyés par un widget lorsqu'un évènement se produit.
- Les slots : ce sont les fonctions appelées par les signaux.

Exemple

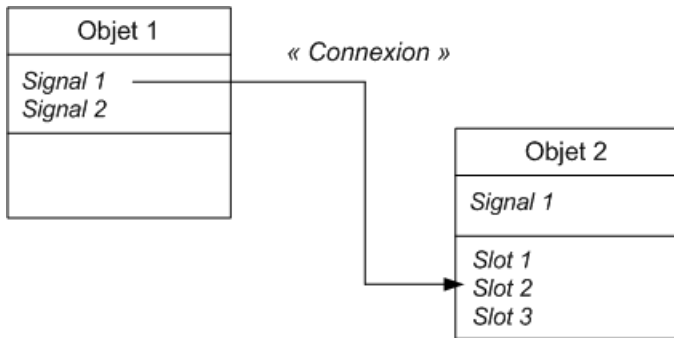




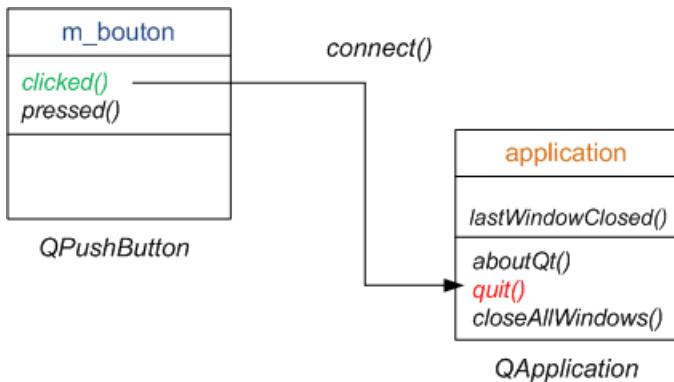
Signaux et slots II

Concrètement, ce système permet de relier des boutons à des méthodes, afin de rendre la fenêtre dynamique en utilisant la méthode `QObject : :connect()`.

Exemple :



Exemple :





La méthode connect()

La méthode connect() est une méthode statique qui permet de relier des signaux à des slots. Elle prend 4 arguments :

- un pointeur vers l'objet qui émet le signal ;
- le nom du signal que l'on souhaite « intercepter » ;
- un pointeur vers l'objet qui contient le slot récepteur ;
- le nom du slot qui doit s'exécuter lorsque le signal se produit.

Exemple :

```
1 #include "MaFenetre.h"
2
3 MaFenetre::MaFenetre() : QWidget()
4 {
5     setFixedSize(300, 150);
6
7     m_bouton = new QPushButton("Quitter", this);
8     m_bouton->setFont(QFont("Comic Sans MS", 14));
9     m_bouton->move(110, 50);
10
11     // Connexion du clic du bouton à la fermeture de l'application
12     QObject::connect(m_bouton, SIGNAL(clicked()), qApp, SLOT(quit()));
13 }
```


Exemple :

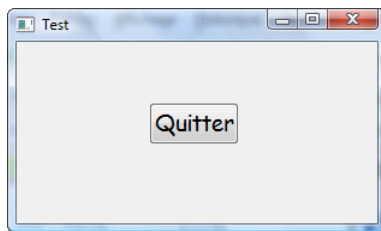




Table des matières

1 Widgets

2 Signaux et slots

3 Layouts

- Description
- Exemple
- Les différents types de Layout
- Exemple
- Layout
- Exemple

4 Pour aller plus loin

5 Conclusion



Le Layout, c'est quoi ?

Les layouts permettent de positionner les différents éléments (widgets) sur la fenêtre de manière relative.

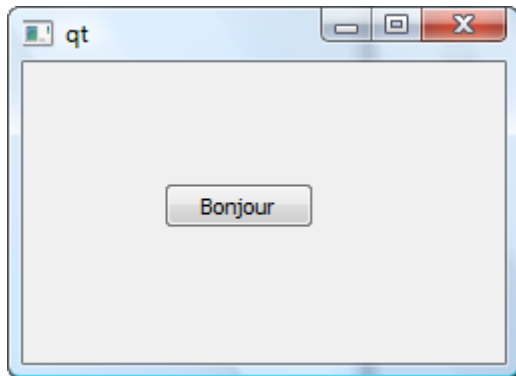
Il existe deux types de positionnements :

- Absolu : précis au pixel près.
- Relatif : plus souple, se positionne par rapport aux autres widgets et à la fenêtre.

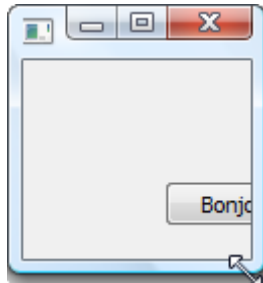
Exemple :

```
1  #include <QApplication>
2  #include <QPushButton>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication app(argc, argv);
7
8      QWidget fenetre;
9
10     QPushButton bouton("Bonjour", &fenetre);
11     bouton.move(70, 60);
12
13     fenetre.show();
14
15     return app.exec();
16 }
```

Exemple :



Exemple :



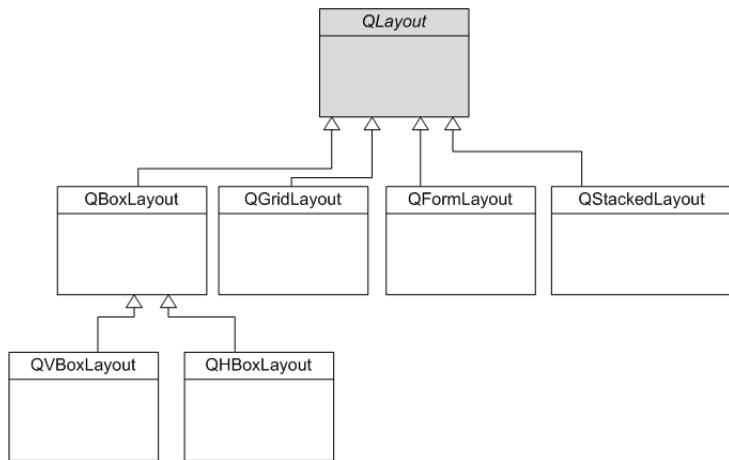


Les différents types de Layouts

Il existe plusieurs types de layout, selon ce qu'on veut faire. Voici les classes principales :

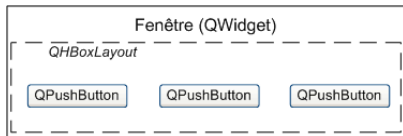
- QVBoxLayout ;
- QHBoxLayout ;
- QVBoxLayout ;
- QGridLayout ;
- QFormLayout ;
- QStackedLayout.

Illustration

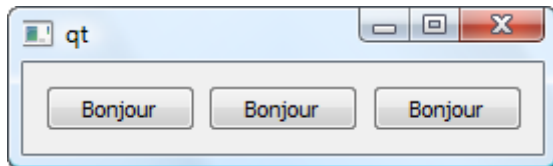


Horizontal, vertical, et plus encore !

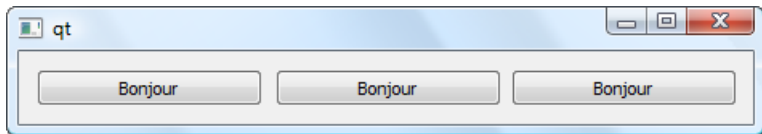
Pour placer des widgets à l'horizontale ou à la verticale les uns par rapport aux autres, il suffit de créer un objet `QHBoxLayout` (resp. `QVBoxLayout`) et d'ajouter les widgets. Ils se mettront naturellement les uns à la suite des autres dans l'ordre voulu. Le principe est le même pour les autres placements.



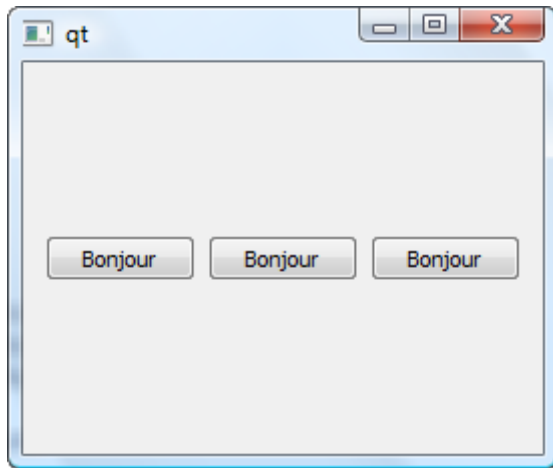
Exemple : Layout Horizontal



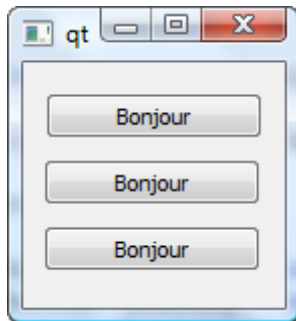
Exemple : Layout Horizontal



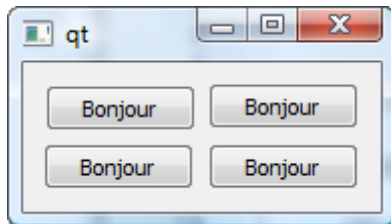
Exemple : Layout Horizontal



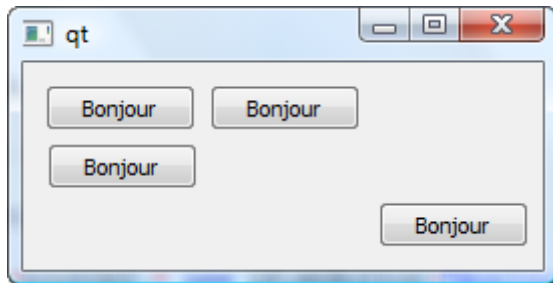
Exemple : Layout Vertical



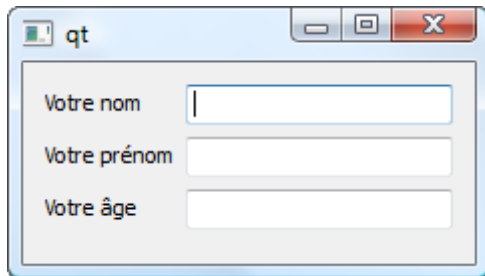
Exemple : Layout Grille



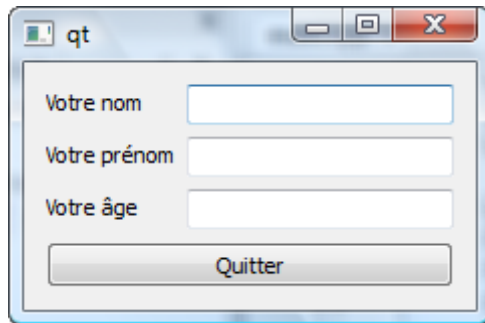
Exemple : Layout Grille avec espaces



Exemple : Layout Vertical entrée de texte



Exemple : Combinaison de Layouts



Exemple : Combinaison de Layouts


Fenêtre (QWidget)

QVBoxLayout

QFormLayout

Votre prénom :

Votre nom :

Votre âge : 

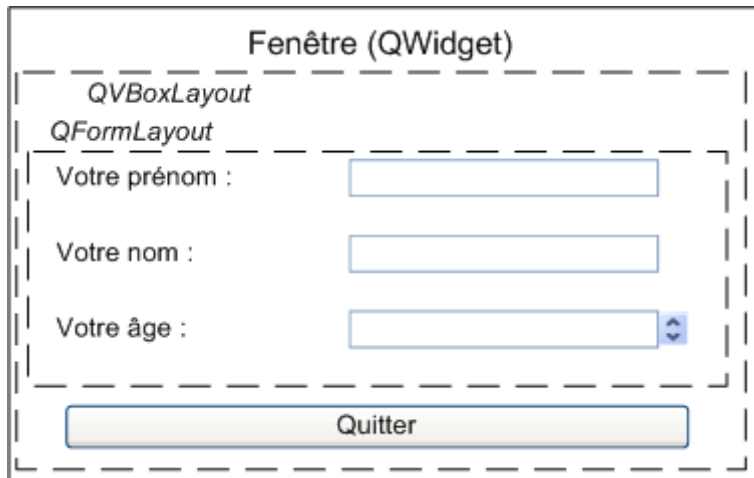




Table des matières

- 1 Widgets
- 2 Signaux et slots
- 3 Layouts
- 4 Pour aller plus loin**
 - Boîtes de dialogue usuelles
 - Exemple
- 5 Conclusion



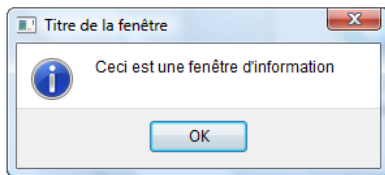
Boîtes de dialogue usuelles

Les boîtes de dialogue usuelles sont des classes/fenêtres à part créées car utilisées régulièrement. Elles permettent de remplir des fonctions bien précises :

- Ouvrir un fichier
- Enregistrer un fichier
- Choisir une couleur
- Afficher un message
- Rechercher un texte (CTRL-F)
- Sélectionner une police

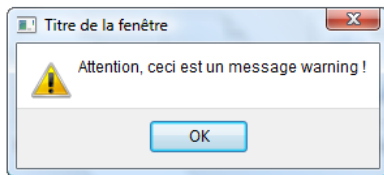
Exemple : information

```
1 QMessageBox::information(this, "Titre de la fenêtre", "Ceci est une fenêtre  
d'information");
```



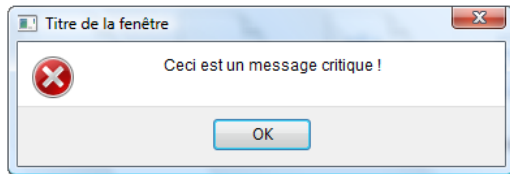
Exemple : warning

```
1 QMessageBox::warning(this, "Titre de la fenêtre", "Attention, ceci est un message  
warning!");
```



Exemple : critique

```
1 QMessageBox::critical(this, "Titre de la fenêtre", "Ceci est un message critique!");
```



Exemple : question

```
1 QMessageBox::question(this, "Voici une boite de dialogue usuelle qui  
  permet de poser une question. C'est facile non ?", QMessageBox::Yes |  
  QMessageBox::No);
```

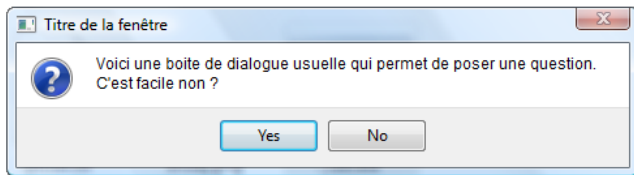




Table des matières

- 1 Widgets
- 2 Signaux et slots
- 3 Layouts
- 4 Pour aller plus loin
- 5 Conclusion**



Conclusion

Avantages :

- Excellente documentation
- Nom des fonctions est explicite
- Qt est multiplateforme : Windows, Linux et Mac.
- Des bindings permettent d'utiliser Qt avec d'autres langages que le C++ (Java, OCaml, Perl, PHP, Python, Ruby, Scheme)

Inconvénients :

- Il faut savoir coder !
- Il existe des frameworks avec plus de widgets
- Il y a des frameworks plus complets si on veut coder sur un OS bien précis