Développement Qt

Les signaux et les slots

Les signaux et les slots forment un **mécanisme de communication entre objets Qt** :

- ➔ Le **signal** représente un **événement** (*event*)
- ➔ Le **slot** représente le **gestionnaire d'événement** (*handler*)

Toutes les classes qui héritent de **`QObject`** ou d'une de ses sous-classes (par exemple **`QWidget`**) **peuvent contenir des signaux et des slots**. Il faut aussi ajouter la macro **`Q_OBJECT`** au sein de cette classe.

Un **signal** se déclare comme une méthode mais il n'aura pas définition. On utilisera un **nom** pour préciser l'événement qu'il représente.

Un **slot** est une méthode (il se déclare et se définit). On utilisera un **verbe** pour préciser l'action qui sera déclenchée en réponse à un signal.

L'association d'un signal à un slot est réalisée par une connexion avec l'appel **`connect()`** (une méthode statique de la classe **`QObject`**).

L'association d'un signal à un slot peut être supprimée par la méthode **`disconnect()`** (une méthode statique de la classe **`QObject`**).

```
bool QObject::connect(const QObject* sender,   const char* signal,
                      const QObject* receiver, const char* method,
                      Qt::ConnectionType type = Qt::AutoConnection) const
```

Exemple :

```
connect(boutonQuitter, SIGNAL(clicked()), qApp, SLOT(quit()));
```

Remarque : **`qApp`** est un pointeur global qui contient toujours l'adresse de l'objet **`QApplication`** instancié dans le **`main()`** (il suffit d'inclure **`<QApplication>`** pour pouvoir y accéder).

Les classes Qt peuvent posséder des signaux et des slots (voir la documentation des classes : https://doc.qt.io/qt-5/).

Remarque : L'émission d'un signal d'une classe Qt est "**automatique**". Par exemple lorsqu'on appui sur un bouton, le signal **`clicked()`** de la classe **`QPushButton`** est émis.

Il est possible de créer ses propres signaux et slots.

Pour déclarer un signal personnalisé, on utilise le mot clé **`signals`** dans la déclaration de la classe et il faut savoir qu'un **signal** :

- n'a pas de valeur de retour (donc **`void`**) et
- n'a pas de définition de la méthode (donc pas de corps **`{}`**)
- peut avoir des paramètres

Pour émettre un signal, on utilise la méthode **`emit`** :

```
emit nomDuSignal( parametreDuSignal );
```

Les slots personnalisés se déclarent et définissent comme des méthodes **`private`**, **`protected`** ou **`public`**. On utilise le mot clé **`slots`** dans la déclaration de la classe. Les slots n'ont pas de valeur de retour (donc **`void`**). On utilisera un **`signal`** pour émettre une valeur de retour si besoin.

Remarque : Les slots étant des méthodes normales, ils peuvent aussi être appelés directement comme toute méthode.

Exemple :

```
#include <QObject>

class MaClasse : public QObject
{
   Q_OBJECT
   public:
      MaClasse( QObject *parent=nullptr ) : QObject(parent) {}

   signals:
      void unSignal( int ); // un signal personnalisé

   public slots:
      void unSlot( int valeur ) // un slot personnalisé
      {}
};
```

Remarque : Le mécanisme des signaux et slots fournit un contrôle des types. La signature d'un signal doit correspondre à la signature du slot récepteur (en réalité, un slot peut avoir une signature plus courte que celle du signal qu'il reçoit car il peut ignorer les arguments en trop).

Particularités :

- Un *signal* peut être connecté à un autre *signal*. Dans ce cas, lorsque le premier *signal* est émis, il entraîne l'émission du second *signal*.

- Un *signal* peut être connecté à plusieurs *slots*. Attention : les *slots* seront activés dans un ordre arbitraire.

- Plusieurs *signaux* peuvent être connectés à un seul *slot*.

Les **signaux et les slots sont faiblement couplés** : un objet qui émet un *signal* ne sait pas (et ne se soucie pas de) quel(s) *slot(s)* vont recevoir ce *signal*. De la même façon, le *slot* d'un objet interceptant un *signal* ne sait pas quel objet a émis le *signal*.

Remarque : il est tout de même possible pour le *slot* d'appeler la méthode **`sender()`** (de la classe **`QObject`**) pour obtenir l'adresse de l'objet qui a émis le *signal* et donc déclenché le *slot*.

```
void MaClasse::unSlot()
{
   QObject* objet = sender();

   // on peut tester si l'adresse correspond à un objet membre (attribut)
   if( objet == boutonOuvrir )
   {
      ...
   }

   // ou :
   // on peut "tenter" un cast (transtypage descendant ou downcast)
   QPushButton* bouton = qobject_cast<QPushButton*>(sender());
   if( bouton != nullptr )
   {
      ...
   }
}
```

Utiliser ses propres types (classes, structures, …) dans les paramètres des signaux et slots :

Contraintes :

- il faut un constructeur par défaut, un constructeur de copie, un destructeur
- il faut déclarer le type avec Q_DECLARE_METATYPE(MonType)

Exemple :

- **Mesure.h**

```
class Mesure : public QObject
{
   Q_OBJECT
public:
   explicit Mesure(QObject *parent = 0, double temperature = 0., int humidite = 0,
int luminosite = 0);
   Mesure(const Mesure &mesure);
   virtual ~Mesure();
private:
   double temperature;
   int humidite;
   int luminosite;
};
```

```
Q_DECLARE_METATYPE(Mesure)
```

- **Mesure.cpp**

```
Mesure::Mesure(QObject *parent, double temperature, int humidite, int luminosite)
: QObject(parent), temperature(temperature), humidite(humidite),
luminosite(luminosite)
{
}

Mesure::Mesure(const Mesure &m) : QObject(m.parent()), temperature(m.temperature),
humidite(m.humidite), luminosite(m.luminosite)
{
}

Mesure::~Mesure()
{
}
```

Il faut ensuite enregistrer le type avec qRegisterMetaType<MonType>() avant de faire une connexion signal/slot.

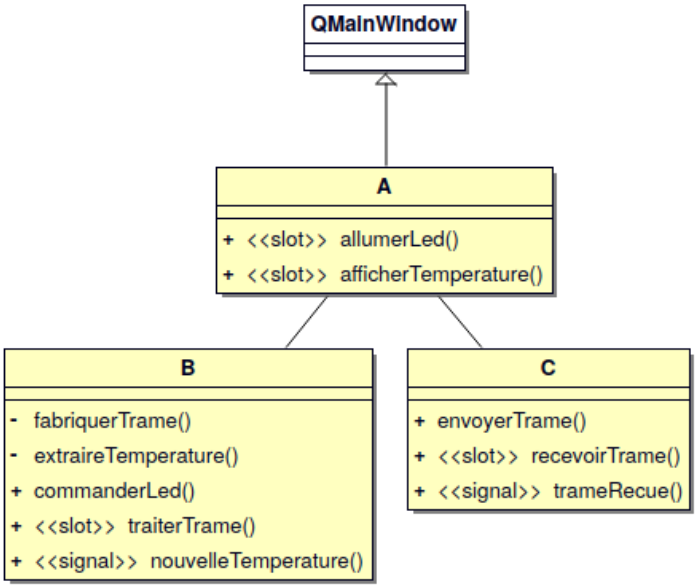
Exemple :

```
qRegisterMetaType<Mesure>();
connect(sonde, SIGNAL(nouvelleMesure(Mesure)),
        this,  SLOT(actualiserMesure(Mesure)));
```

UML :

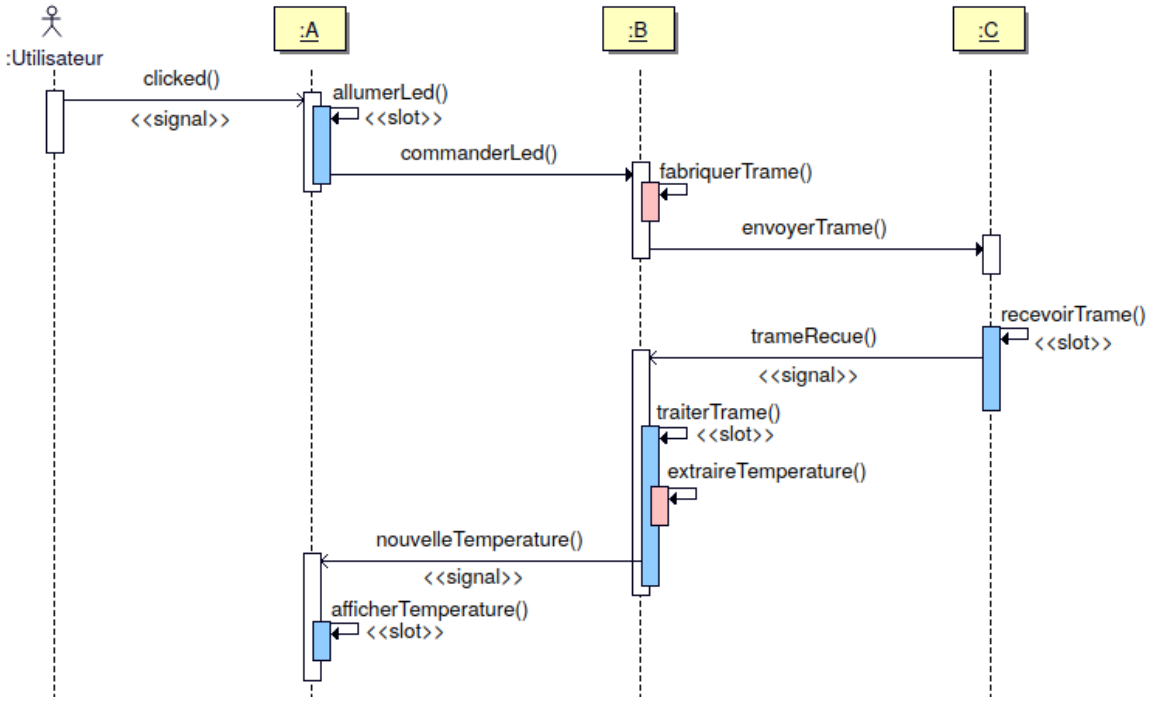
Les signaux et les slots n'existent pas en **UML**. On utilisera alors la notion de « **stéréotype** ».

On fera apparaître les stéréotypes **« signal »** et **« slot »** dans le diagramme de classes :



Remarque : ceci est un EXEMPLE. Les classes n'ont pas été nommées car aucune analyse n'a été réalisée !

On fera de même dans un diagramme de séquence :



Rappel : Les **signaux et les slots sont faiblement couplés** ! Ici, un objet de la classe **C** émet un *signal* qui déclenche un *slot* dans un objet de la classe **B**. Pourtant, les classes **B** et **C** n'ont pas de relation. Dans cet exemple, c'est la classe **A** qui a assuré la connexion *signal/slot*.