Fiche – Signal / Slot

I - Présentation

Un **événement** correspond généralement à une **action utilisateur** (clic souris, frappe clavier) ou à une **action d'un autre objet** ou **du système**. Ces évènements ont pour objectif de lancer **l'exécution d'un sous-programme** ou du moins d'une portion de code (fonction/méthode).

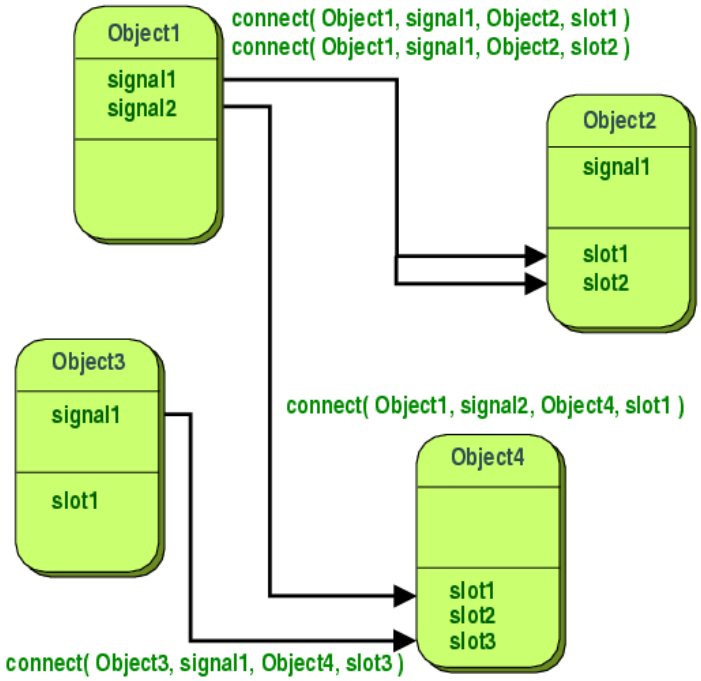
Sous Qt, les **événements** sont appelés des **signaux** et les **méthodes** (appelées par un signal) s'appellent des **slots**. On parle donc avec **Qt de gestion Signal/Slot**.

Beaucoup de classes Qt contiennent des signaux et des slots. Il suffit de regarder la doc pour les connaitre mais attention, **certains signaux/slots sont écrits dans des classes mère** et donc ne sont pas forcément visibles dans les classes dérivées.

La liaison entre une signal et un slot en Qt se fait avec la méthode connect(....) de la classe QObject.
Il y a 2 méthodes (une normale et une statique). Nous utiliserons la **méthode statique** qui a l'avantage de pouvoir **être appelée sans avoir à créer d'objet** de type QObject.
Voici le prototype de la méthode :

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

sender : adresse de l'objet émetteur du signal
signal : le signal émis
receiver : adresse de l'objet récepteur du signal
method : méthode (slot) à appeler
Qt::ConnectionType : Le fonctionnement par défaut suffira pour le moment.



II - Exemple

```
.....
QSlider slider;
QProgressBar bar;
QObject::connect(&slider,SIGNAL(valueChanged(int),&bar,SLOT(setValue(int)));
.....
```

Cet exemple montre la connexion d'un signal de la classe QSlider avec un slot de la classe QProgressBar.

Vous remarquerez qu'il est nécessaire d'utiliser les **macros SIGNAL(...) et SLOT(...)** et que si le **signal a un type en paramètre alors le slot doit avoir le même type en paramètre** : le signal fournit une donnée.