第3章　SIGNAL/SLOTを使ってみよう

3.1　SIGNAL/SLOTの基本的な使い方

Qt で最も良く使い、Qt の特徴でもある機能がシグナル/スロット (Signal and Slot) です。

シグナルは、例えばボタン (QPushButton) が押された時には clicked という関数が呼ばれます。この clicked 関数がシグナルとなります。そしてシグナルが発生した時には SLOT で指定した関数が呼び出されます。例えば、SLOT に quit 関数 (この関数はプログラムを終了する時に呼び出す) を指定しておけば、ボタンを押した後、プログラムが終了します。わかりにくいと思うので、例を挙げて説明します。

次のコードを見てください。

```
1   #include <QApplication>
2   #include <QPushButton>
3
4   int main(int argc, char** argv)
5   {
6           QApplication app(argc, argv);
7           QPushButton* button = new QPushButton("Quit");
8           QObject::connect(button, SIGNAL( clicked() ),
9                            &app, SLOT(quit()) );
10          button->show();
11          return app.exec();
12  }
```

このコードを実行すると、図 3.1 のようになります。

connect 関数の第一引数はシグナル (信号) が発生する部品のアドレスを渡します。例の場合、Quit と表示されている button をセットしています。そして、第 2 引数でシグナルとなる関数を指定します。関数をシグナルに設定する場合、SIGNAL(

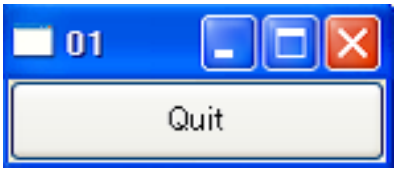

図 3.1: 実行結果 (on Windows)

) のカッコで囲った中に関数を書きます。QPushButton の場合、ボタンがクリックされると clicked 関数が呼び出されます。この clicked 関数がシグナルとなります

そして第 3,4 引数で発生したシグナルを app に結び付けています。

第 3 引数にはスロット側の部品のアドレスを渡します。そして第 4 引数として SLOT( ) のカッコで囲った中に書いた関数をスロットとなる関数と設定します。QApplication クラスの関数である quit 関数はプログラムを終了する時に使います。

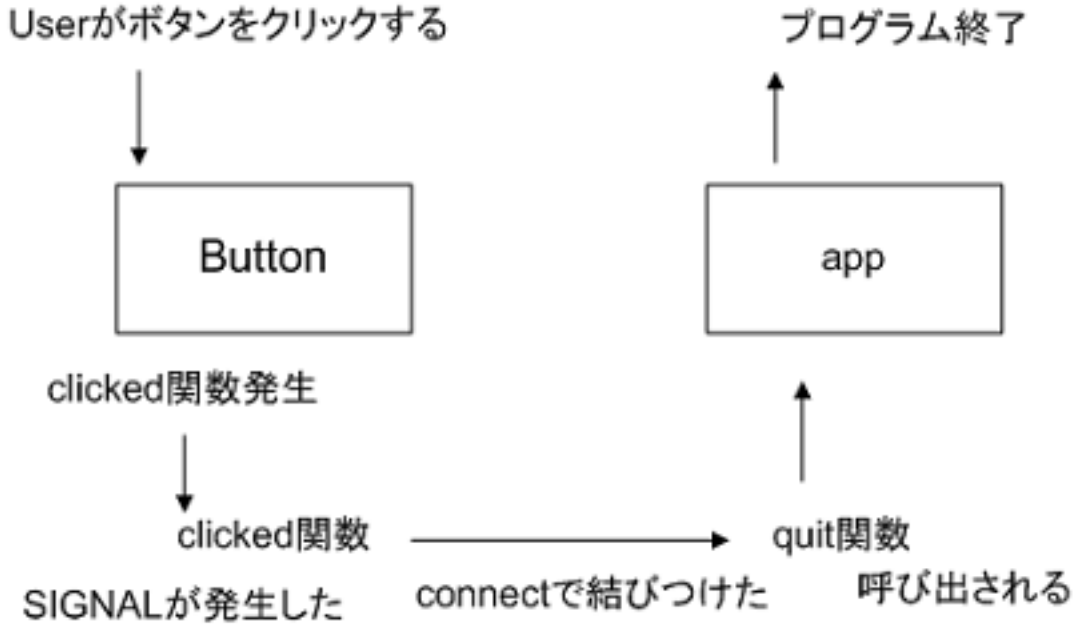

図 3.2: Signal/Slot のイメージ図

3.2 複数の Slot について

次は図 3.3 のようなウィンドウを作ります。

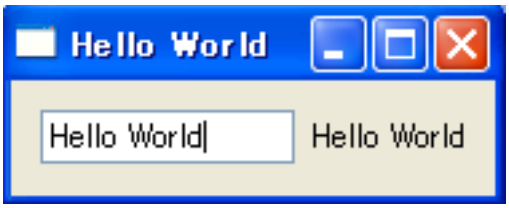

図 3.3: 実行結果 (on Windows)

QLineEdit と QLabel という部品を使い、Signal を textChanged 関数 (QLineEdit が編集されたら発生) とします。そして今回は Slot を複数設定してみましょう。

Signal が発生したら、QLabel にも QLineEdit と同じ内容が表示されるようにします。QLabel に文字をセットする場合、setText 関数を使います。

また同時に Window のタイトルも QLineEdit と同じ内容を表示させてみます。Window のタイトルをセットするには setWindowTitle 関数を使います。

では次のようなコードを書いてみましょう。

```
1   #include <QApplication>
2   #include <QLabel>
3   #include <QHBoxLayout>
4   #include <QLineEdit>
5
6   int main(int argc, char** argv)
7   {
8           QApplication app(argc, argv);
9           QLabel* label = new QLabel("Hello");
10          QLineEdit* edit = new QLineEdit;
11          QWidget* window = new QWidget;
12          QHBoxLayout* layout = new QHBoxLayout;
13
14          QObject::connect(edit,SIGNAL(textChanged(QString)),
15                           label,SLOT(setText(QString)) );
16          QObject::connect(edit,SIGNAL(textChanged(QString)),
17                           window,SLOT(setWindowTitle(QString)) );
18
19          layout->addWidget(edit);
20          layout->addWidget(label);
21          window->setLayout(layout);
```

```
22          window->show();
23
24          return app.exec();
25  }
```

Signal/Slot のイメージ図は図 3.4 となります。

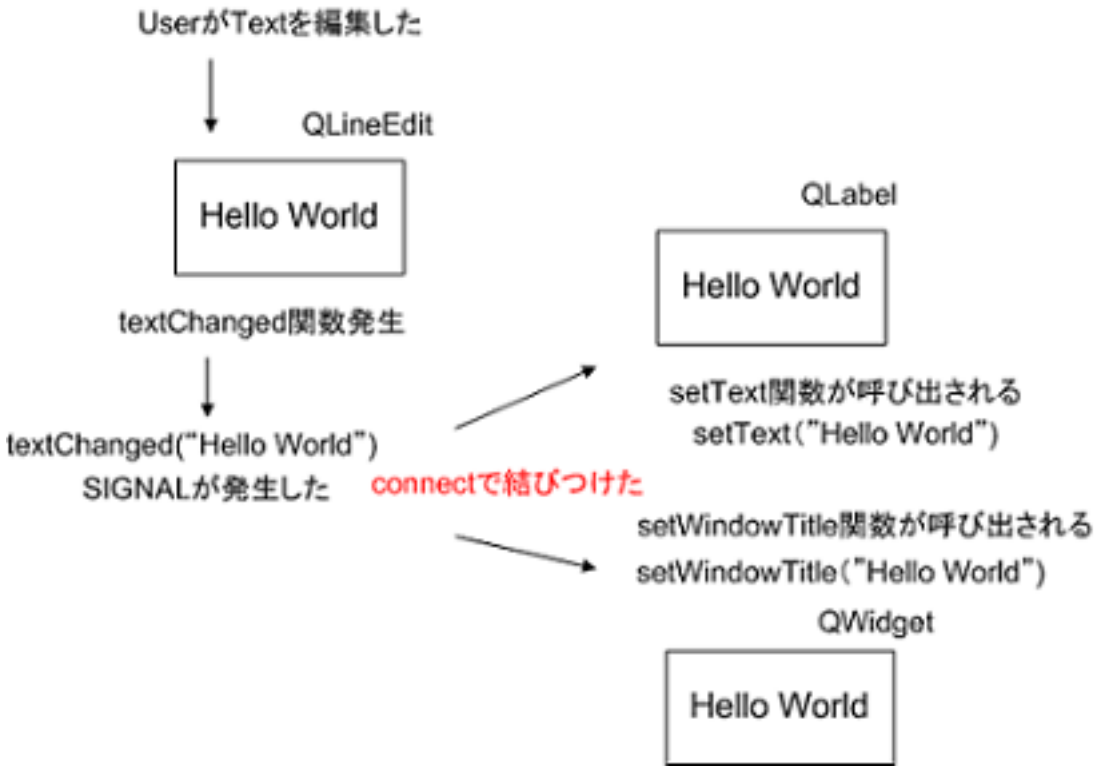

図 3.4: Signal/Slot のイメージ図

14,15 行目では edit の textChanged 関数と label の setText 関数を結び付けています。QString というのは、C++でいう string と同じです。textChanged() だけではダメです。ちゃんと引数が付いている textChanged(QString) を呼び出さなければなりません。そして、setText 関数で label に edit で編集された文字列をセットしています。

16,17 行目も同様に setWindowTitle 関数で window の Title に edit で編集された文字列をセットしています。

3.3 Connection の解除

いままで connect 関数を使い、Signal と Slot を設定していました。しかし、場合によってはセットした connect を解除したい場合があるかもしれません。そういった場合、disconnect 関数を使います。使い方は connect とほとんど同じです。

3.1 節のコードに disconnect 関数を付け加えたものが次のコードです。

```
1   #include <QApplication>
2   #include <QPushButton>
3
4   int main(int argc, char** argv)
5   {
6           QApplication app(argc, argv);
7           QPushButton* button = new QPushButton("Quit");
8           QObject::connect(button, SIGNAL( clicked() ),
9                            &app, SLOT(quit()) );
10          QObject::disconnect(button, SIGNAL( clicked() ),
11                           &app, SLOT(quit()) );
12          button->show();
13          return app.exec();
14  }
```

connect の部分が disconnect になっただけです。これでコネクションを解除することができます。もちろん、このプログラムは実行してボタンを押しても何も起こりません。

3.4 Signal/Slot のまとめ

connect 関数

```
connect(sender, SIGNAL(signal), receiver, SLOT(slot) );
```

sender：信号が発生する部品のアドレスを渡す
SIGNAL(signal)：signal に信号とする関数を渡す
receiver：信号を受け取る部品のアドレスを渡す
SLOT(slot)：信号を受け取った際に呼び出す関数を渡す

参考文献

[1] Jasmin Blanchette & Mark Summerfield, C++ GUI Programming with Qt4.

[2] Trolltech, Qt Assistant Tutorial and Examples Qt Tutorial

[3] Trolltech, Qt Assistant All Classes

[4] Trolltech, Qt Assistant Core Features Layout Management