第1章　Hello Qt!

1.1　Hello Qt!

最初は画面に *HelloQt!* と表示するプログラムを作ることにします。まずは何も考えず次のソースコードを入力し、実行できるかどうかのテストをしてみてください。

```
1   #include <QApplication>
2   #include <QLabel>
3
4   int main(int argc, char** argv)
5   {
6           QApplication app(argc, argv);
7           QLabel* label = new QLabel("Hello Qt!");
8           label->show();
9           return app.exec();
10  }
```

コンパイルを行うには Linux の場合ターミナルを実行しソースコードが置いてあるフォルダまで移動してください。その後、

qmake -project

をタイプしてください。次に

qmake

をタイプし、最後に

make

とします。これでフォルダに実行ファイルが作成されたはずです。

Windows で VisualStudio を使っている場合は、コマンドプロンプトを実行しソースコードが置いてあるフォルダに移動したあと、

qmake -project -t vcapp -o hello.pro

をタイプし、次に

qmake

を実行します。そうすると、hello.vcproj という VisualStudio のプロジェクトファイルが作成されます。このプロジェクトを実行し、コンパイルを行ってください。

プログラムを実行すると Windows 環境の場合、図 1.1 のように表示されると思います。

図 1.1: Windows で実行した Hello

では一行ずつ解説していきます。

1,2 行目の部分は、QApplication と QLabel のクラスの定義をインクルードしています。Qt を使っている GUI アプリケーションには必ず一つの QApplication のオブジェクトが無ければなりません。また、すべての Qt クラスはヘッダーファイルとクラスが同じ名前に設定されています。

6 行目は QApplication オブジェクトを作成しています。QApplication はアプリケーションに関するリソースを管理するものであり、コンストラクタには argc と argv を渡さなければなりません。このため、Qt で作成したプログラムでもコマンドライン引数を使うことが出来ます。

7 行目では Hello Qt!と表示するラベルを作成しています。そして 8 行目でラベルを表示しています。

そして 9 行目でコントロールを Qt アプリケーションに渡します。コントロールを渡した先ではイベントループが起こっています。

これでプログラムは完成です。プログラムをコンパイルして実行してみてください。うまくコンパイルができたでしょうか？

ここで、QLabel は new したのに delete していないのでメモリリークが起こりますが、プログラムが終了すればオペレーティングシステムが勝手に処理してくれるので、細かいことは気にしないでください。

1.2 ラベルについて (QLabel)

1.1 節のプログラムで QLabel を使用しました。ここで QLabel の面白い使い方を見てみましょう。前節の

```
QLabel* label = new QLabel("Hello Qt!");
```

の部分を次のコードに置き換えてコンパイル、実行してみてください。

```
QLabel* label = new QLabel("<i>Hello Qt!</i>");
```

図 1.2 のように、斜体文字となりました。

図 1.2: Windows で実行した Hello2

 という書き方は HTML で主に使用されています。このように HTML スタイルの書き方を QLabel ではサポートしています。他にも

```
QLabel* label = new QLabel("<h2><i>Hello<br>Qt!</i></h2>");
```

とすると、図 1.3 となります。

図 1.3: Linux で実行した Test2

HTML についてはここでは説明しません。興味のある方はインターネットで調べてみてください。とほほの WWW 入門 (http://www.tohoho-web.com/www.htm) なんかがお勧めです。

1.3 ボタンを作ってみよう (QPushButton)

前節ではラベルを作ったので、今度はボタンを作ってみましょう。次のようなコードを書いたとします。

```
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("Hello Qt!");
8           button->resize(200,50);
9           button->move(100,50);
10          button->show();
11          return app.exec();
12  }
```

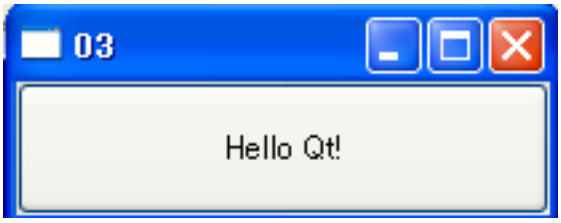

図 1.4: QPushButton の例 (on Windows)

このコードの 2 行目及び 7 行目は、1.1 節の QLabel の部分が QPushButton に変わっただけです。このようにすることでボタンを作成することができます。

8 行目、9 行目は今回始めて出てきた関数です。resize 関数は、ボタンの大きさを変更するものです。この場合は、高さ 200Pixels, 幅 50Pixels となります。

move 関数は、ウィンドウ（この場合ボタン）がデスクトップ上に表示される位置を決めるものです。関数は move(x 座標, y 座標) となっており、デスクトップ画面の一番左端が (0,0) となっています。x 座標はデスクトップの左から右にいくに従い値が増加し、y 座標は上から下に行くに従い値が増加します。

これら resize, move 関数は特に設定しなくても問題ありません。また、これらの関数は QPushButton のみならず、QLabel などの部品でも使うことができます。

QPushButton は QLabel とは違い HTML のタグを使うことはできません。

さて、このプログラムの実行結果は図 1.4 となります。

ここでせっかくボタンを作ったのだから、例えば、ボタンをクリックしたらプログラムが終了するなど何かアクションを起こしたいと思うかもしれません。けど、もう少し待ってください。この事については第 3 章で取り扱います。まずは Widget[1]の基本的な使い方を勉強してしまいましょう。

1.4 フォントを変えてみよう (QFont)

前節でボタン (QPushButton) を作りましたが、もしかしたら文字フォントを変えてみたいと思ったかもしれません。(そんなことは無い？)

Qt ではフォントを変えるのも簡単です。前節のプログラムに次のコードを書き加えるだけです。

```
#include <QFont>

 button->setFont( QFont("Times", 15, QFont::Bold) );
```

QFont を使う場合は、QFont をインクルードしなければなりません。

QFont のコンストラクタは、

```
  QFont(const QString & family, int pointSize = -1,
int weight = -1, bool italic = false );
```

と定義されています (他にもありますが省略します)。

まずは、setFont 関数ですが、この関数に QFont クラスのオブジェクトを渡してあげることでフォントの変更ができます。

QFont のコンストラクタの第 1 引数の引数はフォントの名前を指定してあげます。例では Times というフォントを使うことにしました。他にも System や Windows の場合だと Tahoma なんていうフォントもあります。いろいろ試してみてください。

第 2 引数にはフォントの大きさを指定します。

第 3 引数にはフォントの幅を指定します。例ではフォントの幅を QFont::Bold としました。これは 75 という数字に置き換えられます。ボールド文字 (太い文字) にしたい時には QFont::Bold を使えばよいでしょう。他にも QFont::Normal(50 に置き換えられる) や QFont::Light(25 に置き換えられる) などがあります。0〜99 の値の範囲ならどんな値でもよいです。

[1]Window と Gadget を混ぜた言葉で、日本語に直すとウィンドウの部品ということです。QPushButton や QLabel などが部品に当たります。

第 4 引数は bool 型となっており、イタリック体（斜体）にしたい場合はこの引数に true を与えてあげればよいです。

図 1.5 は `button->setFont( QFont("Times", 15, QFont::Bold , true) );` を追加した場合の実行例です。

図 1.5: QFont の例 (on Windows)

1.5 とりあえず日本語を表示してみよう

もしかしたらもうお気づきの方もいるかもしれません。今までボタンやラベルのキャプションには英語しか使ってきませんでした。なぜ日本語を使わなかったのかというと、実は日本語の扱いは厄介なものだからです。次のようなコードを実行したとしましょう。これは 1.1 節のコードの 7 行目を変えただけです。

```
1   #include <QApplication>
2   #include <QLabel>
3
4   int main(int argc, char** argv)
5   {
6           QApplication app(argc, argv);
7           QLabel* label = new QLabel("こんにちは Qt");
8           label->show();
9           return app.exec();
10  }
```

この実行結果は図 1.6 となります。

なんと文字化けしてしまっています。Linux と Windows どちらも使ったことのある方なら、文字コードの問題は厄介なものだと知っていると思います。

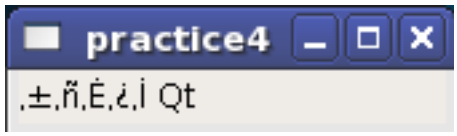

図 1.6: 日本語の表示 (on Linux)

Windows の場合、使われている文字コードは大体が Shift-JIS です。Unix 系の場合は EUC-JP や UTF-8 などでしょうか。この文字コードの違いにより文字化けが起こってしまいます。

この問題を回避するためには、Qt に自分が使っている文字コードを教えてあげる必要があります。その方法は、次のようなコードになります。

```
1   #include <QApplication>
2   #include <QLabel>
3   #include <QTextCodec>
4   #include <QString>
5
6   int main(int argc, char** argv)
7   {
8           QApplication app(argc, argv);
9           QTextCodec::setCodecForTr(QTextCodec::codecForLocale());
10          QLabel* label = new QLabel(QObject::tr("こんにちは Qt"));
11          label->show();
12          return app.exec();
13  }
```

実行結果は、図 1.7 となります。

(a)on Linux (b)on Windows

図 1.7: 日本語の表示

重要なのは、9 行目と 10 行目です。9 行目の

```
QTextCodec::setCodecForTr(QTextCodec::codecForLocale());
```

という部分で文字コードを指定しています。

setCodecForTr 関数の引数には、QTextCodec 型の値を渡します。

例では、QTextCodec::codecForLocale() を渡していますが、この関数はシステムで使われている標準的な文字コードを返します。よって自分が使っている環境にあった文字コードを指定しているので大体の場合はこの方法で問題ないと思います。

ただ、人によっては文字コードを指定したい場合があるかもしれません。その場合、

```
QTextCodec::setCodecForTr(QTextCodec::codecForName("Shift-JIS")
);
```

とします。codecForName 関数の中に直接使用したい文字コードを書き込みます。

10 行目は QObject::tr 関数を使用しています。この関数の中に文字を書く事により文字コードが正常に変換されます。また、tr 関数は後に自分の作ったプログラムを国際化したい場合にも使われます。

さて、とりあえず日本語は使えるようになりました。しかしながら、かなり説明が雑だと思います。このあたりの説明は後の章でもう一度詳しく説明することになると思いますが、今はこういうものだと思ってください。

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