Jconsole用GUI環境としてのXOJO

須田祐司

May 1, 2016

Contents

1.1 はじめに

J のバージョンが 6 から 7 へ上がったのは GUI 環境の変更に伴うものであった。7 では gtk が採用されが、gtk を習得する前に、バージョン 8 となり、今度は Qt が採用された。筆者は主にバージョン 6 の Windows 版で GUI を使っていたので、J の最新版である Qt の GUI フォーム編集を使いこなせない状況が続いていた。最近、XOJO という RealBASIC の後継開発プラットフォームの存在を知り、その GUI 設計の容易さから、jconsole 用の GUI 環境として活用を検討したので報告する。J Primer で GUI サンプルとして使われている摂氏華氏変換フォームを例に XOJO の使用法を説明する。XOJO は Mac、Linux、Windows 版があるが、この数年の間に筆者の開発環境が Windows から Mac へ移行しているので今回は OSX 最新版 El Capitan 環境で実験した。その後、Windows7 環境で同様の実験を行ったところ Windows 環境では Mac と同様のシェル機能が標準では使えないことが判明したので、コマンドプロンプト用のバッチファイルを設定することで解決した。今回の XOJO の GUI 環境を使うと、J で開発した関数を J6 の Form Editor と同様に自由設計のフォームを使って制御することが可能になる。

1.2 準備

1.2.1 Mac の場合

J64-804 をアプリケーションフォルダにインストールする。次に$HOME に添付 j.zip を解凍すると作業用フォルダ j が作成され実験用ファイルが展開される。

J インストール先のフォルダ
/Applications/j64-804/

実験用ファイル
$HOME/j/c_to_f.ijs
$HOME/j/f_to_c.ijs
$HOME/j/do_with_j.ijs
$HOME/j/fahrenheit.txt
$HOME/j/celsius.txt
$HOME/j/executeWithJ.sh
$HOME/j/xojo_GUI_for_Mac_jconsole_FC_shell_echo_cat_mode.xojo_binary_project
$HOME/j/xojo_GUI_for_Win32_and_Win64_jconsole_FC.xojo_binary_project

上記ファイルが揃ったところで最新の xojo for mac 2016r1 をダウンロードしてインストールすると xojo_GUI_for_Mac_jconsole_FC_shell_echo_cat_mode.xojo_binary_project が実行可能になる。

1.2.2 Windows の場合

J804_win64.exe 或いは J804_win32.exe を実行し、インストール先をデフォルトから c:\ に変更してインストールする。次に c:\ に添付 j.zip を解凍すると作業用フォルダ j が作成され実験用ファイルが展開される。

J インストール先のフォルダ
64 ビット版の場合
c:\j804_win64\j64-804
32 ビット版の場合
c:\j804_win32\j804\

実験用ファイル
c:\j\c_to_f.ijs
c:\j\f_to_c.ijs
c:\j\do_with_j.ijs
c:\j\fahrenheit.txt
c:\j\celsius.txt
c:\j\executeWithJ.sh
c:\j\xojo_GUI_for_Win32_and_Win64_jconsole_FC.xojo_binary_project
c:\j\xojo_GUI_for_Mac_jconsole_FC_shell_echo_cat_mode.xojo_binary_project

上記ファイルが揃ったところで最新の xojo for mac 2016r1 をダウンロードしてインストールすると xojo_GUI_for_Win32_and_Win64_jconsole_FC.xojo_binary_project が実行可能になる。

1.3 XOJO の GUI と Jconsole を組み合わせてアプリケーションを作る手順

1.3.1 XOJO のインストールとプログラミング

インターネット検索で XOJO のサイトへ行き、ユーザー登録（無料）をすると XOJO がダウンロード可能になる。XOJO はローカル環境で使う限り無料である。ライセンスを購入するとビルドして実行ファイルを作ることが可能になる。現在の最新版は 2016Release1 である。GUI の設計は Application Main Window へ control を drag drop して配置でき、サイズと位置は自由設計なので j6 の GUI フォームエディタと同様の操作性で設計できる。XOJO のレファレンスを見ると情報は膨大であるが、XOJO プログラミングの基本は今回作成した xojo_GUI_for_jconsole_FC を探索することで掴んで欲しい。今回の用途で必要なコントロールとしてはボタン、エディット、ラベルであり、本報告ではこの 3 種類のコントロールのみ使用している。左側に並んでいるアイテムを選択すると中央と右側の表示エリアに各アイテムの情報が表示される。それぞれのアイテムを選択した時の画面をキャプチャし添付したので、探索の助けにして欲しい。

1.3.2 XOJO と J のテキスト情報交換メカニズム

jconsole で使う関数の引数を edit control へ入力しボタンアクションでテキストファイルへ出力する (shellModule の WriteToFile メソッド)。同じくボタンアクションで XOJO が提供しているシェル関数で jconsole を処理用の ijs スクリプトを引数として起動し (doWithJ メソッド)、j では rff 関数で引数を読み出し処理した結果は同じくテキストファイルに wtf 関数で書き出してシェルを終了する。次に、XOJO の GUI 側でその結果ファイルをボタンアクションで読み込み (readFromFile メソッド)、別の edit control で表示する。サンプル ijs の do_with_j.ijs にはテキストファイルの入出力用関数 (rff, wtf) が定義済みなので、このファイルを編集し所望の処理を記述する。編集終了後、関数名を決めて別名保存して処理用 ijs を完成させる。つまり j の関数一つあたり一つの ijs を割り当て、都度、jconsole と共に起動させるという構図になる。

1.3.3 Mac と Windows の処理方法の相違

Mac の場合は XOJO から作業ディレクトリへのテキスト書き出しをシェルの cd と echo コマンドを連結して使い、J での計算結果ファイルの読み出しは同じくシェルの cd と cat コマンドを連結して使い、XOJO のシェル関数の Result プロパティで cat された情報を取得している。Windows の場合はコマンドプロンプトでコマンドの連結が出来ないことと、echo の仕様が Mac とは異なるため、XOJO のファイルシステム処理機能（Folderitem と TextStream 入出力関数）を使って

バッチファイルの作成と引数と結果の読み書きを実現している。この違いは各々のファイルの shellModule のソースで確認して欲しい。

1.4 画像処理での応用（今後の課題）

今回のサンプルではテキストでの引数、結果の受け渡しを示したが、XOJO control にある canvas を使うと画像の座標、カラー情報の取得も可能になるので、画像を対象にしたアプリケーション作成も可能である。j での処理結果を画像ファイルで出力してシェルを抜ければ、その画像を XOJO の canvas へロードして結果を表示するというアプリケーションも可能である。

1.5 おわりに

Qt でのフォームエディタがないこと、また、Qt での GUI control の配置が bin で制御されていてサイズと配置について自由度が少ないことから、慣れ親しんだ J6 のフォームエディタに近い XOJO の GUI 開発環境は今後大いに活用できると考えている。GUI は XOJO、処理エンジンは jconsole という構図で J を活用して行きたい。

1.6 program listing: executeWithJ.sh

```
/Applications/j64-804/bin/jconsole $1
```

1.7 program listing: c_to_f.ijs

```
NB. accept x. y.
9!:49 XNAMES =: 1

NB. file read/write fucntions

NB.write string data x. to y. file
write_str_to_file =: 4 : 0
x. 1!:2 <y.
)

NB.write string data x. to y. file
wtf =: 4 : 0
x. 1!:2 <y.
)

NB.read one line string file
read_one_line_file =: 3 : 0
1!:1 <y.
```

```
)

NB.read one line string file
rff =: 3 : 0
1!:1 <y.
)


NB. ...... write your specific program and exit

c_to_f =: 3 : 0
c =. ". rff 'celsius.txt'
f =. 32 + (9 * c) % 5
(5j1 ": f) wtf 'fahrenheit.txt'
)

c_to_f 1

exit 9
```

1.8 program listing: f_to_c.ijs

```
NB. accept x. y.
9!:49 XNAMES =: 1

NB. file read/write fucntions

NB.write string data x. to y. file
write_str_to_file =: 4 : 0
x. 1!:2 <y.
)

NB.write string data x. to y. file
wtf =: 4 : 0
x. 1!:2 <y.
)

NB.read one line string file
read_one_line_file =: 3 : 0
1!:1 <y.
)

NB.read one line string file
rff =: 3 : 0
1!:1 <y.
```

```
)


NB. ...... write your specific program and exit

c_to_f =: 3 : 0
c =. ". rff 'celsius.txt'
f =. 32 + (9 * c) % 5
(5j1 ": f) wtf 'fahrenheit.txt'
)

f_to_c =: 3 : 0
f =. ". rff 'fahrenheit.txt'
c =. ((f - 32) * 5)%9
(5j1 ": c) wtf 'celsius.txt'
)

f_to_c 1

exit 9
```

1.9 program listing: do_with_j.ijs

```
NB. accept x. y.
9!:49 XNAMES =: 1

NB. file read/write fucntions

NB.write string data x. to y. file
write_str_to_file =: 4 : 0
x. 1!:2 <y.
)

NB.write string data x. to y. file
wtf =: 4 : 0
x. 1!:2 <y.
)

NB.read one line string file
read_one_line_file =: 3 : 0
1!:1 <y.
)

NB.read one line string file
rff =: 3 : 0
```

```
1!:1 <y.
)


NB. ...... write your specific program and exit


NB. ......

exit 9
```