Department of Computer Science
National University of Singapore

CS3249 User Interface Development

AY2013-14 Semester 2

Lab 2: Custom Widget

Objectives

- Learn to create and extend custom widget.
- Learn to use `QLayout` subclasses to layout and align widgets.
- Learn to use signals and slots for communication between widgets.

Part 0. Preparation

1. Under Home Folder, create a folder with your name, e.g., `myname`. This is your working folder.

Part 1. Creation of Custom Widget

This part guides you through the process of creating the `findDialog` widget discussed in the lecture as a subclass of `QWidget`.

1. Create a folder called `myeditor-1` under your working folder `myname`.
2. Copy everything in `myeditor` that you saved in Lab 1 into `myeditor-1`.
3. Download `finddialog.zip` for Lab 2 into `myname`. Open `finddialog.zip`, and copy `example.txt` into `myeditor-1` and `find.png` into `myeditor-1/images`.
4. Open `myeditor.qrc` and add `find.png` into it.
5. Create a new file called `FindDialog.h` and add the class definition for `FindDialog` as shown in the lecture notes. Remember to add the standard macros

   ```
   #ifndef FINDDIALOG_H
   #define FINDDIALOG_H
   #endif
   ```

   and include header files for `QWidget` and the required class declarations.
6. Create a new file called `FindDialog.cpp` and add the implementation of class functions, as shown in the lecture notes, for

```
FindDialog::FindDialog()
void FindDialog::enableFindButton(const QString &str)
void FindDialog::findText()
```

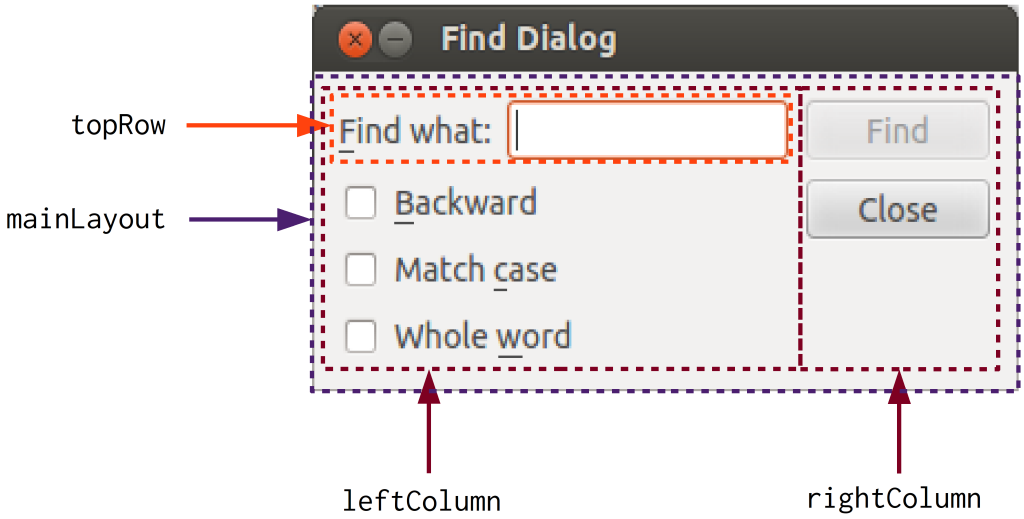
7. Edit `MyEditor.h` as follows:
   - Include header file `FindDialog.h`.
   - Add public function `~MyEditor()`.
   - Add private slot `findText()`. Make sure the arguments of the slot function are correct.
   - Add private variables `findAction` and `findDialog`. Make sure their data types are correct.
8. Open `MyEditor.cpp` and edit `~MyEditor()`, `close()`, `createWidgets()` and `createActions()` as shown in the lecture notes.
9. Compile the programs in three steps

```
qmake -project
qmake
make
```

10. Run `myeditor-1`, open `example.txt` and test `findDialog`.

Part 2. Extension of Custom Widget

This part guides you through the process of extending `FindDialog`. by adding check boxes and button. You will also learn how to use `QLayout` subclasses to layout and align the widgets. The layout of the extended `FindDialog` is as follows:



1. Create a folder called `myeditor-2` under your working folder `myname`.
2. Copy everything in folder `myeditor-1`, except the files `myeditor-1.pro` and `myeditor-1`, to folder `myeditor-2`.

3. Edit `FindDialog.h` in `myeditor-2` as follows:
    - Add three check boxes of type `QCheckBox*` for `backwardBox`, `matchCaseBox` and `wholeWordBox`. Remember to include the class declaration for `QCheckBox`.
    - Add another button of type `QPushButton*` for `closeButton`.
    - Change the signal `find()` to the following

      ```
      void find(const QString &str, QTextDocument::FindFlags
      flags);
      ```

      The flags are used to indicate options for text search, namely backward search, case-sensitivity and whole-word search, which are indicated by the check boxes.
    - Add a new signal close() as:

      ```
      void close();
      ```

      This signal is used to close the dialog.
4. In `FindDialog.cpp` of `myeditor-2`, edit `FindDialog::FindDialog()` as follows:
    - Remove the original `QHBoxLayout` called `layout`.
    - Create a `QHBoxLayout` called `topRow` to layout `label` and `targetText`.
    - Create a `QVBoxLayout` called `leftColumn` to layout `topRow`, `backwardBox`, `matchCaseBox` and `wholeWordBox`. Use `addLayout()` to add layout and `addWidget()` to add widget.
    - Create a `QVBoxLayout` called `rightColumn` to layout `findButton` and `closeButton`.
    - Create a `QHBoxLayout` called `mainLayout` to layout `leftColumn` and `rightColumn`.
    - (Optional) You can fix the size of `FindDialog` though `mainLayout` as follows:

      ```
      mainLayout->setSizeConstraint(QLayout::SetFixedSize);
      ```

      `mainLayout` will calculate the optimal size automatically.
    - Set `mainLayout` as the layout of `FindDialog`.
    - Connect the signal `clicked()` of `closeButton` to the slot `hide()` of `FindDialog`.
5. In `FindDialog.cpp` of `myeditor-2`, edit `FindDialog::findText()` as follows:
    - Define a local static variable `flags` of type `QTextDocument::FindFlags`.
    - If backwardBox is checked, add the flag `QTextDocument::FindBackward` to `flags` by bit-wise OR operation (`|=`).
    - If `matchCaseBox` is checked, OR `QTextDocument::FindCaseSensitively` to `flags`.
    - If `wholeWordBox` is checked, OR `QTextDocument::FindWholeWords` to `flags`.
    - Emit the signal `find(targetText->text(), flags)`.
6. Edit the function arguments of the slot function `MyEditor::findText()` in

MyEditor.h and MyEditor.cpp to match those of the signal FindDialog::find(). In addition, call the find function of textEdit with the flags as follows:

```
textEdit->find(str, flags);
```

This allows textEdit, an instance of QPlainTextEdit, to search for str with the options specified in flags.

7. Compile the programs in three steps

```
qmake -project
qmake
make
```

8. Run myeditor-2, open example.txt and test findDialog.

Part 3. QDialog vs. QWidget

FindDialog can be defined to inherit QDialog instead of QWidget. This part explores the slight difference between QDialog and QWidget.

1. Make a copy of myeditor-2 and call it myeditor-2w.
2. Edit FindDialog.h to change the base class of FindDialog to QDialog.
3. Compile the programs.
4. Make a copy of the new myeditor-2 and call it myeditor-2d.
5. Run myeditor-2w and test the findDialog.
6. Run myeditor-2d and test the findDialog.
7. What's the difference between QDialog implementation vs. QWidget implementation?

Submission

After completing the lab exercise, run myeditor-2d and myeditor-2w and show them to the Lab TA for verification. Then, print and submit the following to the Lab TA:

- MyEditor.h, MyEditor.cpp, FindDialog.h (QDialog version) and FindDialog.cpp (QDialog version). Remember to write your name, matriculation number and lab group as comments in the source files.
- A screen shot of the revised MyEditor and FindDialog. To save a screen shot into a file, run the program and press ALT+PRTSC. By default, the screen shot will be saved into the Pictures directory under Home. You can save it into another directory if you wish.

Important Note

Save your programs into a flash drive for subsequent use.