Demonstration Script for Lecture 1

NOKIA

Demonstration Script

This demonstration shows the basic functionality of QtCreator and Qt. The intention is not to give a detailed understanding, but to show what Qt is capable of and what the rest of the course will be about.

QtCreator – Overview

1. Start QtCreator
2. Walk through the welcome pages.
    - **Getting Started** has links to examples and tutorials.
    - **Community** links to entry points in the Qt community.
    - **Develop** lets you create and manage sessions and projects.
3. Create a new **Qt4 GUI Project**. Name it ListDemo or similar.
4. Do not add any extra modules, but discuss some of them.
    - **QtNetwork**, networking. TCP and UDP, HTTP, FTP, etc.
    - **QtOpenGL**, enables OpenGL acceleration of painting, as well as direct access to the OpenGL API (for custom 3D graphics).
    - **QtSql**, interaction with databases. Can handle all from in-memory/single file sqlite databases to remote Oracle servers, etc.
    - **QtWebKit**, web rendering and processing classes. It is possible to embedd Qt C++ contents in web pages and add interaction between HTML, JavaScript, C++ and Qt.
    - **Phonon,** multimedia framework. Interesting not only because of its features, but that it has been developed in close collaboration with the KDE project.
5. Let creator create a template application based on **QWidget**, lets call the class Widget for simplicity.
6. When the project has been created, walk through the project pane. Show sources, headers, forms. Also, make sure to right click on the project and show that this is where you can add new classes (*add new*), change release/debug (*build configuration*), and if you have multiple projects, change which one you want to run (*run configuration*).

Designing User Interfaces

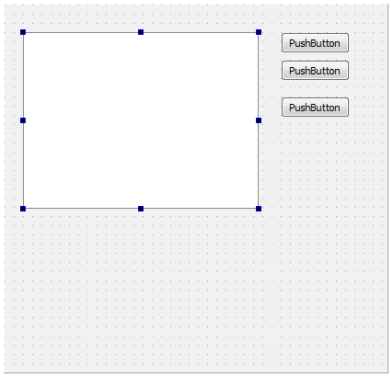
7. Open the form design mode. Go through the panes, tell the class that this is what we will be using later on, you just want to get the naming right.
    - **Toolbar** (editing modes) – you can work with the design, but also the connections between different widgets (will be shown later), as well as the tab order and buddies (QLabels connected to a specific other widget).
    - **Widgets** – here all the widgets are shown. For those with a small

NOKIA

screen, you can switch it to icon mode (right click). The widgets are divided into groups (buttons, item views/widgets, containers, input widgets, display widgets).

- **Object tree** – shows the hierarchy of the widgets in the current form.
- **Properties** – shows the property of the current widget. Notice that each class' inheritance tree is visible as well. For instance, the QObject has a name, then QWidget adds the rest of the properties.

8. Now add three QPushButtons and a QListWidget as shown below.



9. Now rename and re-title the buttons according to the table below. You can change the name in the property editor (show the inheritance tree QObject – QWidget – QAbstractButton – QPushButton), the text can be changed from the property editor or by double clicking on the button and type.

Button	Property	New value
Top	Name	addButton
Top	Text	Add...
Middle	Name	deleteButton
Middle	Text	Delete
Bottom	Name	clearButton
Bottom	Text	Clear

10. Preview (from the menu Tools – Form Editor – Preview) the widget and show that the widgets do not stretch or adapt to different window sizes.
11. Introduce the concept of layouts. Grids, horizontal boxes, vertical boxes.
12. Apply a grid layout and show that the buttons are distributed evenly along the side of the list widget. Break the layout and add a spacer below the buttons, then re-apply the grid layout.
13. Preview the proper dialog, show that you can preview it in different styles (Windows, Plastique, etc).
14. Switch to signal/slot editing mode in the toolbar.
15. Connect the clear button's `clicked()` signal to the list widget's `clear()` slot.
16. Show that the connection shows up in the signals slots editor dock (at the bottom).

17. Switch back to widget editing mode in the toolbar.
18. Right click on the add button and pick Go To Slot...
19. Pick the `clicked()` signal from the dialog that pops up.

Basic Code Editing

20. Mention that the name of the newly created slot ties it to the button. Be careful when changing names in either end (slot or widget).
21. The code editor works as an ordinary editor. Move around, show line numbers.
22. There are some useful short-cuts to go through:
    - F4 switches between header and implementation.
    - F2 switches between header and implementation, but also the current method (if you're standing in your method's name).
    - Ctrl+K brings up the locator used for searching and navigation (can find classes, methods, files, but also go to a specific line, etc)
23. Now start typing, go to the top and add an include for `QInputDialog`.
24. Ctrl+k, enter "`m on_a`" <return>, this brings you to the `on_addButton_clicked` method.
25. Enter code "`QString newText = QInputDialog::getText(`", emphasis the code completion.
26. complete the line with "`0,"Enter text", "Text");`", again emphasis the help you get as you type arguments.
27. Now complete the slot method to look as this:

```
void Widget::on_addButton_clicked()
{
    QString newText = QInputDialog::getText(this, "Enter text", "Text:");
    if( !newText.isEmpty() )
        ui->listWidget->addItem(newText);
}
```

28. Now build the application (Ctrl+Shift+b) and show the build progress bar (above the run/debug/compile buttons) and the compile output pane (Alt+4).
29. Run the application (Ctrl+r), show that the add and clear buttons work, but that the delete one is dead.
30. Go to the design view and repeat the stages to create an `on_deleteButton_clicked()` slot.
31. Enter the following code in the slot method. Mention the `foreach` macro and that deleting an item removes it from the list. The `QListWidgetItem` tells the parent list about its deletion, i.e. no dangling pointers.

```
void Widget::on_deleteButton_clicked()
{
    foreach (QListWidgetItem *item, ui->listWidget->selectedItems())
        delete item;
}
```

NOKIA

32. Run the application and show that it works.

Handling Enabled States

33. Mention that the delete button always is enabled, that is not correct. It needs to be enabled or disabled as soon as the selection of the list changes.
34. Switch to the header (F4) and create a private slot called updateDeleteEnabled.
35. Switch back to the source (F4) and create the method frame:

```
void Widget::updateDeleteEnabled()
{
}
```

36. Now, enter some code in it (DO NOT ADD THE FINAL SEMI-COLON)

```
ui->deleteButton->setEnabled(ui->listWidget->selectedItems().count()!
=0)
```

37. Go to the constructor.
38. Explain that we will connect changes in the selection to this slot.
39. After the `ui->setupUi` call, enter the following code. Point out the code completion for both signal and slot (it is our custom slot!).

```
    connect(ui->listWidget->selectionModel(),
                SIGNAL(selectionChanged(QItemSelection,QItemSelection
)),
            this, SLOT(updateDeleteEnabled()));
```

40. Now try to run the project (Ctrl+r).
41. Point out the build issue (“expected ';'...”). Show that this is visible both from the compile output (Alt+4) as well as the build issues (Alt+1). Also, show the wavy red line under the “`}`”, indicating the problem live in the editor.
42. Fix the problem and run the application.
43. Notice that the button is enabled regardless of selection, until you add an item and remove it. The method needs to be called once to initialize it.
44. Add a call to the updateDeleteEnabled method just after the connection in the constructor.
45. Run the application and show that all works as expected now.

Basic Debugging

46. Not all errors are caught at compilation. To hunt these down, a debugger is used.
47. Add a breakpoint in the first line of the `on_addButton_clicked` slot method.
48. Start a debugging session (F5).
49. Click the add button to trigger the break point.
50. Show the components of the debugger (call stack, locals, breakpoints, threads).

NOKIA

51. Step over the current instruction (F10, or the button in the debug toolbar).
52. Enter a string in the dialog and see that string in the locals window. (Use international characters, they work.
53. Continue (F5) and close the application.
54. To exit the debugger, switch to edit mode (the left side buttons or Ctrl+2).

End of demonstration!

NOKIA