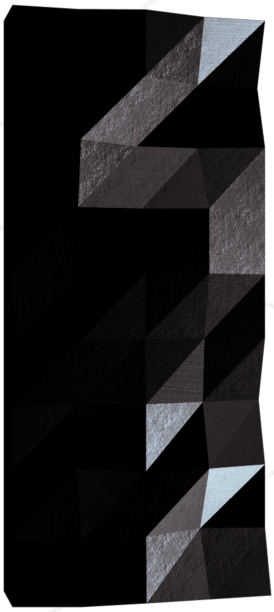

QML and C++ integration

Using QML object from C++ and vice versa

Qt training training@qt.io
42 Pedago Staff pedago@42.fr

Summary: QML and C++ objects communicate via the meta-object system. The QML engine allows bindings between the objects, exposing new properties and objects and registering new QML types.

Contents

Chapter I

General instructions

Unless explicitely specified, the following rules will apply every day of this Piscine.

- This subject is the one and only trustable source. Don't trust any rumor.
- This subject can be updated up to one hour before the turn-in deadline.
- The assignments in a subject must be done in the given order. Later assignments won't be rated unless all the previous ones are perfectly executed.
- Be careful about the access rights of your files and folders.
- You must follow the `turn-in process` for each assignment. The url of your `GIT` repository for this day is available on your intranet.
- Your assignments will be evaluated by your Piscine peers.
- In addition to your peers evaluation, a program called the "Moulinette" will also evaluate your assignments. Fully automated, The Moulinette is tough and unforgiving in its evaluations. As a consequence, it is impossible to bargain your grade with it. Uphold the highest level of rigor to avoid unpleasant surprises.
- You <u>must not</u> leave in your turn-in repository any file other than the ones explicitly requested by the assignments.
- You have a question? Ask your left neighbor. Otherwise, try your luck with your right neighbor.
- Every technical answer you might need is available in the `mans` or in the Qt Documentation, which is available both in `QtCreator` IDE and on http://doc.qt.io.
- Remember to use the Piscine forum of your intranet and also Slack!
- You must read the examples thoroughly. They can reveal requirements that are not obvious in the assignment's description.
- Use the latest Qt version. Qt version 5.10 or newer is recommended.

- Many Qt classes are available as standard C++ classes. Qt classes should be preferred to standard classes, as you are supposed to learn Qt.
- Basic Qt coding style should be used, so read http://wiki.qt.io/Qt_Coding_Style before writing any assignments.
- Deprecated macros, functions or classes must not be used.
- Any build system, such as `qmake`, `cmake` or Qt Build System, can be used in the assignments. Instructions are based on `qmake`.
- Any editor or IDE, supporting Qt, can be used. However, `QtCreator` usage is strongly recommended.
- By Thor, by Odin! Use your brain!!!

Chapter II

Assignment 00: Exposing properties

Create a simple QML UI with a `Rectangle`, containing a `Text` object. Expose the colour and text properties from the `main()` function, so that they are available in all QML files. Use the exposed properties in QML to change the rectangle colour and `Text` item's text to light green and "Hello world", respectively.

Chapter III

Assignment 01: Bindings and signals

Implement similar functionality to the previous assignment, but this time define the background colour and text as `QObject` subclass properties. Use again the properties to change the QML rectangle colour and text item's text to green and "Hello world", respectively. After binding the QML rectangle colour and `Text` item's text, you are not allowed to change the binding anywhere in this assignment.

When the user clicks on the rectangle, its colour must be changed to red. In a signal handler, set the `Text` item's text to "Background colour changed " + <the new text value>. Create another binding, which prints the new text value to the debug console, whenever your `QObject` subclass text property changes.

Chapter IV

Assignment 02: Object registration

Let's create an application, which stores employee data into the model. This time the employee data and the model are implemented in C++ and used in QML.

Subclass `QObject` to create a simple employee type. The subclass should have two properties: a name of type `QString` and a salary of type `qreal`.

Make your new type available in QML. Create a row of two text editors, which show employee name and salary. Create an employee object as well to see some data in the row.

Chapter V

Assignment 03: C++ models

In the previous assignment, we were able to show a single employee data item on the window. Let's complete the application to show employees from the model.

Subclass `QAbstractListModel`. The model should contain employees. In addition to pure virtual functions, add a function to add a new employee to the model. The model should support two roles: employee name and salary, which are defined in the employe data type. Register the employee model type as a singleton. Do not add any employees in C++, but add ten employees in QML to show some initial data as in the image below.

Show the model data in a list view, which uses a delegate with a simple row of two `TextInput` items: name and salary. If either name or salary is modified in the user interface, the new values must be stored into the model.

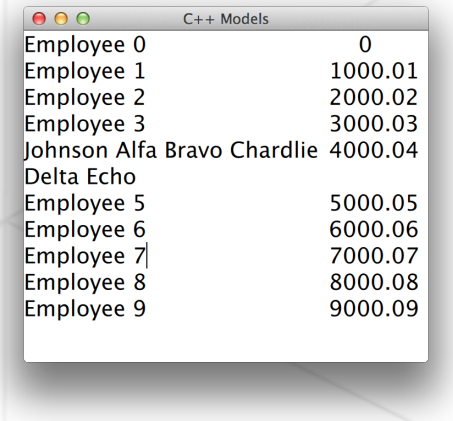

Figure V.1: C++ model in QML

Chapter VI

Assignment 04: Painted items

Implement a custom button in C++. The button must have a background colour and text properties, which may be changed in QML, In addition, it must have an event handler, which emits a signal, when the button is pressed.

Register the type in QML and implement a simple QML program, which changes the button colour, if the button is pressed. You must not use any QML signal handlers to implement the required functionality.

Chapter VII

Assignment 05: Scene graph items

Implement a custom scene graph node, which renders an image. The image is given as a string in a QML code and the item should create a `QSGTextureNode` to render the texture as shown below.

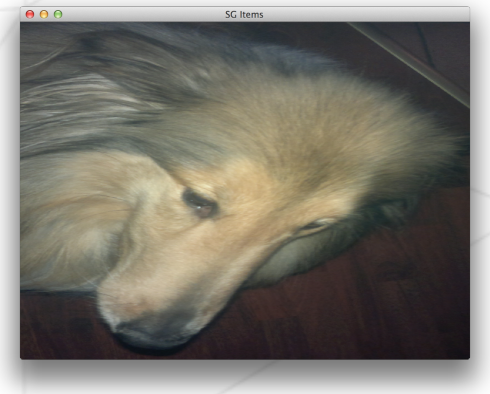

Figure VII.1: Texture SG node