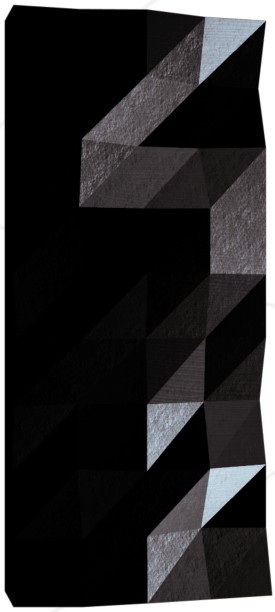

Application engine

Application business logic using Qt libraries

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

Summary: Qt provides a variety of cross-platform programming APIs. Today we will get familiar with most important of these.

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 must not 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: Video player

Use the template in the `res` folder to implement a basic audio/video player. Use `QMediaPlayer`.

- The `Open` button should use a dialog to add media files to the player. `QFileDialog` is very useful.
- The functionality of the three other buttons is self-explanatory.

Chapter III

Assignment 01: Subtitles

Add subtitles to the video player, you implemented in the previous assignment. The subtitles may contain any random text and there is no need to sync the subtitles with the actual video content. You may just show the same subtitle for the whole duration of the video.

Figure III.1: Subtitles

There are plenty of ways to implement the subtitles, but in this assignment you are asked to change the `MainWindow` implementation so that it uses a `VideoWidget` and `VideoSurafec` classes. You may add the classes to your project from the `res` folder.

The only class you need to change is `VideoWidget`. All member functions have been declared, but you need to complete their implementations.

- If the video buffer does not have a handle, you should support ARGB32_Premultiplied, ARGB32, RGB32, RGB565, and RGB555 pixel formats. Otherwise, no pixel format is supported. The `isFormatSupported()` function returns true, if the format is not valid, frame size is not empty and the handle type is not `NoHandle`.

- Start the video surface, only if the format is not invalid and the frame size is not empty. Store the format, size, and the surface rectangle to the corresponding member variables. Note that the variables are defined for you. Also update widget and video surface geometry and rectangle, respectively.

- There are no special conditions to stop the video surface.

- The `present()` function should repaint the widget, the frame, surface format, and size are equal.

- The `paint()` function should paint the frame pixels and add subtitle text. Draw a rectangle around the text, so that the text is visible whatever the video frame pixel colours are.

Chapter IV

Assignment 02: XML and JSON

Write a program, which reads JSON files and converts them to XML files. No graphical UI is needed. The JSON file and XML files names may be hardcoded in the source code.

Parse recursively JSON arrays and objects. The array may contain another array or a JSON object. A JSON object may have key and value pairs. Use keys as XML element names. Write values (strings, Booleans, doubles) inside XML start and end elements. Undefined values can be just ignored. The example JSON files in the `res` folder should produce something similar to the XML file below.

```
<?xml version="1.0" encoding="UTF-8"?>
<glossary>
    <GlossDiv>
        <GlossList>
            <GlossEntry>
                <Abbrev>ISO 8879:1986</Abbrev>
                <Acronym>SGML</Acronym>
                <GlossDef>
                    <GlossSeeAlso/>
                    <para>A meta-markup language, used to create markup languages such as DocBook.</para>
                </GlossDef>
                <GlossSee>markup</GlossSee>
                <GlossTerm>Standard Generalized Markup Language</GlossTerm>
                <ID>SGML</ID>
                <SortAs>SGML</SortAs>
            </GlossEntry>
        </GlossList>
        <title>S</title>
    </GlossDiv>
    <title>example glossary</title>
</glossary>
```

Chapter V

Assignment 03: Databases

Using the template in the `res` folder, implement an application, which stores employee data into a SQL database. The template code uses `QStandardItemModel`, but nothing prevents you use more convenient model classes, if you wish.

- The program should either create a database or read data from the database. when it is started. The database should contain three fields: employee name, employee address, and salary. All fields may be at most 255 character strings for simplicity.
- When the button is clicked, add new employee data into the database. Obviously, the changes should be visible in the view. Use the model member variable in `MainWindow`.
- When the program is re-started, the data shown be read and shown from the database, if the database exists.

Chapter VI

Assignment 04: Inter-process communication

Implement an application, which shares images between two processes using shared memory and a file watcher. Once again, the template can be found in the `res` folder. The template contains two projects: server and client. You may right-click the project name and run several projects in `Qt Creator`. The run order of the projects is irrelevant in this assignment.

- In the server side, shared memory and file watcher functionality has been implemented in the corresponding `QObject` subclasses.
- In `FileLoader`, use `QFileSystemWatcher` to detect changes in a folder, where you write the file. Use platform-specific temp folder. The image label member is used to show the loaded image, while the text label is used to show any status information.

`QDataStream allows de-serialisation from a file to many Qt types.`

- Use Qt shared memory to load image data in the image label member. Pay attention to different error situations: cannot create shared memory, shared memory too small and so on. Use message box to clearly show errors to the user.
- In the client side, you just need to serialise the image into a file or shared memory, depending which button the user clicked on. You must handle the deletion of a shared memory segment in both the client and server sides.
- Remove the files, created temporarily by the client or the server.

Chapter VII

Assignment 05: Networking with sockets

The program template is similar to the previous assignment, so it is consisting of two subprojects.

Implement the following functionality in the server.

- The server listens to local socket connections in a port 4242.
- There may be any number of clients, served by the main thread.
- The server echoes all incoming messages to all clients, including the message sender.
- After all client connections have been closed, the server window is closed. The server window must be a frameless window, as it is in the template.

The client is more straightforward. The client establishes the connection with the server and sends the text in the line editor, when the return key is pressed or a push button is clicked on. All received data is shown in the text editor.

Chapter VIII

Assignment 06: Networking with HTTP

Use OpenWeatherMap at `http://openweathermap.org` to get the weather information in different cities around the world. The UI and basic initialisation have been implemented in the template in the `res` folder. You need to implement the networking functionality.

The URL of the service and the basic query have been implemented in the function, which handles the `Return` key press in the `Widget` class. The widget contains three labels to show the temperature, weather description, and an icon, describing the weather. The city name and optionally a country code, is given in the line edit widget.

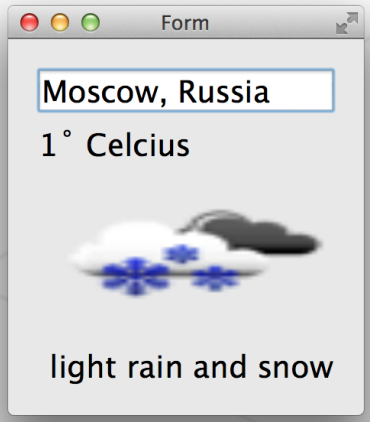

Figure VIII.1: Weather service

You may read the details of the JSON data, returned from the service from `http://openweathermap.org/current`. The `weather` JSON array provides you with two objects. `description` is an object, providing a description string value and `icon` value defines a png file name of the description icon. You need to load the png file from the `http://openweathermap.org/img/w/<icon_name>.png`. The `QImage::load()` function is very useful. Finally, the temperature is given in a `main` object, using a key `temp`.

Pay attention to proper memory management.

Chapter IX

Assignment 07: Charts

Extend the previous assignment with line charts.

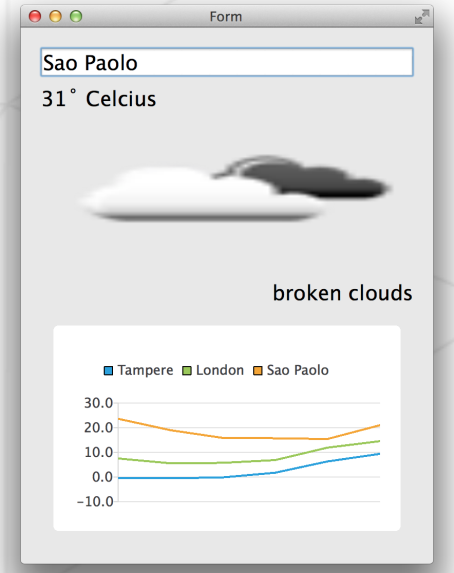

Figure IX.1: Weather service with charts

Add a line chart widget to the UI, as shown below. Change the URL to `http://api.openweathermap.org/data/2.5/forecast`. Instead of a single temperature and weather description, the request returns a JSON array of objects. The objects themselves have similar structure as in the previous assignment. Use a line chart to show the temperature forecast. The number of days can be controlled with `cnt` property in the HTTP request.