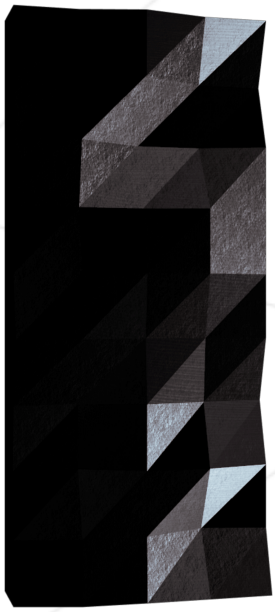

# Widget UI

## GUI applications based on widgets

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

*Summary: Widgets are used in user interfaces in desktops, as they provide a traditional mouse-based user experience. Another UI approach is Qt Quick, which is used in modern touch-based user interfaces in desktop, mobile, and embedded. Qt provides developers with a large number of widgets from simple labels and push buttons to generic scroll areas, multiple interface documents (MDI), and main windows with menus, tool bar, and status bar.*

# 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

# Day-specific instructions

Let's start creating graphical user interfaces for Qt applications. Today and in the rest of the assignments this week, use an application template "Qt Widgets Application" in `Qt Creator`. The template wizard allows you to decide, whether you want to use `QtDesigner` and `.ui` forms or just a text editor to create user interfaces. Typically UI implementation is faster and easier using designer tools.

# Chapter III

# Assignment 00: Hello world widgets

Create a GUI application, showing a push button with text "Hello World!". The application should be closed, when the button is pressed.

- Use a `QWidget`-based form.
- The push button should expand horizontally, when the window size is changed.

`Layouts are your friends.`

- The font point size in the push button text should be 20.
- Use "Edit Signals/Slots" editor. Press `F4` in `QtDesigner` to start the editor. Use the editor to quit the application, when a user presses on the button.

# Chapter IV

# Assignment 01: Layouts

Let's implement a trivial browser application. Look at the screen shot below.

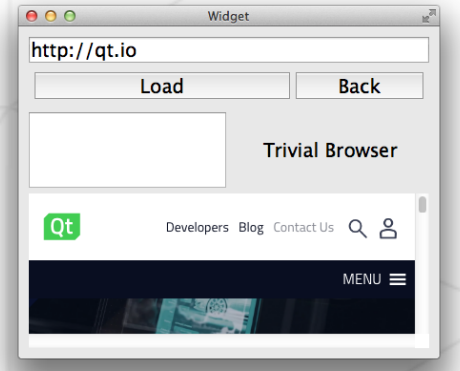


Figure IV.1: Trivial browser

Design the layout of the application first.

- The top-most widget is an edit line, which is used to give the URL.
- There are two buttons: `Load` button should be twice as wide as `Back` button also, if the window size changes.

`Look at the meaning of stretch in layouts.`

- The next line contains a text browser and label. The text browser does not have any special functionality here. The horizontal space should be equally divided between the text browser and the label. The label text should be aligned to the horizontal and vertical center of the label.

- Note that the webengine view widget must be added in the code in the main widget constructor. Add it to the bottom of the layout.

`By default, a widget application uses QtCore, QtGui, and QtWidgets modules. To avoid compilation and linker errors, you need to add webenginewidgets to your project.`

Let's implement the functionality as well.

- If the return key is hit in the line editor, the web view should load the URL in the editor. Note that the webengine does not automatically add any protocol, like `http`.
- `Load` should load the URL and `Back` should go back to the previous page.

`QtDesigner context menu has Go to slot....`

# Chapter V

# Assignment 02: Dialogs

Let's add some dialogs to the application, you created in the previous assignment. You may copy the previous assignment and modify existing files. It may take a while before the web page is loaded, so it would be useful to show some progress to the user. Add a progress bar in the layout above the webengine view. The progress bar should be visible only, when it shows the load progress.

Change the `Load` button functionality in the following way:

- Open a window modal dialog. You may create your own dialog or use a suitable `QDialog` subclass.
- The dialog may just ask the user for a URL. If the dialog is accepted, load the URL to the webengine view.

The following screen shots help you understanding, what is required.

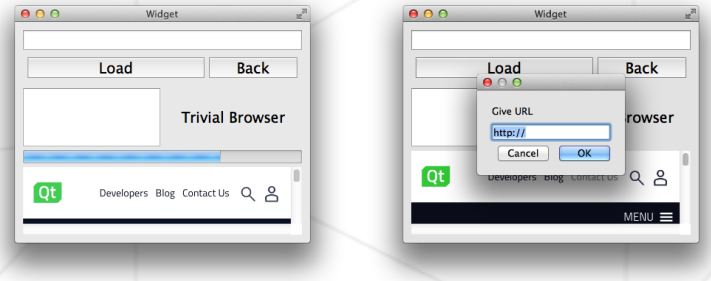


Figure V.1: Trivial browser with a progress bar and a dialog

# Chapter VI

# Assignment 03: Widget styling

Use style sheets to style your trivial browser according the the following requirements. After styling the application should look similar to the one below.

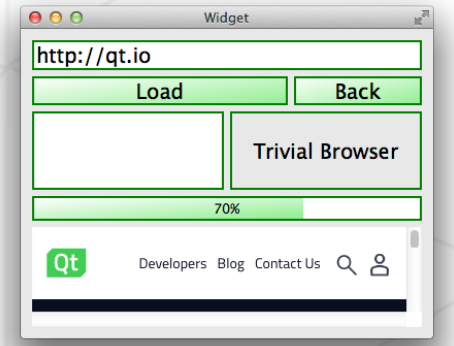


Figure VI.1: Styled trivial browser

- Each widget, including the dialog, should have a 2 pixel wide solid green border.
- Each push button should change the border colour to solid blue, if the mouse is hovering on the top of the button.
- Each push button should have a linear gradient background from the top left to the bottom right corner. The gradient should start from white and end to light green.
- Progress bar text must be aligned to the center of the progress bar.
- The progress bar chunk background should be similar to the push button background.

# Chapter VII

# Assignment 04: Main window

Your boss asks you to develop new office applications for your company. You start by creating an application template, which can be customised to implement a word processor, spread sheet, and other applications. The template should contain a menu bar, tool bar, and status bar, so you decide to use `QMainWindow` to implement it. An example template is shown below.

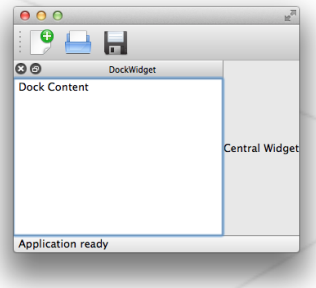


Figure VII.1: Office application template

`In this assignment, you must not use UI forms (.ui files). Implement all requirements in C++.`

Create a Qt widget project with an empty main window. Add menus, a tool bar, a status bar, a docked widget, and a central widget to the main window using the following requirements.

- The first task is to implement menus in the template. The main window has already a menu bar. Add two menus to the menu bar: `File` and `Help`. Both menus have at least one action: `File` has an exit action and `Help` menu has about Qt action. The exit action should exit from the application and about Qt action should call `QApplication::aboutQt`, when triggered. The actions should have icons and text. Icons should be added using a Qt resource system. Some suitable icons are located in `res/icons`.

- The tool bar should have three actions: new file, open file or save file. When an action is triggered, just show a dialog, displaying the text, corresponding the action.
- The status bar should show "Application ready" for ten seconds after the application is started.
- The application template should contain one docked widget, which should contain a list widget with a single text item.
- The central widget can be a label with some text.

# Chapter VIII

# Assignment 05: Localisation

Localise the application template from the previous assignment.

- Localise all UI strings in the application to any other language but English. Use the localised language in the application.
- Replace the label in the central widget so that it shows by default a Qt logo from the `icons` folder or a localised flag. Use such locale settings in the application that a localised flags is shown.

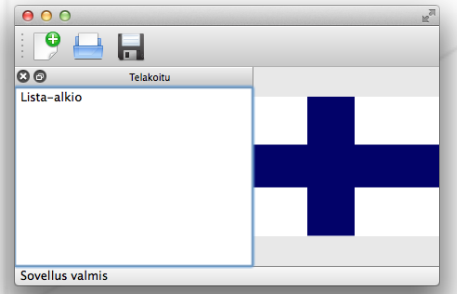


Figure VIII.1: Localised main window

# Chapter IX

# Assignment 06: Custom widgets

Draw a custom widget, which has a radial gradient background and a Qt image on the widget center.

- The radial background should have a radius of five pixels. Use a reflect or spread pattern to fill the widget area with the gradient. The gradient size (width/height) must follow the widget size changes.
- Draw an image on the widget center. If the widget is clicked, rotate the image 360 degrees in clockwise direction. The rotation center should be the image center. The widget may have a fixed size, e.g. the pixels size of the Qt image in the `res` folder.

`QWidget does not have a rotation property. What about adding a custom property, which is animated and used in image rotation?`

- Draw a reflected image below the centered image (look at the image below).

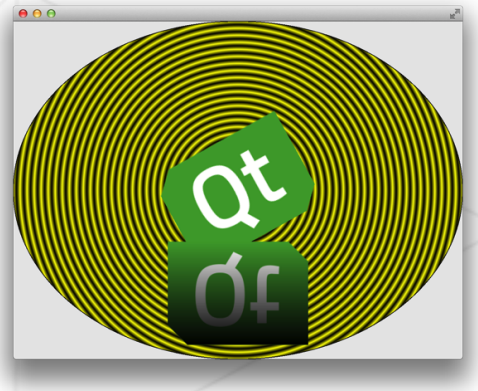


Figure IX.1: Custom widget

- Draw a rectangle with a gradient colour. The colour can be black or white, but the gradient should change the alpha value from 0 to 255.

- Compose the rectangle with the reflected image to have a fading effect as shown in Fig. IX. There is no need to rotate the rectangle or the reflected image.

`Rectangle is not an image, but image composition modes work only for images.`