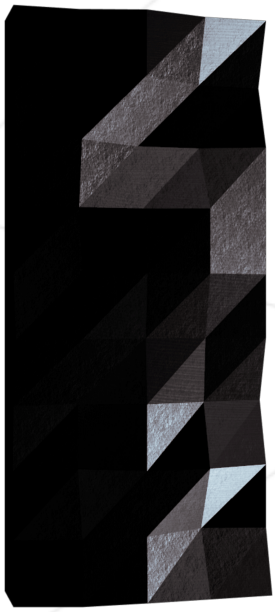

Qt Quick Controls

Using Qt Quick Controls 2 in composing UIs

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

Summary: Qt Quick Controls provide ready-made UI building blocks, including layout and styling classes.

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

Today we will use Qt Quick Controls 2 to create an application UI. In each assignment, new features are added into the same application UI, which is created with the `Qt Creator` application template. Rather than creating a new project for each assignment, you may just copy the source code of the previous assignment to a new folder, rename the project, and implement the requested features. Instructions do not make a lot of sense, if you do not do the assignments in the given order.

Chapter III

Assignment 00: UI Form with controls and Qt Quick layouts

Create your application using the `Qt Quick Controls 2 Application` template in `QtCreator`. You may remove all other content except the `Application Window` and its four properties: `visible`, `width`, `height`, and `title`.

Create a simple UI as shown in the figure below. You must create the UI form and the corresponding component separately. You may use `Qt Quick Designer` or just a text editor. Name your component as `MainPage.qml`. Concentrating on styling does not give you any additional score at this point, as there will be another assignment, focusing on custom styles.

Use Qt Quick Layouts to expand button, slider and progress bar horizontally across the window. Timer duration label and slider are grouped inside one layout. Use another layout to group the first layout, the button, and the progress bar.

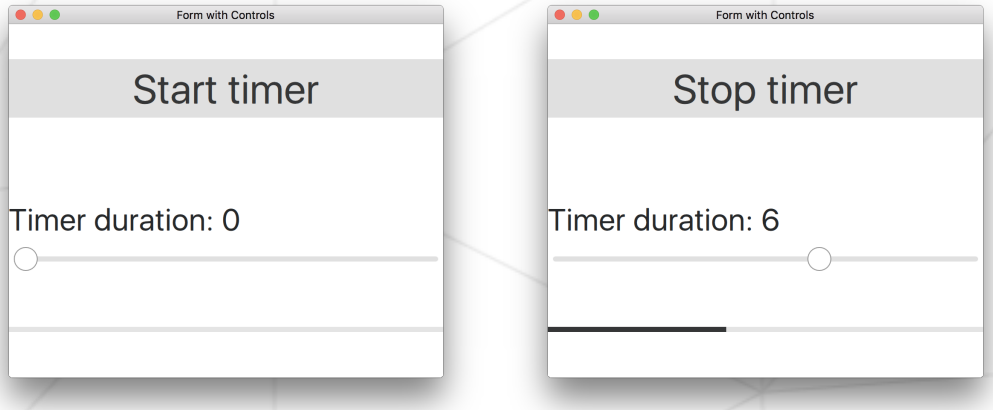

Figure III.1: Simple form, showing timer progress

Functional requirements include:

- Timer duration label must show the slider value.
- Choose the slider range freely, but it must be non-zero, e.g. 0-9. The slider value should determine the timer duration in seconds.

- When a user clicks on the button, show timer progress in the progress bar. The progress bar must be updated in every 20 ms.
- The button text must indicate the timer state, i.e., whether the user can start the timer (stopped) or stop the timer (running).

`It may be useful to have two timers.`

Chapter IV

Assignment 01: Application window

Copy the the previous assignment source code to a new folder and continue adding new features.

Add a header and footer to the application as shown in the figure below.

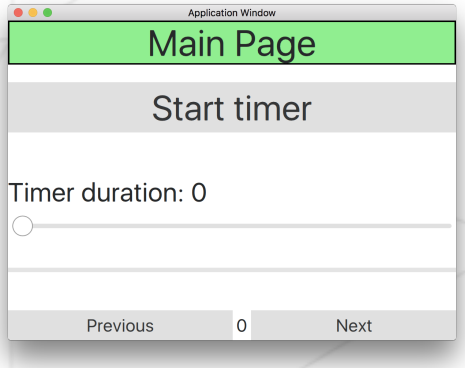

Figure IV.1: Application window with header and footer

The header must contain text in a bordered rectangle. The text shows the page title. The font size does not need to scale.

The footer must have two buttons to navigate to the previous and next page. Between the buttons, there is a indicator, which will show the page stack depth later. At this point, the indicator may show any hard-coded value.

The user must not be able to shrink the window so small that all the controls could not be visible in the window.

Chapter V

Assignment 02: Views

Our application contains just a single page. Add a generic UI form and the corresponding component. Name it as `CustomPage.qml`. Each page QML file must have a `Page` control. At the beginning, add just a label showing the page title and stack view depth.

Use `StackView` control to navigate between the pages. The initial page should be `MainPage`. By clicking on the left button, the user can navigate to the previous page and by clicking on the right button, the user may navigate to a new page.

The user can push as many pages to a stack that fit into available memory. In the assignment, all pushed pages can have the same type. However, each page must be identifiable by the page title, which is also shown in the window header. The stack depth indicator at the footer should now show the actual stack depth. The depth is at least 1, as there is always at least the initial page in the stack.

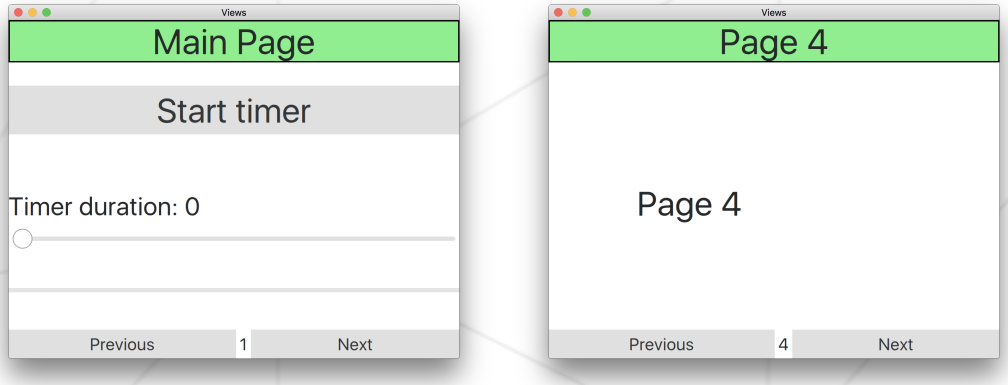

Figure V.1: Stack view with a main and custom page

Animate page changes in the following way:

- When a new page is pushed onto the stack, the scale and opacity of the existing page must be animated from 1.0 to 0.0. Correspondingly, the scale and opacity of the new page must be animated from 0.0 to 1.0.
- When a page is popped from the stack, the current page moves to the right and the new page appears from the left, which is the default behaviour.

Chapter VI

Assignment 03: Qt Quick Designer

Change only the custom page UI form to implement a Body Mass Index (BMI) calculator.

The calculator must work with metric and imperial (UK) units. Imperial units, used in the USA, can be ignored.

You must use different controls to give the body mass and height. If you decide to use `Tumbler` to input mass, you must use, e.g. `TextField` to input the height.

The controls must not allow user to give invalid values. The maximum height must be limited to 3.0 m.

Whenever either the body mass or height changes, it should be shown in the BMI value.

Implement everything into a `CustomPageForm.ui.qml` file.

Do not forget scalability. Font sizes may be fixed, but make sure all texts are readable, if the window size is changed.

`It is possible to split Qt Quick Designer view into a UI editor and text editor as shown below.`

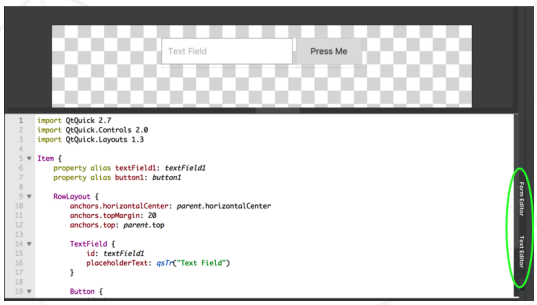

Figure VI.1: Qt Quick Designer with UI and text editors

Chapter VII

Assignment 04: Controls

Let's improve the navigation between several application pages.

Add `Drawer`, including `ComboBox` to the application. The idea is that user can quickly rewind to a previous page, if he or she has opened a large number of pages.

If there are no pages in the stack, the application must indicate this to the user in the combo box, as shown in the figure below. The page numbering in the combo box items must start from 1. The combo box must contain as many items as there are pages to which the user can navigate to. Note that the user cannot navigate to the current page.

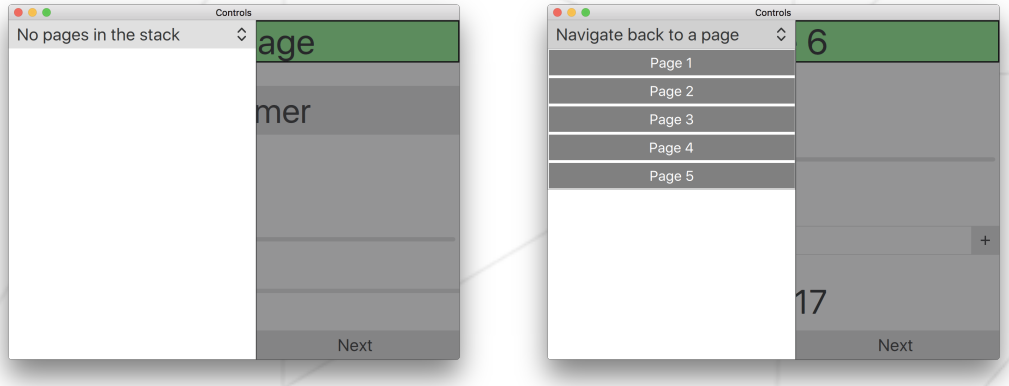

Figure VII.1: Combo box with 0 and a few pages in the stack

You must modify the presentation of the items in the combobox. The item text must be aligned horizontally centre and the background colour and font must be changed from the default one.

The font in the combo box display text and item texts must scale according to the window size.

And the final trivial requirement is that the application will navigate to the page, selected by the user.

Chapter VIII

Assignment 05: Styling

Let's finalise the application by customising the style of the UI controls.

- The first requirement is trivial. Just change the Qt Quick Controls style from `Default` to `Universal`.
- Create and define the application theme, i.e., a QML file, which contains nothing else but properties, used in theming. The theme must be a singleton. This makes it easy to do simple UI styling. Replace all hard-coded magic numbers (sizes, positions, margins, and font sizes) with property values in the theme. Possibly, your code contains no magic numbers. Do not worry, we will use the theme in the next step.
- Add a background image to a drawer. The image must be defined in the theme. There is a Qt logo in the `res` folder, but you are free to use any image you want.
- Create a new style custom style for the `Page` Qt Quick Control. The only required difference to the existing `Page` is a background image. Each page should have by default a background image, defined in the theme. The image visibility should be controllable with a custom `backgroundVisible` property. Adding new properties to a `Page` instance in `main.qml` is not acceptable solution.