QT.LABS.CONTROLS overview

Meetup #5 : Qt Labs controls

Guillaume Charbonnier
gcharbonnier@a-team.fr

Développement mobile avec Qt/QML
La Cantine Numérique le 30th of March 2016

Workshop objectives

Today we will have a code review of the gallery example.

sources :
http://code.qt.io/cgit/qt/qtquickcontrols2.git/tree/examples/controls/gallery
%QTDIR%/Examples/Qt-5.6/qtquickcontrols2/controls/gallery

Context

Qt supports several ways for designing UI :

- historical : based on QWidget (C++)
- with QML :
  - manually using Item based (Window, Rectangle, MouseArea..)
  - using Qt.Quick.Controls
  - using the brand new Qt.labs.controls in technology preview with Qt 5.6
- Canvas3D, Qt3D

The aim of this workshop is to discover Qt.labs.controls

Comparaison with QtQuick Controls 1.0

- features are almost identical (Hover is not supported)
- more efficient (loading time and memory footprint)
- simpler (ex : ScrollView vs Flickable using indicator/scroolbar)
- universal basic style + Universal + Material (a Qt style on its way)
- based on template concept (template beeing everything not visual)
- new elements (Drawer, RangeSlider, SwipeView) and some controls rewritten from Qt.Extra
- DPIAware

	Qt Quick Controls	Qt Labs Controls
==	ApplicationWindow, BusyIndicator, Button, CheckBox, ComboBox, Label, Menu, ProgressBar, RadioButton, Slider, SpinBox, Switch, TextArea, TextField, ToolBar, ToolButton	
	StatusBar, TreeView, TableView	no direct replacement
	ColorDialog, FileDialog, FontDialog	no direct replacement
!=	MessageDialog, Dialog	Popup
!=	Action	Shortcut
!=	Calendar	MonthGrid, DayOfWeekRow, WeekNumberColumn
!=	GroupBox	GroupBox, Frame
!=	ScrollView	ScrollBar, ScrollIndicator
!=	Stack, StackView, StackViewDelegate	StackView
!=	Tab, TabView	TabBar, SwipeView

Style

- select a style
- parameters
- customizing

3 styles are available :

- **default** : universal, simplier and lightweight
- **Universal** : style following Microsoft guidelines
- **Materiel** : style following Google guidelines

One can select the desired style (preceding order) :

- using a command line parameter : *-style STYLE*
- using a environment variable : QT_LABS_CONTROLS_STYLE
- using a config file :/qtlabscontrols.conf dans le QRC

if the selected style is not available, default style will be used.

Material style :

- Material.theme : Material.Light / Material.Dark
- Material.primary : any color (Material.BlueGray by default)
- Material.accent : any color (Material.Teal by default)

Universal style:

- Universal.theme : Universal.Light / Universal.Dark
- Universal.accent : any color (Universal.Teal by default)

Creating our own style

- Make a copy of an existing style (i.e Material)
- Modify the copy following the guidelines : https://doc-snapshots.qt.io/qt5-5.6/qtlabscontrols-customize.html
- From your QML document, import the new style (optionally using an alias if one want to use the original controls at the same time)

```
import QtQuick 2.6
import Qt.labs.controls 1.0
import “../MyStyle/” as C

Pane {
    C.Label {
        width: parent.width
        text: "My Label"
    }
}
```

High-DPI support

Method 1 : setting AA_EnambleHighDpiScaling attribute before constructing QGuiApplication

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
    QGuiApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);
    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
    return app.exec();
}
```

Method 2 : setting environment variable **QT_AUTO_SCREEN_SCALE_FACTOR** to 1

see https://doc-snapshots.qt.io/qt5-5.6/highdpi.html

Hello World

main.cpp

main.qml

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
    QGuiApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);
    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
    return app.exec();
}
```

main.cpp

```
import QtQuick 2.6
import Qt.labs.controls 1.0

ApplicationWindow {
        title: "My Application"
        width: 640; height: 480
        visible: true

        Button {
                text: "Hello World!"
                anchors.centerIn: parent
                onClicked: Qt.quit()
        }
}
```

Controls overview

- containers
- boutons
- input controls
- menus
- indicators
- misc...

Containeurs

- ApplicationWindow
- Container
- Drawer
- Frame
- GroupBox
- Page
- Pane
- StackView
- SwipeView
- TabBar
- ToolBar

These controls are used for adding child controls.
It inherits from **Container**.

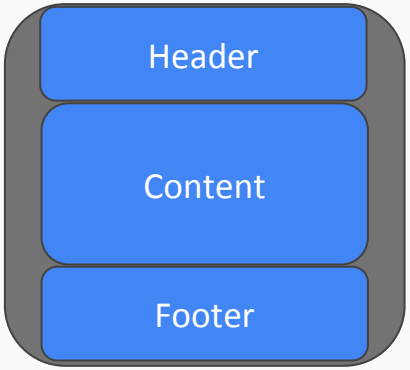
The root container is **ApplicationWindow**, it contains an header, a content and a footer.

Frame and **GroupBox** are used to visually group others controls.
A **GroupBox** is a **Frame** having a title.

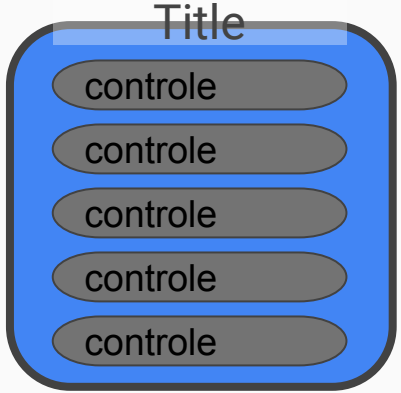
For a screen page, one can use a **Pane** or a **Page** (**Pane** being a **Page** plus a header and a footer)

StackView are used for stacking others containers, alternatively a **SwipeView** can be used to swipe between containers.

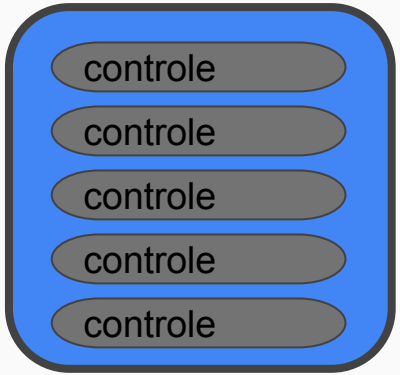
Finally the library also propose **TabBar** and **ToolBar**.



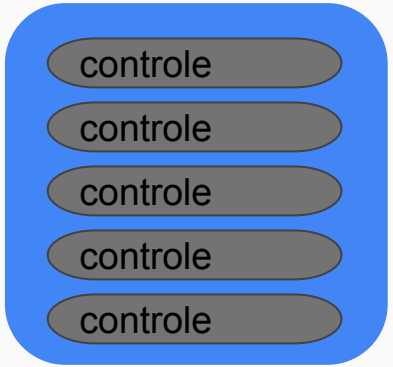
ApplicationWindow
Page



GroupBox

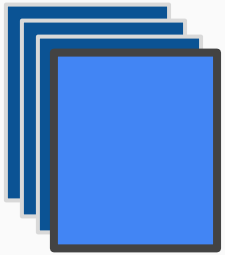


Frame

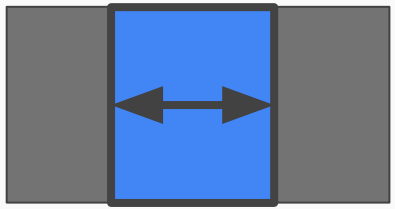


Pane

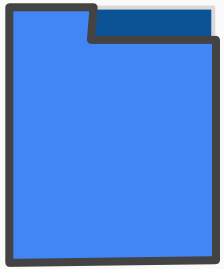
ToolBar



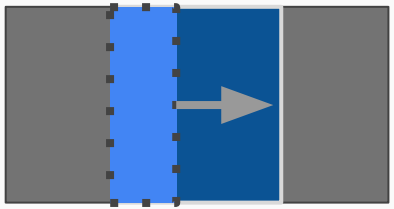
StackView



SwipeView



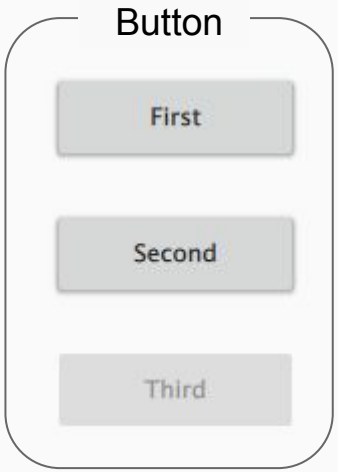
TabBar

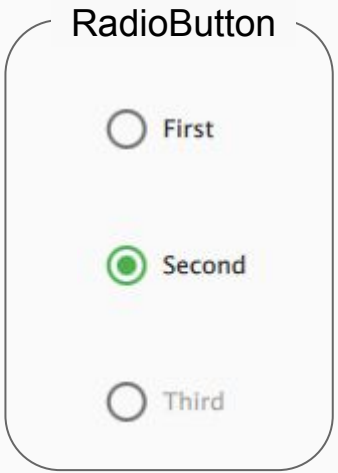


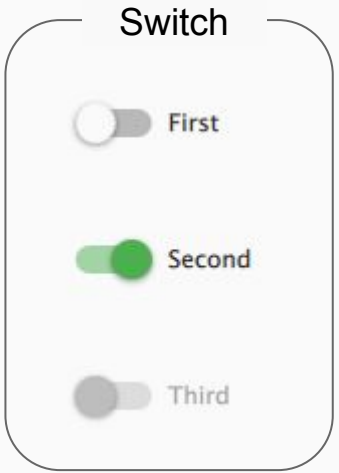
Drawer

Button Controls

- AbstractButton
- Button
- CheckBox
- RadioButton
- Switch
- ToolButton

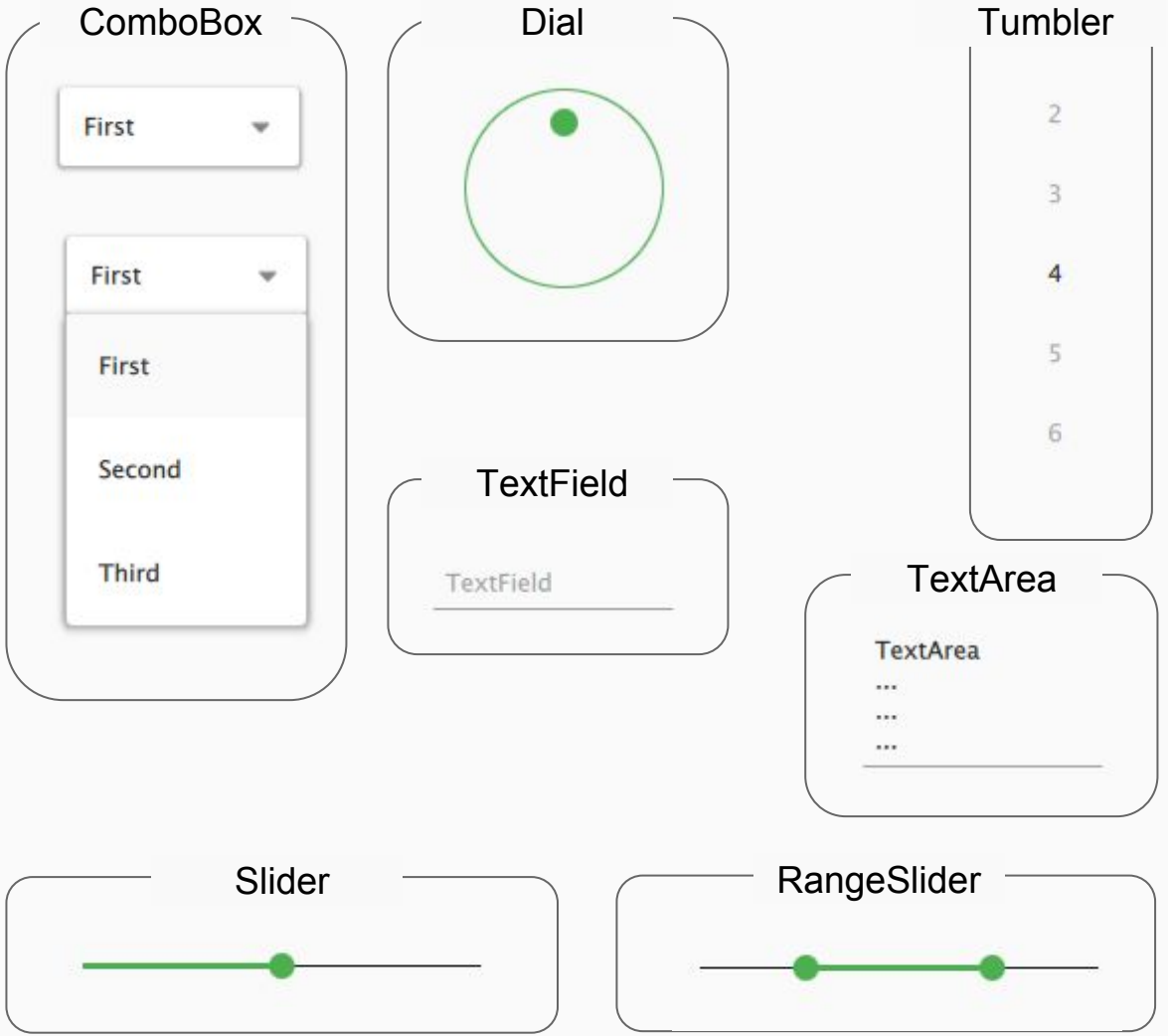






Input controls

- ComboBox
- Dial
- RangeSlider
- Slider
- TextArea
- TextField
- Tumbler



Menu controls

- Menu
- MenuItem

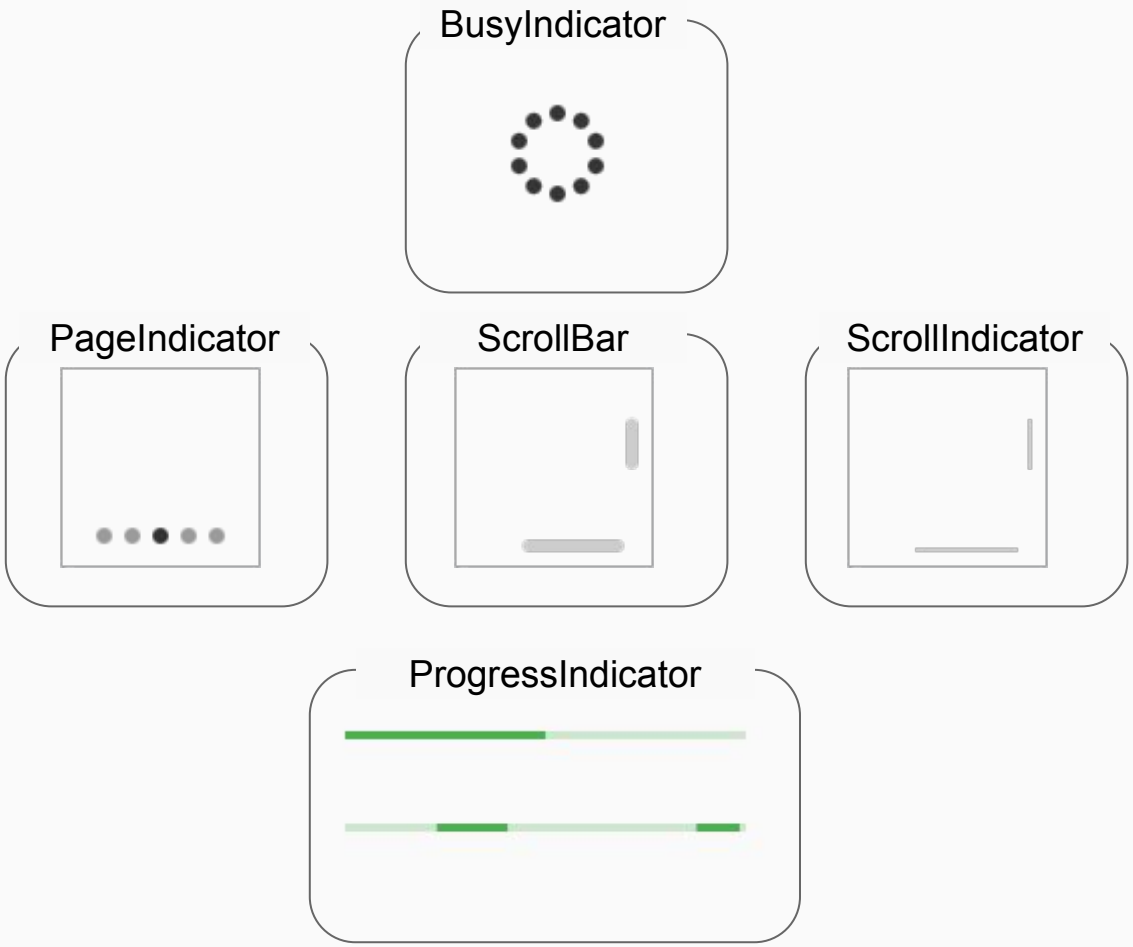
```
Button {
        id: button
        onClicked: menu.open()

        Menu {
                id: menu
                y: button.height

                MenuItem {
                        text: "New..."
                        onTriggerred:console.log(“triggerred”)
                }
        }
}
```

Indicator controls

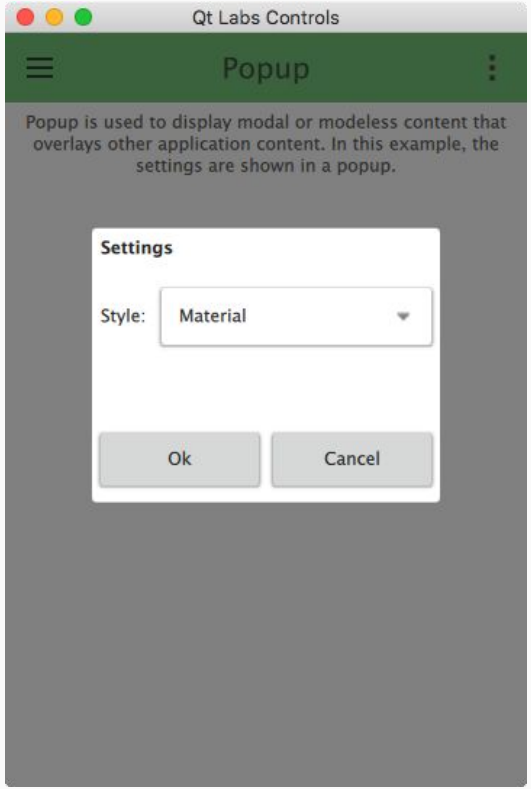
- BusyIndicator
- PageIndicator
- ProgressBar
- ScrollBar
- ScrollIndicator



Divers

- Popup
- Label
- Control
- ItemDelegate
- ButtonGroup
- SpinBox
- TabButton

- **TabButtons** can be used with TabBar
- **Label** are used to display text
- **Control** is the base element inhereted by the specialized controls.
- **ButtonGroup** is not visible but used to group interacting controls.
- **ItemDelegage** can be used with **ListView** and **ComboBox** to display a model item with a predefined behavior (a text, clickable..)

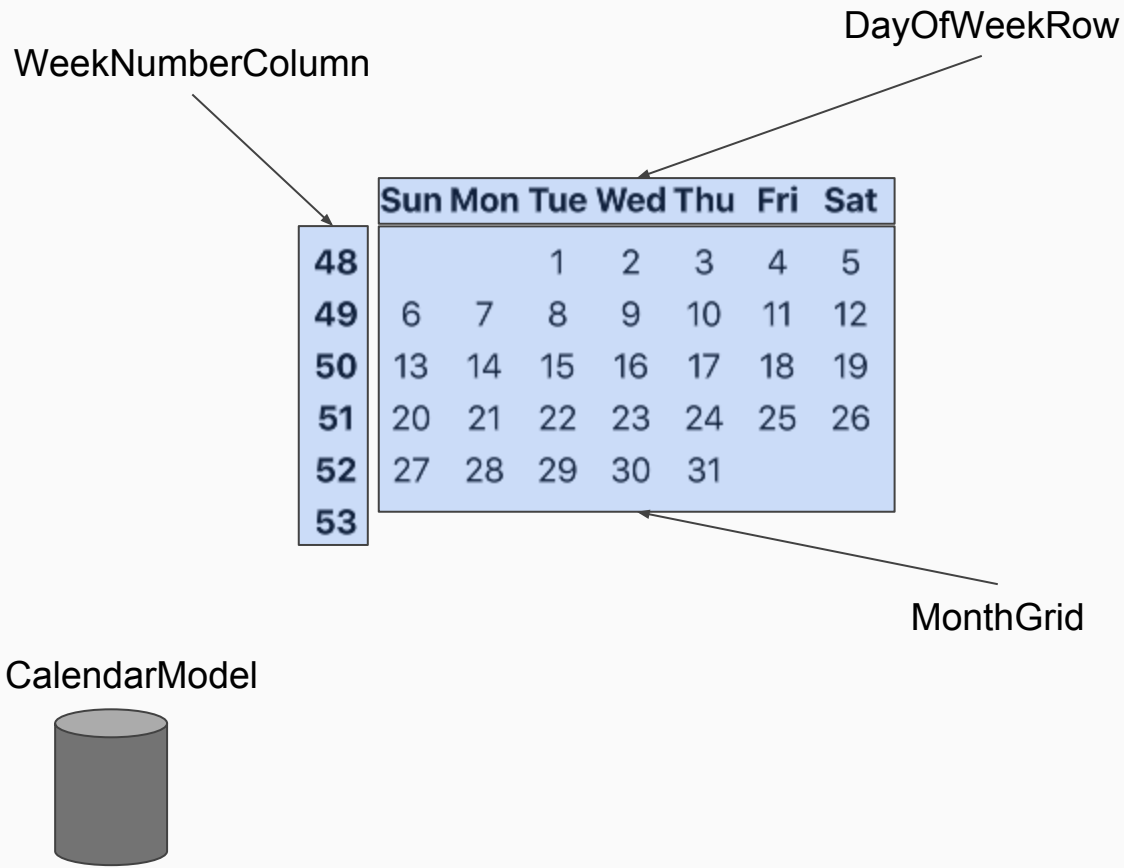


Popup

SpinBox

Qt.labs.calendar

- Calendar
- CalendarModel
- DayOfWeekRow
- MonthGrid
- WeekNumberColumn



Code Review

Now, let’s discover the gallery example

Guillaume Charbonnier
gcharbonnier@a-team.fr