

## Qt Quick Controls dans Qt 5.1

Dans la nouvelle architecture de Qt 5, la création d'interface utilisateur ne se focalise plus uniquement sur les Qt Widget et classes dérivées : l'utilisation de Qt Quick est fortement recommandée. Cependant, l'une des remarques faite sur Qt Quick est l'absence des widgets habituellement utilisés, ce qui nécessite de les recréer. Les Qt Quick Controls de Qt 5.1 répondent à ce besoin.

### 1 Importation des modules

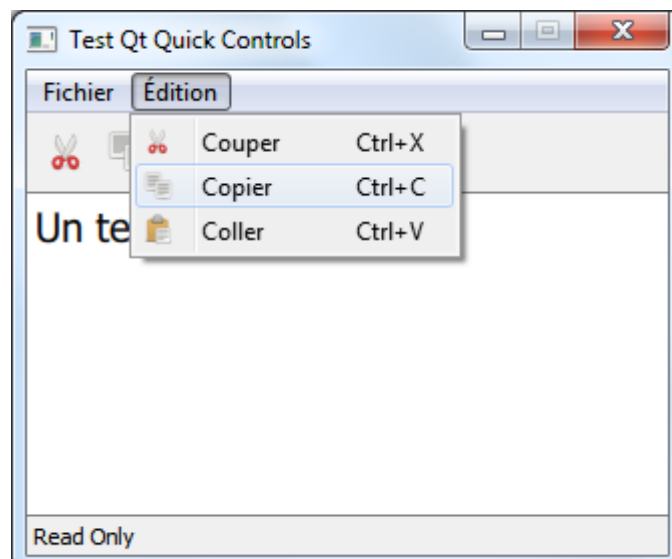
L'utilisation des Qt Quick Controls nécessite d'importer les modules Qt Quick 2.1 et Qt Quick Controls 1.0 :

```
import QtQuick 2.1
import QtQuick.Controls 1.0
```

## 2 Créer une fenêtre principale

Il est possible de créer une fenêtre principale (`ApplicationWindow`), avec des menus (`MenuBar`, `Menu` et `MenuItem`), une barre de statuts (`StatusBar`) et une barre d'outils (`ToolBar` et `ToolButton`).

```
ApplicationWindow {
    title: "Test Qt Quick Controls"
    menuBar: MenuBar {
        Menu {
            title: "Fichier"
            MenuItem { text: "Ouvrir..."; shortcut: "Ctrl+O" }
        }
    }
    toolBar: ToolBar {
        Row {
            ToolButton { iconSource: "Gnome-edit-cut.svg" }
        }
    }
    TextEdit {
        text: "Un texte quelconque..."
    }
    statusBar: StatusBar {
        Row {
            Label { text: "Read Only" }
        }
    }
}
```



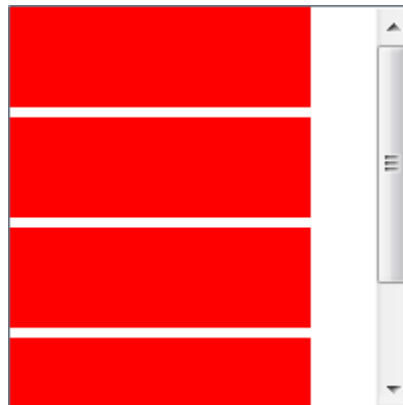
*Une fenêtre principale avec menus, barre d'outils  
et barre de statuts*

### 3 Créer des vues

#### 3.1 Créer une vue scrollable

Une vue scrollable permet d’afficher des éléments dont la taille est plus importante que celle de la fenêtre. Des barres de défilement sont automatiquement ajoutées si nécessaire pour déplacer verticalement et horizontalement le contenu de la vue.

```
ScrollView {  
    width: 200; height: 200  
    Column {  
        spacing: 5  
        Repeater {  
            model: 5  
            Rectangle {  
                color: "red"  
                width: 150; height: 50  
            }  
        }  
    }  
}
```

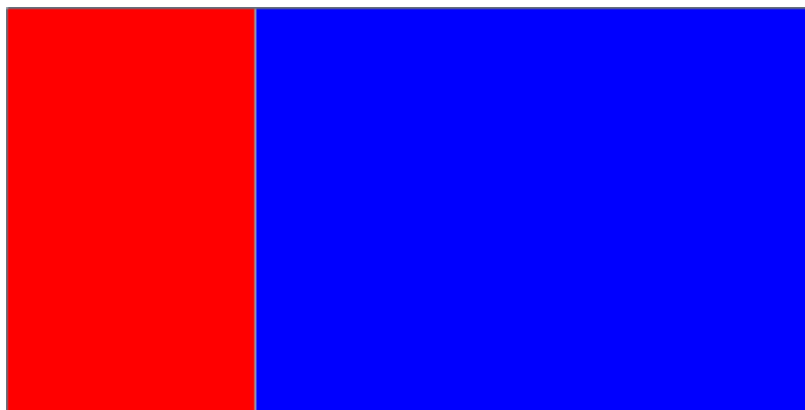


*Une vue scrollable*

## 3.2 Créer une vue splittée

Une vue splittée permet de découper horizontalement ou verticalement une fenêtre en plusieurs éléments. La taille des éléments est redimensionnable avec la souris en cliquant au centre, contrairement aux layouts.

```
SplitView {  
    width: 400; height: 200  
    orientation: Qt.Horizontal  
    Rectangle {  
        width: 200  
        color: "red"  
    }  
    Rectangle {  
        width: 200  
        color: "blue"  
    }  
}
```

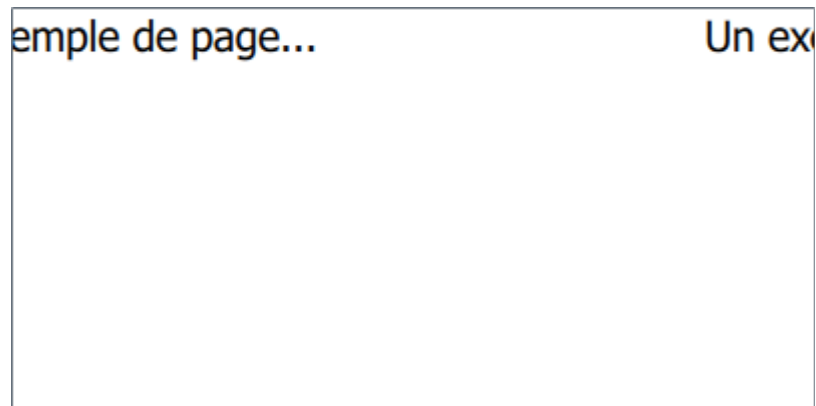


*Une vue splittée*

### 3.3 Créer une vue d'empilement

Un `StackView` permet d'empiler des éléments avec la fonction `push` et de les dépiler avec la fonction `pop`. À chaque fois qu'un élément est empilé, celui-ci devient l'élément visible, les éléments précédents étant masqués. Lorsqu'un élément est dépilé, l'élément en haut de la pile est affiché à nouveau. Une animation est lancée à chaque empilement ou dépilement.

```
StackView {
    width: 400; height: 200
    MouseArea {
        anchors.fill: parent
        onClicked: {
            push(component)
        }
    }
    Component {
        id: component
        Text {
            font.pointSize: 16
            text: "Un exemple de page..."
        }
    }
}
```

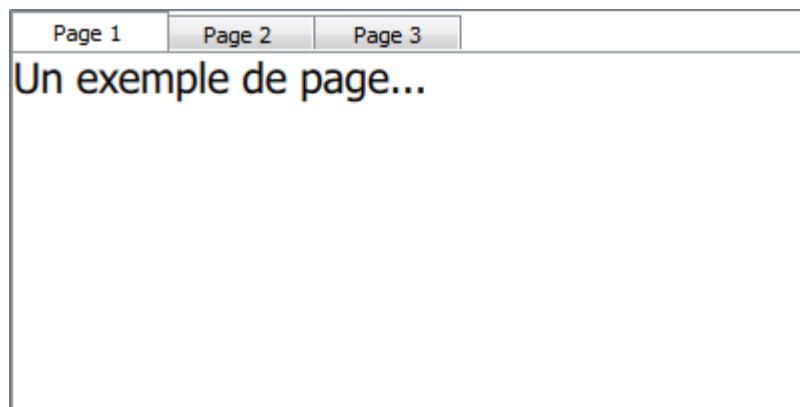


*Une vue d'empilement*

### 3.4 Créer une vue avec onglets

Pour créer une vue avec onglets, il faut ajouter les différentes pages avec la fonction `addTab`, qui prend un titre et un élément. L'utilisateur peut cliquer sur les onglets pour changer de page.

```
TableView {  
    width: 400; height: 200  
    Component.onCompleted: {  
        addTab("Page 1", component)  
        addTab("Page 2", component)  
        addTab("Page 3", component)  
    }  
    Component {  
        id: component  
        Text {  
            font.pointSize: 16  
            text: "Un exemple de page..."  
        }  
    }  
}
```



*Une page avec onglets*

### 3.5 Créer une table

Une `TableView` permet de créer une table similaire à `QTableView` de Qt Widget.

```
TableView {  
    width: 300; height: 200  
    TableViewColumn{ role: "polygone" ; title: "Polygone" ; width: 100 }  
    TableViewColumn{ role: "cotes" ; title: "Côtés" ; width: 100 }  
    model: ListModel {  
        ListElement{ polygone: "Triangle"; cotes: 3 }  
        ListElement{ polygone: "Rectangle"; cotes: 4 }  
        ListElement{ polygone: "Pentagone"; cotes: 5 }  
        ListElement{ polygone: "Hexagone"; cotes: 6 }  
    }  
}
```

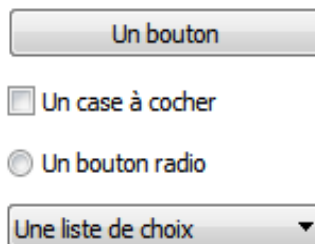
Polygone	Côtés	
Triangle	3	
Rectangle	4	
Pentagone	5	
Hexagone	6	

*Une table*

## 4 Ajouter des widgets

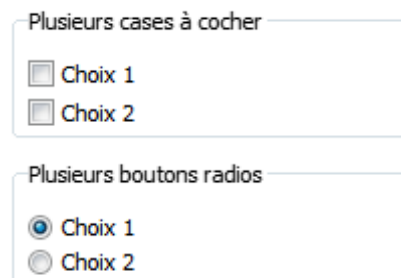
Le code suivant permet de créer un bouton, une case à cocher, un bouton radio et une liste de choix :

```
Button { text: "Un bouton" }
CheckBox { text: "Un case à cocher" }
RadioButton { text: "Un bouton radio" }
ComboBox { model: [ "Une liste de choix", "Autre choix" ] }
```



Les `GroupBox` permettent de regrouper plusieurs éléments :

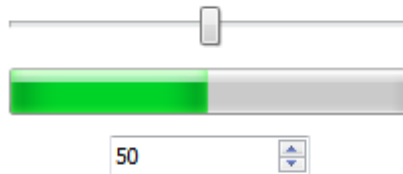
```
GroupBox {
    title: "Plusieurs cases à cocher"
    Column {
        spacing: 2
        CheckBox { text: "Choix 1" }
        CheckBox { text: "Choix 2" }
    }
}
GroupBox {
    title: "Plusieurs boutons radios"
    Column {
        ExclusiveGroup { id: group }
        RadioButton {
            text: "Choix 1"
            exclusiveGroup: group
            checked: true
        }
        RadioButton {
            text: "Choix 2"
            exclusiveGroup: group
        }
    }
}
```





Pour sélectionner ou afficher une valeur :

```
Slider {  
    minimumValue: 0  
    maximumValue: 100  
    value: 50  
}  
ProgressBar {  
    minimumValue: 0  
    maximumValue: 100  
    value: 50  
}  
SpinBox {  
    minimumValue: 0  
    maximumValue: 100  
    value: 50  
}
```



Et pour terminer, des éléments pour afficher du texte, similaires à `Text`, `TextInput` et `TextEdit` :

```
Label {  
    text: "Un texte non modifiable"  
    font.pixelSize: 18  
    font.italic: true  
    color: "steelblue"  
}  
TextArea { text: "Un texte sur\nplusieurs lignes, modifiable" }  
TextField { width: 100; text: "Un texte sur une ligne, modifiable" }
```

*Un texte non modifiable*

Un texte sur  
plusieurs lignes, modifiable

Un texte sur une ligne, modifiable

## 5 Pour en savoir plus...

*Cet article est une simple vue d'ensemble des Qt Quick Controls, extrait du livre [Créer des applications avec Qt 5](#). N'hésitez pas à faire des retours sur ce que vous en pensez.*



*Les images SVG fournies avec les codes d'exemple proviennent de [WikiMedia Common](#) : [Category:GNOME Desktop icons, actions](#).*

*Merci à [winjerome](#) pour sa relecture orthographique.*