Basic UI Components for Qt and WP7

This article demonstrates how to create some basic UI components, including menus, toolbars, buttons, progress bars, checkboxes, etc., in both Qt and WP7.

Introduction

This article shows how to create some of the basic UI components in both Qt and WP7. It is intended primarily for developers who are familiar with Qt and want to start learning Windows Phone 7.

The UI elements that we demonstrate in Qt are `TextBox`, `ProgressBar`, `Button`, `RadioButton`, `CheckBox`, `MenuBar`. Once we are done with each component in Qt we show how to create the same UI elements for Windows Phone 7 using XAML.

Qt UI Components

WP7 UI Components

Implementation

Application Menu Bar

Qt Project

This is the `ToolBar` in Qt with `ToolButton` which provides a different style of appearance compared to a normal `Button`.

```
// tool bar
    ToolBar {
         anchors.bottom: parent.bottom
         tools: ToolBarLayout {
             ToolButton {
                 iconSource: "toolbar-back"
                  onClicked: pageStack.depth <= 1 ? Qt.quit() : pageStack.pop()
             }
             ToolButton {
                 iconSource: "toolbar-search"
                 onClicked: mainText.text ="Search"
             }
             ToolButton {
                 iconSource: "toolbar-refresh"
                 onClicked: mainText.text = "Refresh "
             }
             ToolButton {
                 iconSource: "toolbar-menu"
                 onClicked: (myMenu.status == DialogStatus.Closed) ? myMenu.open() : myMenu.close()
             }
         }
     }
```

WP7 project

The XAML code below provides an Application Bar that has four icon buttons using `ApplicationBarIconButton` Element.

(C) Copyright Nokia 2013. All rights reserved.

```
<!--Sample code showing usage of ApplicationBar-->
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton x:Name="BackButton" IconUri="/Images/appbar.back.rest.png" Text="Back" Click="BackButton
            <shell:ApplicationBarIconButton x:Name="SearchButton" IconUri="/Images/appbar.feature.search.rest.png" Text="Search" Cli
            <shell:ApplicationBarIconButton x:Name="RefreshButton" IconUri="/Images/appbar.refresh.rest.png" Text="Refresh" Click="R
            <shell:ApplicationBarIconButton x:Name="FavButton" IconUri="/Images/appbar.favs.rest.png" Text="favourite" Click="FavBut
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>
```

Application Menu Item

In addition to the icons in the Menu Bar we can also add text-based menu items. These menu items are displayed in a list view.

Qt Project

This is the Menu Item that has been added to the Application Menu Bar, which can be used for accessing particular function.

```
    //menu items
    Menu {
        id: myMenu
        visualParent: pageStack
        MenuLayout {
            MenuItem
            {
                text: qsTr("Menu1")
                onClicked: mainText.text = "Menu1 is Clicked"
            }
            MenuItem{
                text:qsTr("Menu2")
                 onClicked: mainText.text = "Menu2 is Clicked"
            }
            MenuItem{
                text:qsTr("Menu3")
                platformSubItemIndicator: true
                onClicked: subMenu.open()
            }
        }
    }
    // sub menu
    ContextMenu {
        id: subMenu
        MenuLayout {
            MenuItem {
                text: "Red"
                onClicked: mainText.text = "Red is Clicked"
            }
            MenuItem {
                text: "Green"
                onClicked: mainText.text = "Green is Clicked"
            }
            MenuItem {
                text: "Blue"
                onClicked:mainText.text = "Blue is Clicked"
            }
        }
    }
```

WP7 project

The below WP7 code in XAML creates three menu items in the menu bar.

```
<!--Sample code showing usage of ApplicationBar-->
    <phone:PhoneApplicationPage.ApplicationBar>
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem x:Name="Menu1" Text="Menu 1" Click="Menu1_Click"/>
                <shell:ApplicationBarMenuItem x:Name="Menu2" Text="Menu 2" Click="Menu2_Click"/>
                <shell:ApplicationBarMenuItem x:Name="Menu3" Text="Menu 3" Click="Menu3_Click"/>
            </shell:ApplicationBar.MenuItems>
     </phone:PhoneApplicationPage.ApplicationBar>
```

Note: Here to show the difference between the code structure, I have put the Menu Bar and Menu Item code seperately, but both these codes are combined together and interrelated in the project source code

Text Field

Text file is used to get text from user input.When user clicks on the text filed area, virtual keyboard pops up and allows user to edit text on it. This is a single line of editable plain text.

Qt project

```
// text box
    TextField {
        anchors.top:statusBar.bottom
        anchors.left: parent.left
        anchors.right: parent.right
        id:textField
        width: 160
         placeholderText: "Enter Text Here"
        validator: IntValidator{bottom: 0; top: 100}
    }
```


(C) Copyright Nokia 2013. All rights reserved.

WP7 project

```
<TextBox Height="72" HorizontalAlignment="Left" Margin="39,26,0,0" Name="textBox1" Text="Enter Text Here" VerticalAlignment="Top" Wi
```

Progress Bar

The progress bar that we have used in the section are indeterminate progress bar (they show that progress is occurring, but not how much is complete/remaining).

Qt project

```
ProgressBar {
        anchors.top:textField.bottom
        anchors.left: parent.left
        anchors.right: parent.right
         id: progressBar
         indeterminate: true
         visible: simpletimer.running
     }
```

WP7 project

```
<ProgressBar Height="5" HorizontalAlignment="Left" Margin="29,121,0,0" Name="progressBar1" VerticalAlignment="Top" Width="391" />
```

Button

We will create a button with text inside. On click of the button we will call the event handler to perform some task.

Qt project

```
   //button
     Button {
                id: button
                anchors.top: progressBar.bottom
                anchors.left: parent.left
                anchors.right: parent.right
                text: "Signals"
                onPlatformReleased: {
                    mainText.text =  "Button Released"
                }
                onClicked: {
                    mainText.text =  "Button Clicked"
                }
                onPlatformPressAndHold: {
                    mainText.text ="Button Press and Hold"
                }
            }
```

WP7 project

```
<Button Content="Button" Height="72" HorizontalAlignment="Left" Margin="29,152,0,0" Name="button1" VerticalAlignment="Top" Width="39
```

Radio Button

Lets create a `RadioButton` with a line of Text. When user select the radio button we catch the event and process it.

Qt project

```
     // radio button
     Row {
         anchors.top:button.bottom
         id: radioButton
        anchors.left: parent.left
         RadioButton {
             id: button1
             text: "Radio Button"
             checked:  false
             onClicked: if(checked) {mainText.text = "Radio Checked"} else {mainText.text ="Radio Un-Checked"}
         }
     }
```

WP7 project

```
<RadioButton Content="RadioButton" Height="72" HorizontalAlignment="Left" Margin="29,251,0,0" Name="radioButton1" VerticalAlignment=
```


(C) Copyright Nokia 2013. All rights reserved.

Check Box

CheckBox returns whether the box is checked or not, and the events are handled.

Qt project

```
//CheckBox
    CheckBox {
        anchors.top: button.bottom
        anchors.right: parent.right
         id: checkBox
         text: "Check Box "
         checked: false
         onClicked: if(checked) {mainText.text ="Check Box Checked"} else {mainText.text = "Check Box Un-Checked"}
     }
```

WP7 project

```
  <CheckBox Content="CheckBox" Height="72" HorizontalAlignment="Left" Margin="255,251,0,0" Name="checkBox1" VerticalAlignment="Top"
```

Text Block

This is a constant block of text, that has no option to edit by user input. In our case we display the selected control name in this Text Box.

Qt project

```
Text {
        id:mainText
        anchors.centerIn: parent
        text: qsTr("Hello world!")
        color: platformStyle.colorNormalLight
        font.pixelSize: 20
    }
```

WP7 projet

```
   <TextBlock x:Name="PageTitle" Text="hello world !" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
```

Note: When user clicks on any of the UI Control, we catch the event and display the name of the control in the Text Box.

Source Code

- The full source code for Qt example is available here: File:UIComponentQt.zip
- The full source code for WP7 example is available here: File:UIComponentWP7.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.