Two ways to control rotation in QML

This article explains how to control the orientation of your app in Qt Quick. Please see the "See also" which provides links to official information on this topic.

Introduction

I was hunting around the web for an easy way to do rotation in Qt apps, and if possible, QML for an N9 app I'm writing. I ended up trying various solutions, including using a C++ class and Qt Mobility to handle it and trying to call QtMobility from within QML. These are things people have said would work and have used it seems, but I was sure there had to be a simpler way. Now I'm not claiming the way I ultimately used was anything new, and I'm sure many QML app developers have used it before, but once I found it, it was so simple. That crucial point is that you create your new app with the base QML object as a PageStackWindow.

PageStackWindow

The default window for QML apps for Harmattan and Belle is now a `PageStackWindow`, and doing this gives you some inner objects right out of the bat – a Page called mainPage, and then within the `PageStackWindow` QML itself, a ToolBarLayout for you to set up the standard app toolbar. You don't need to use either of these (although you do need at least one Page for the PageStackWindow) and you can create a a single screen, full screen app just as easily as you can a multi-screen one with the toolbar.

The key to why use a `PageStackWindow`, even if you don't want a multi-page app, is one simple property: `inProtrait`. (This Property is inherited from Window, but the code in this article shows usage with `PageStackWindow`)

Property on Page to control a state

So this gives you 2 options: one is to create a Boolean property on your Page and then in your `PageStackWindow`, assign that property the value of `inProtrait`. In your page you can then use that property to control a State. Here is the `PageStackWindow` code for this:

```
PageStackWindow {
 id: window
 initialPage: myPage
 showStatusBar: true
 showToolBar: true

 ToolBarLayout {
     id: toolBarLayout
         ToolButton {
            flat: true
            iconSource: "toolbar-back"
            onClicked: window.pageStack.depth <= 1 ? Qt.quit() : window.pageStack.pop()
         }
    }
     MainPage {
        id:myPage
        tools: toolBarLayout
        portrait: inPortrait
    }
}
```

Two Pages using inPortrait and PageStack.Replace to switch between them

The alternative is if you have different layouts or options in Landscape compared to Portrait that would make it hard to control/re-layout using states. In this case, create 2 Pages, and switch between them when inPortrait Changes, as well as setting the initial Page based on orientation. In this example I am not using the toolbars provided by default. The `PageStack.Replace` method

switches the current page at the top of the stack.

```
PageStackWindow {
id: appWindow

    onInPortraitChanged: {
        if (inPortrait && pageStack.currentPage!==mainPage)
            pageStack.replace(mainPage);
        else if (!inPortrait && pageStack.currentPage!==mainPageLandscape)
            pageStack.replace(mainPageLandscape);
    }

    initialPage: inPortrait ? mainPage : mainPageLandscape

    PortraitPage {
        id: mainPage
    }

    LandscapePage {
        id: mainPageLandscape
    }
}
```

Summary

So there you go, a simple ways to control rotation all in QML, allowing your application logic / view models to not have to worry about screen layout and device rotation. I also published this article at Controling rotation in QML – looked hard, but was easy.