使用stackView来做focus导航，详细代码在官方文档“touch”

现在我们开始仿照官网的demo做一个小东西。

开始之前先看效果图



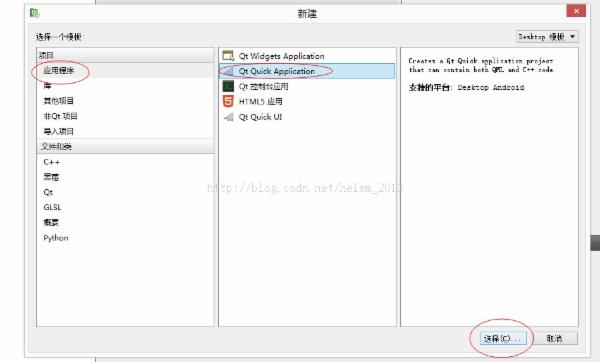
android如下：



下面开始正文：

因为在开始搞这个demo的时候，我遇到一个问题，所以为了避免这个，我将从创建项目开始。

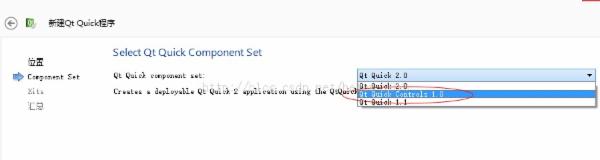
OK，废话不多说，第一步，打开你的QtCreator,选择 文件-新建文件或项目-应用程序-Qt Quick Application -选择



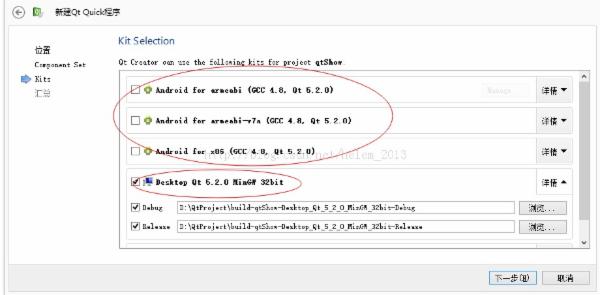
输入项目名称和路径，下一步



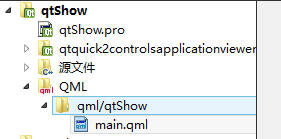
这一步需要注意的是，选择，我第一次创建的时候选择了Qt Quick2.0,但是项目中用到了Controls，虽然我在项目中引入了包，但是运行的时候一直没有出效果，而且会报引入出错的问题，然后我再重新创建一个项目，引入这个，就OK了，大家看需求弄，不过我这里是需要Controls的，所以选Qt Quick Controls 1.0



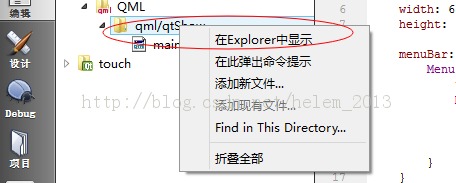
然后下一步，选android的时候根据版本来，桌面也勾选，毕竟测试用的



根据需求完成后点击完成，下面就是我们的项目

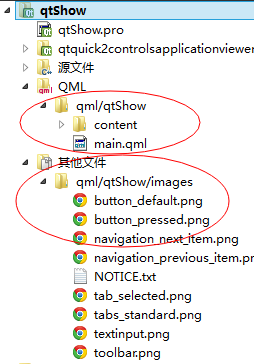


然后，选中qml/yourProject文件夹，右击，在Explorer中显示



然后在打开的文件夹中创建一个文件夹content和一个images文件夹，把你的图片资源拷贝到images文件夹下

然后看看你的QtCreator里面的项目结构



双击main.qml,清除里面的代码

然后在里面输入如下代码

//main.qml

import QtQuick 2.1

import QtQuick.Controls 1.0

import "content"//因为使用了文件目录，所以要引入

//使用ApplicationWindow前，必须引入QtQuick.Controls包

//ApplicationWindow提供了一个顶级应用程序窗口。

/\*\*

\* ApplicationWindow里面有3个子元素分别为toolBar，statusBar，menuBar

\* 从子元素的内容可以知道，它是为了方便界面功能块划分的

\* 详细请参照API

\*/

ApplicationWindow {

width: 360

height: 600

//在界面定义一个矩形，颜色黑色，大小为填满父组件

Rectangle {

color: "#212126"

anchors.fill: parent

}

//实现返回功能

//监听按钮方法

Keys.onReleased: {

//当事件的来源为返回键，当当前显示页为主页按返回则退出，反之则显示第一页

if (event.key === Qt.Key\_Back) {

//当前处于顶端的不是第一个

//stackView的depth属性是只读的

if (stackView.depth > 1) {

//stackView的pop事件，详情看API，参数为空表示去第一页

stackView.pop();

//可继续操作

event.accepted = true;

} else { Qt.quit(); }//退出

}

}

//StackView提供了一个基于堆栈的导航模型

StackView {

id: stackView //id必须唯一哦

anchors.fill: parent //填充父组件的大小

//设置初始化StackView界面

initialItem: Item {

//宽高都继承自父组件

width: parent.width

height: parent.height

//内容为列表

ListView {

model: pageModel//使用id为pageModel的model

anchors.fill: parent//充满父组件

//实例化列表，为列表填充内容

delegate: AndroidDelegate {

text: title//内容标题

//对应的内容页

onClicked: stackView.push(Qt.resolvedUrl(page))

}

}

}

}

//列表的model

ListModel {

id: pageModel

//列表元素

ListElement {

title: "按钮"

page: "content/ButtonPage.qml"

}

ListElement {

title: "滑动条"

page: "content/SliderPage.qml"

}

ListElement {

title: "进度条"

page: "content/ProgressBarPage.qml"

}

ListElement {

title: "导航条"

page: "content/TabBarPage.qml"

}

ListElement {

title: "输入框"

page: "content/TextInputPage.qml"

}

ListElement {

title: "列表"

page: "content/ListPage.qml"

}

}

//接着继续stackView未完的代码

//导航

toolBar: BorderImage {//还记得么，前面我们讲过，当对象只有一个对象的时候可以这么写

border.bottom: 8//底部边框

source: "images/toolbar.png"//图片

width: parent.width//宽继承自父组件的宽

height: 60//高

//返回按钮标示

Rectangle {

id: backButton//id

width: opacity ? 50 : 0//不透明的时候为60，透明为0

anchors.left: parent.left//在父组件的左边描点

anchors.leftMargin: 20//左边距

opacity: stackView.depth > 1 ? 1 : 0//界面不为主页的时候显示，为主页则不显示

anchors.verticalCenter: parent.verticalCenter//相对于父组件垂直居中

antialiasing: true

height: 45//高

radius: 4//边角

color: backmouse.pressed ? "#222" : "transparent"//点击的时候显示色

Behavior on opacity { NumberAnimation{} }//透明度的变化

Image {

anchors.verticalCenter: parent.verticalCenter//图片垂直居中

source: "images/navigation\_previous\_item.png"//左箭头图片

}

MouseArea {

id: backmouse

anchors.fill: parent//相对于矩形图片区域描点

anchors.margins: -10//边距

onClicked: stackView.pop()//stackView返回操作，stackView的第一页

}

}

Text {

font.pixelSize: 25//字体大小

Behavior on x { NumberAnimation{ easing.type: Easing.OutCubic} }//x轴的变化

x: backButton.x + backButton.width + 20//在x轴的位置

anchors.verticalCenter: parent.verticalCenter//垂直居中

color: "white"//颜色

text: "示例"//文字

}

}

}

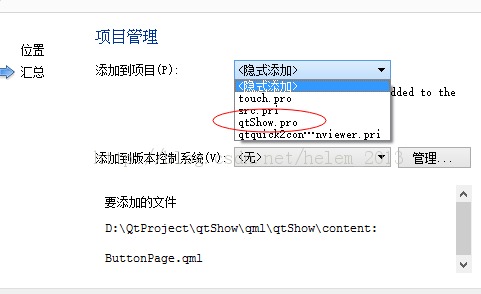
完成之后，额，发现有一个地方报错了，就是关键字AndroidDelegate下面有红色报错提示，那表示代码正常，只是你需要加入一个名叫AndroidDelegate文件

下面去创建这个文件

单击QtCreator 导航的文件- 新建文件或项目-文件和类-选Qt - QML File (Qt Quick 2)-选择（图略）

输入文件名-AndroidDelegate.qml 路径注意是content文件夹里面（点浏览-选content）下一步

注意，要添加到你的项目中去



然后完成

输入下面代码，

//AndroidDelegate.qml

import QtQuick 2.0

Item {

id: root

width: parent.width //继承父组建的宽度

height: 60

property alias text: textitem.text //变量

signal clicked

//透明矩形，点击的时候可见

Rectangle {

anchors.fill: parent//描点充满父组建

color: "#11ffffff"//透明

visible: mouse.pressed//点击可见

}

//文本

Text {

id: textitem

color: "white"//字体颜色

font.pixelSize: 24//字体大小

text: modelData//显示的文字

anchors.verticalCenter: parent.verticalCenter//父组件垂直居中

anchors.left: parent.left//靠左

anchors.leftMargin: 30//左边距

}

//黑色矩形

Rectangle {

anchors.left: parent.left//描点位置为父组建左侧

anchors.right: parent.right//右侧

anchors.margins: 15//外边距

height: 1//高

color: "#424246"//黑色

}

//图片

Image {

anchors.right: parent.right//父组建描点

anchors.rightMargin: 20//右边距20

anchors.verticalCenter: parent.verticalCenter//垂直居中

source: "../images/navigation\_next\_item.png"//图片路径

}

//点击响应

MouseArea {

id: mouse

anchors.fill: parent//描点充满整个父组建

onClicked: root.clicked()//父组建点击响应

}

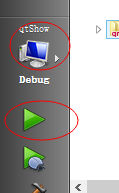
}

完成之后点击运行

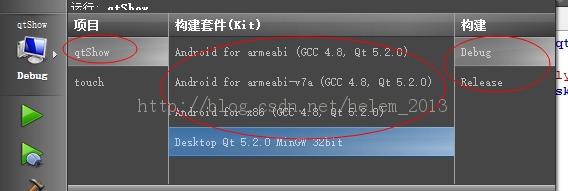


如果要看android版

请看QtCreator右下角的



点击电脑标志，选择就行了



稍等片刻，点击运行，就可以看到手机效果了（前提是你的配置没问题）



点击



转场



因为其他界面文件，我们还没写，所以这里显示的就这样了

如需添加，请参照下面代码，因为时间关系，本人只写了其中的一个文件，其他的大家可以参考官方的源码（这个是根据源码改的，官方的项目叫touch）

//ButtonPage.qml

/\*\*

\* 列表的按钮进入的界面

\* 后面我会在这个的基础上改一些东西

\*/

import QtQuick 2.1

import QtQuick.Controls 1.1

import QtQuick.Controls.Styles 1.1

//

Item {

width: parent.width//继承父组建的宽

height: parent.height//继承父组建的高

property real progress: 0//自定义一个属性

//动画效果:SequentialAnimation 作用是序列化多个动画，直白的说就是让多个动画依次发生

SequentialAnimation on progress {

loops: Animation.Infinite //loops: 指定循环次数，Animation.Infinite表示无限循环

running: true//运行状态

NumberAnimation {

from: 0

to: 1//从0到1

duration: 3000//持续时间3秒

}

NumberAnimation {

from: 1

to: 0

duration: 3000

}

}

Column {

spacing: 40

anchors.centerIn: parent//父元素的中心

Button {

text: "点击我，我不动"

style: touchStyle

}

Button {

style: touchStyle

text: "点击我，我也不动"

}

Button {

anchors.margins: 20

style: touchStyle

text: "点击我会返回哦"

onClicked: if (stackView) stackView.pop()

}

//布局

Row {

spacing: 20

Switch {

style: switchStyle

}

Switch {

style: switchStyle

}

}

}

//组件

Component {

id: touchStyle

ButtonStyle {

panel: Item {

implicitHeight: 50

implicitWidth: 320

BorderImage {

anchors.fill: parent

antialiasing: true

border.bottom: 8

border.top: 8

border.left: 8

border.right: 8

anchors.margins: control.pressed ? -4 : 0

source: control.pressed ? "../images/button\_pressed.png" : "../images/button\_default.png"

Text {

text: control.text

anchors.centerIn: parent

color: "white"

font.pixelSize: 23

renderType: Text.NativeRendering

}

}

}

}

}

Component {

id: switchStyle

SwitchStyle {

groove: Rectangle {

implicitHeight: 50

implicitWidth: 152

Rectangle {

anchors.top: parent.top

anchors.left: parent.left

anchors.bottom: parent.bottom

width: parent.width/2 - 2

height: 20

anchors.margins: 2

color: control.checked ? "#468bb7" : "#222"

Behavior on color {ColorAnimation {}}

Text {

font.pixelSize: 23

color: "white"

anchors.centerIn: parent

text: "开"

}

}

Item {

width: parent.width/2

height: parent.height

anchors.right: parent.right

Text {

font.pixelSize: 23

color: "white"

anchors.centerIn: parent

text: "关"

}

}

color: "#222"

border.color: "#444"

border.width: 2

}

handle: Rectangle {

width: parent.parent.width/2

height: control.height

color: "#444"

border.color: "#555"

border.width: 2

}

}

}

}