Introducción a QML – Primer ejemplo I

Como sabéis la mejor estrategia para crear una aplicación en cualquier plataforma es el **“divide y vencerás”** y eso es lo que haremos para nuestro primer ejemplo de la serie.

En este primer ejemplo no seremos demasiado estrictos en la implementación ya que es pequeño y no requiere de demasiado orden.

Fondo de aplicación

Empezamos creando un fichero para el fondo del aplicación (**Backround.qml**)

```
import Qt 4.7

Rectangle {
    id: background
    anchors.fill: parent;
    color: "#343434"
    Image {
        source: "images/stripes.png"
        fillMode: Image.Tile
        anchors.fill: parent
        opacity: 0.3
    }
}
```

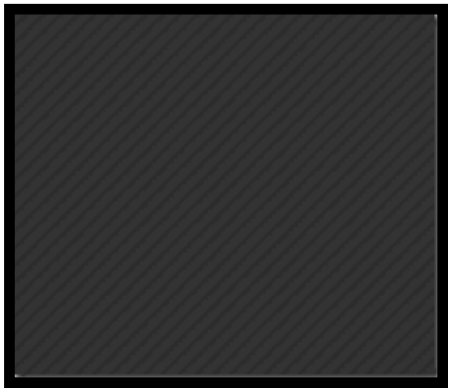
Hemos creado un elemento con identificativo **background**. Como color de fondo ponemos un gris y fijamos el anclaje.

Este elemento contendrá un elemento Imagen:

Establecemos la opacidad a 0.3 (**opacity**).

También establecemos que el modo de rellenado (**Image.Tile**).

Todo ello formará un fondo en forma de una imagen repetida angularmente:



Botones para comandos

Creamos otro fichero para los botones de comandos de nuestra aplicación.

```
import Qt 4.7

Rectangle {
    id: container
    property string colorString: "#343434"
    property alias text: label.text

    color: colorString
    smooth: true

    signal clicked

    Background {
        id: background1
    }

    MouseArea {
        id: mouseArea
        anchors.fill: parent
        onClicked: container.clicked()
    }

    BorderImage {
        id: buttonImage
        source: "images/toolbutton.sci"
        width: container.width
        height: container.height
        anchors.centerIn: parent
    }

    Text {
        id: label
        width: 0
        font.bold: true
        style: Text.Raised
        styleColor: "black"
        font.pixelSize: 12
        color: "#ffffff"
        transformOrigin: "Center"
        anchors.verticalCenterOffset: 2
        anchors.horizontalCenterOffset: 2
        anchors.centerIn: parent
        smooth: true
        verticalAlignment: "AlignVCenter"
        horizontalAlignment: "AlignHCenter"
    }
}
```

Creamos una propiedad pública llamada **colorString** (contendrá el color del fondo para concordarlo en toda la aplicación)

Establecemos el fondo en base al elementos definido en el fichero (Background.qml) creado en el paso anterior.

Establecemos un área para la gestión del clic del ratón (**MouseArea**)

Establecemos una imagen para el borde del botón (**BorderImage**)

Creamos una etiqueta para el texto del mensaje (**Text**).

Establecemos un alias para el texto de la etiqueta (**property alias text: label.text**)

Por último creamos una señal llamada **clicked** y la disparamos cuando se pulsa clic sobre mouseArea (**onClicked: container.clicked()**).

Toolbar de acciones

Ahora vamos a crear una fila de botones en forma de toolbar. Tendrá n botones en modo fila (Row). Cada uno de los botones dispondrá de una imagen representativa que tiene que ser parametrizable en cada botón. El resultado que pretendemos es:

Parece lógico que es necesario separar lo que es la definición del botón de la definición de la toolbar, por lo que crearemos dos ficheros **ButtonsRect.qml** y **ButtonsArea.qml**

ButtonsRect.qml

```
import Qt 4.7

Rectangle {
    id: rootArea
    property string colorString: "#343434"
    property alias imageString: image1.source

    property Item rootRect
    color: colorString
    radius: 16

    border.color: "#FFFFFF"; border.width: 2

    width: 96; height: 96;

    MouseArea {
        id: mousearea1
        anchors.fill: parent
        BorderImage {
            id: buttonImage
            width: parent.width
            height: parent.height
            Image {
                id: image1
                smooth: true
                scale: 0.7
                width: parent.width
                height: parent.height
                source: imageString
            }
        }
    }
}
```

Destacamos:

La propiedad **scale** permite reducir el tamaño de la imagen un 70% para más adelante conseguir un efecto de estirado al pasar sobre ella

La propiedad **smooth** suaviza el pixelado de la imagen al ser reducida.

Como con la propiedad imageString exponemos la imagen del botón

Creamos un borde en el rectángulo de ancho 2 y en blanco

ButtonsArea.qml

```
import Qt 4.7

Rectangle {
    id: rootRect
    property string selectedColor: "#343434"

    width: 606; height: 96; color: selectedColor; radius: 4

    anchors.margins: 5

    Background {
        id: background1
    }

    Row {
        spacing: 5
        ButtonsRect { imageString: "icos/diagram_v2-20.png"; rootRect: rootRect }
        ButtonsRect { imageString: "icos/diagram_v2-21.png"; rootRect: rootRect  }
        ButtonsRect { imageString: "icos/diagram_v2-22.png"; rootRect: rootRect  }
        ButtonsRect { imageString: "icos/diagram_v2-23.png"; rootRect: rootRect  }
        ButtonsRect { imageString: "icos/diagram_v2-24.png"; rootRect: rootRect  }
        ButtonsRect { imageString: "icos/diagram_v2-25.png"; rootRect: rootRect  }
    }
}
```

Destacamos:

Como con Row creamos una fila que contendrá cada uno de los botones.

como establecemos el margen entre botones en la fila (**spacing**)

> Bueno… ya tenemos un botón y una toolbar. En el siguiente post añadiremos unos controles de edición.