In-app advertising how to use Inneractive API in QML application

This code example shows how you can use the standard `QWidget`-based In-app advertising API widget from within a QML app. The approach given can also be used for displaying other `QWidget` based controls from within QML.

Introduction

The Inneractive In-application Advertizing API is the part of Nokia's Monetization Suite. It allows developers to easily display advertising banners within an application, handling everything from user interaction with the banner through to developer payment.

In-application ad servicing solution demo project screen shot

The API exists provides a click-able ad-banner control box that is derived from `QWidget`. This class can be used within Qt C++ applications directly, but is not immediately compatible with QML applications. This article shows how you can re-use this API within your QML application with minimal implementation efforts.

Design approach

The design approach is to create a view container box derived from `QDeclarativeItem` that is available from within QML. The container provides an area where the `InnerActiveAdWidget` can draw itself and also handles re-sizing events from QML document.

The implementation in three steps

1. Create a view box container class to define the rectangular area in QML in which the banner is displayed (this is the minimal implementation of a visual QML element). The `AdBannerWidget` class declaration below shows the important code for making the element interact with QML. Implementation details can be seen in proejct source code

```
    class AdBannerWidget : public QDeclarativeItem
    {
...
    private:
        //these methods below are to handle events from QML side about the box geometry and painting
        void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);
        void geometryChanged(const QRectF & newGeometry, const QRectF & oldGeometry);
...
    private:
        // data
        InnerActiveAdModule* ipAdModule; // reference to ad engine
        InnerActiveAdWidget* ipAdWidget; // reference to ad widget
    };
```

2. Register `AdBannerWidget` type for QML before document loading --- the most convenient place i think is in Qt application setup -- function `main()`

```
qmlRegisterType<AdBannerWidget>("com.izinin.inneractive.ad", 1, 0, "AdBanner");
```


(C) Copyright Nokia 2013. All rights reserved.

3. Use the view box in QML document is shown below

```
    AdBanner{
        id: adbanner
        height: 70
        width: Math.min(screen.width, screen.height)
    }
...
    states: [
        State {
            name: "landscape"; when: screen.width > screen.height
...
            PropertyChanges {
                target: adbanner
                x:optionbar.x
                y:optionbar.y + optionbar.height + screen.margin
            }
        },
...
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.