How to install and use Qt Quick extension plug-in in Symbian

Introduction

This article shows how to create a DLL (Qt Quick plug-in) with Qt SDK/Creator and then use it in a Qt Quick (QML) test application. Existing Qt example code (\declarative\cppextensions\plugins) has been used for this purpose.

How create Qt Quick plug-in

QML is an easier way to create good looking user interfaces. QML applications can use plug-ins where we can implement application logic as a DLL called QML extension plug-in. QDeclarativeExtensionPlugin is a plug-in interface that makes it possible to create QML extensions that can be loaded dynamically into QML applications. These extensions allow custom QML types to be made available to the QML engine.

To create the extension plug-in we need to do the following:

- Subclass QDeclarativeExtensionPlugin
- Implement its registerTypes method
- Export the class by using Q_EXPORT_PLUGIN2() macro

Q_EXPORT_PLUGIN2() macro should be used in the following way:

```
Q_EXPORT_PLUGIN2(qtqmltplugins, QExampleQmlPlugin);
```

Where *qtqmltplugins* is the plug-in name, corresponding to TARGET specified in plug-in .pro file and *QExampleQmlPlugin* is the class that is being exported. There should be exactly one occurrence of this macro in the source code for a Qt plug-in, and it should be used where the implementation is written rather than in a header file.

- A qmldir file, containing meta-data for the new module:

```
#<TypeName> [<InitialVersion>] <File>
Clock 1.0 Clock.qml
#plugin <Name> [<Path>]
plugin qtqmltplugins
```

Plug-in .pro file

```
TEMPLATE = lib
CONFIG += qt plugin
QT += declarative

# Destination directory that is added to the installation path
DESTDIR = com/nokia/TimeExample

# Name of our plug-in
TARGET  = qtqmltplugins
SOURCES += plugin.cpp

# qdeclarativesources holds all the files needed by the plug-in
qdeclarativesources.files += \
    com/nokia/TimeExample/qmldir \
    com/nokia/TimeExample/center.png \
    com/nokia/TimeExample/clock.png \
    com/nokia/TimeExample/Clock.qml \
    com/nokia/TimeExample/hour.png \
    com/nokia/TimeExample/minute.png

qdeclarativesources.path += $$[QT_INSTALL_EXAMPLES]/declarative/plugins/com/nokia/TimeExample

sources.files += plugins.pro plugin.cpp plugins.qml README
sources.path += $$[QT_INSTALL_EXAMPLES]/declarative/plugins
target.path += $$[QT_INSTALL_EXAMPLES]/declarative/plugins/com/nokia/TimeExample

# List of resources that will be installed
INSTALLS += qdeclarativesources target

# On Symbian, define DEPLOYMENT to generate a SIS file with all the necessary files
# The correct location to deploy plug-in files on Symbian is under /resource/qt/imports/
symbian {
    TARGET.EPOCALLOWDLLDATA = 1

    pluginFiles.sources = $${TARGET}.dll $${qdeclarativesources.files}
    pluginFiles.path = $$QT_IMPORTS_BASE_DIR/$$DESTDIR
    DEPLOYMENT += pluginFiles
}
```


(C) Copyright Nokia 2013. All rights reserved.

Creating a Qt Quick application

We can use Qt Creator wizards to create a Qt Quick application (*qtqmltpluginstest*). Create an application by wizard and that will create a main.qml file. Wizards creates relatively big projects but we don't need to worry with those. We edit the main.qml file. Here are the contents of the file that uses Clock element from Qt plugins that we just created (qtqmltplugins)

```
import QtQuick 1.0
import com.nokia.TimeExample 1.0

 Clock
 { // this class is defined in QML (com/nokia/TimeExample/Clock.qml)
     width: 360
     height: 360
     Time { // this class is defined in C++ (plugin.cpp)
         id: time
     }
     hours: time.hour
     minutes: time.minute
     MouseArea {
         anchors.fill: parent
         onClicked: {
             Qt.quit();
         }
     }
 }
```

Testing in device

These application can be tested easily with real devices by using AppTRK with a USB cable. Compile and deploy the *qtqmltplugin* by selecting "Deploy Project" and see that DLL has been really installed in the device. Next, compile and run *qtqmltpluginstest* to see a new QML UI element (Clock, defined in the plug-in) displayed on screen. If we touch anywhere in the UI, the application is closed.

Example plug-in

These examples are tested with Nokia N8 and Qt 4.7

File:Qtqmltplugins.zip
File:Qtqmltpluginstest.zip

 (C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.