Using Publish and Subscribe APIs in Qt

Overview

This code example shows how to publish and subscribe content in Qt. A common use of this API is to communicate between two applications. Sharing data, that are in private folder, between two application is not possible without Allfiles capability in symbian, so publish and subscribe API is useful in that case. This code example shows how to use QtMobility Publish and Subscribe APIs to communicate between two applications.

This snippet does not require any capabilities. Self-signing is possible for testing application.

Prerequisite

- Install QtMobility APIs:Setting up environment for Qt Mobility API

Headers required

```
#include <qvaluespacesubscriber.h>
#include <qvaluespacepublisher.h>
```

.qcrml file

Create a .qcrml file (here myPublisher.qcrml) in your publisher application and deploy it in c:/resource/qt/crml directory. Note that the **uidValue** in .qcrml file must be the same as UID of application that suppose to publish content.

```
<?xml version="1.0" encoding="UTF-8"?>
<repository target="RProperty" version="" uidName="repo" uidValue="0xE0633550">
<key ref="/mypublisher/name" int="0x1"/>
</repository>
```

.pro file

Modify your .pro file of publisher application.

```
CONFIG += mobility
MOBILITY = publishsubscribe

INCLUDEPATH += D:/QtMobility/src/publishsubscribe
DEPENDPATH += D:/QtMobility/src/publishsubscribe
INCLUDEPATH += D:/QtMobility/src/global

symbian {
        myCrml.sources = myPublisher.qcrml
        myCrml.path = c:/resource/qt/crml
        DEPLOYMENT += myCrml
    }
```

Modify your .pro file of subscriber application.

```
INCLUDEPATH += D:/QtMobility/src/publishsubscribe
DEPENDPATH += D:/QtMobility/src/publishsubscribe
INCLUDEPATH += D:/QtMobility/src/global

CONFIG += mobility
MOBILITY = publishsubscribe
```

Header file

Publisher.h

```
#ifndef PUBLISHER_H
#define PUBLISHER_H

#include <QtGui/QMainWindow>

#include <qmobilityglobal.h>
#include <QDialog>

QTM_BEGIN_NAMESPACE
class QValueSpacePublisher;
QTM_END_NAMESPACE
```


(C) Copyright Nokia 2013. All rights reserved.

```
QTM_USE_NAMESPACE

class QLabel;
class QLineEdit;
class QPushButton;

//class Publisher : public QMainWindow
class Publisher : public QWidget
{
    Q_OBJECT

public:
 Publisher(QWidget *parent = 0);
    ~Publisher();

private:
    void initialiseLayout();
 void resizeEvent (QResizeEvent* event);
 void setWidgetGeometry(QSize* screenSize);

private slots:
    void publishValue();

private:
 QLabel        *labelName;
 QLineEdit     *inputName;
 QPushButton   *saveButton;
 QPushButton   *exitButton;
 QValueSpacePublisher *publisher;
};

#endif // PUBLISHER_H
```

Subscriber.h

```
#ifndef SUBSCRIBER_H
#define SUBSCRIBER_H

#include <QWidget>
#include <qmobilityglobal.h>


QTM_BEGIN_NAMESPACE
class QValueSpaceSubscriber;
QTM_END_NAMESPACE

QTM_USE_NAMESPACE

class QLabel;
class QLineEdit;
class QPushButton;

class Subscriber : public QWidget
{
    Q_OBJECT

public:
 Subscriber(QWidget *parent = 0);
    ~Subscriber();


private:
    void initialiseLayout();
    void setSubscriber();
 void resizeEvent (QResizeEvent* event);
 void setWidgetGeometry(QSize* screenSize);

private slots:
    void getChangedValue();

private:
 QLabel        *labelName;
 QLineEdit     *inputName;
 QPushButton   *saveButton;
 QPushButton   *exitButton;
 QValueSpaceSubscriber *subscriber;
};

#endif // SUBSCRIBER_H
```

Source file

Publisher.cpp

Publisher publish value using QValueSpacePublisher variable that can be subscribed by one or more subscriber.

```
#include "Publisher.h"
#include <qvaluespacepublisher.h>
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QResizeEvent>
#include <QVariant>
#include <QMessageBox>

Publisher::Publisher(QWidget *parent)
    : QWidget(parent), publisher(0)
{
 setWindowTitle("Publisher");

 /*initialize GUI components */
 initialiseLayout();
}

void Publisher::initialiseLayout()
{
```


(C) Copyright Nokia 2013. All rights reserved.

```
 labelName = new QLabel(tr("Name"),this);
 labelName->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

 inputName = new QLineEdit(this);
 inputName->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Fixed);
 inputName->setText("Savaj");

 saveButton = new QPushButton(tr("Publish name"),this);
 exitButton = new QPushButton(tr("Exit"),this);
 exitButton->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Fixed);

 setWidgetGeometry(&(size()));

 connect(saveButton, SIGNAL(clicked()), this, SLOT(publishValue()));
 connect(exitButton, SIGNAL(clicked()), this, SLOT(close()));
}

void Publisher::resizeEvent (QResizeEvent* event)
{
    QSize widgetSize = event->size();
    setWidgetGeometry(&widgetSize);
    QWidget::resizeEvent(event);
}

void Publisher::setWidgetGeometry(QSize* screenSize)
{
 resize(*screenSize);
 QRect rect(0,0,screenSize->width(), screenSize->height());

 int width = 110;
 int height = 50;
 int margin = 5;
 labelName->setGeometry(0,0,width,height);
 inputName->setGeometry(width + margin,0,(width * 2),height);
 exitButton->setGeometry(0,height + margin,width,height);
 saveButton->setGeometry(width + margin, height + margin ,(width * 2),height);
}

void Publisher::publishValue()
{
 if (!publisher)
 {
  /* Constructs a QValueSpacePublisher that publishes values under path /mypublisher*/
  publisher = new QValueSpacePublisher("/mypublisher", this);
 }

 /* Sets the value name on the publisher to data. */
 publisher->setValue("name", QVariant(inputName->text().toAscii()));
}

Publisher::~Publisher()
{
}
```

Subscriber.cpp

Subscriber subscribe data, using QValueSpaceSubscriber variable, that is published by publisher.

```
#include "Subscriber.h"
#include <qvaluespacesubscriber.h>

#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QResizeEvent>

Subscriber::Subscriber(QWidget *parent)
    : QWidget(parent), subscriber(0)
{
 setWindowTitle("Subscriber");

 /*initialize GUI components */
 initialiseLayout();
}

void Subscriber::initialiseLayout()
{
 labelName = new QLabel(tr("Name"),this);
 labelName->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);


 inputName = new QLineEdit(this);
 inputName->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Fixed);


 saveButton = new QPushButton(tr("Subscribe name"),this);

 exitButton = new QPushButton(tr("Exit"),this);
 exitButton->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Fixed);


 setWidgetGeometry(&(size()));

 connect(saveButton, SIGNAL(clicked()), this, SLOT(getValue()));
 connect(exitButton, SIGNAL(clicked()), this, SLOT(close()));

 setSubscriber();
}

void Subscriber::resizeEvent (QResizeEvent* event)
{
    QSize widgetSize = event->size();
    setWidgetGeometry(&widgetSize);
    QWidget::resizeEvent(event);
}

void Subscriber::setWidgetGeometry(QSize* screenSize)
{
 resize(*screenSize);
 QRect rect(0,0,screenSize->width(), screenSize->height());

 int width = 110;
 int height = 50;
 int margin = 5;
 labelName->setGeometry(0,0,width,height);
 inputName->setGeometry(width + margin,0,(width * 2),height);
```


(C) Copyright Nokia 2013. All rights reserved.

```
 exitButton->setGeometry(0,height + margin,width,height);
 saveButton->setGeometry(width + margin, height + margin ,(width * 2),height);
}

void Subscriber::setSubscriber()
{
 /* Constructs a QValueSpaceSubscriber with the specified parent that refers to path /mypublisher. */
 if(!subscriber)
  subscriber = new QValueSpaceSubscriber("/mypublisher",this);

 connect(subscriber, SIGNAL(contentsChanged()), this, SLOT(getChangedValue()));
}

void Subscriber::getChangedValue()
{

 /* Returns a list of sub-paths under the current path */
    QStringList subPaths = subscriber->subPaths();

    for (int i = 0; i < subPaths.count(); ++i)
    {
  /*Returns the value of the subPath under this subscriber path */
        QVariant variant = subscriber->value(subPaths.at(i));
        inputName->setText(variant.toString());
    }
}

Subscriber::~Subscriber()
{
}
```

How to test application

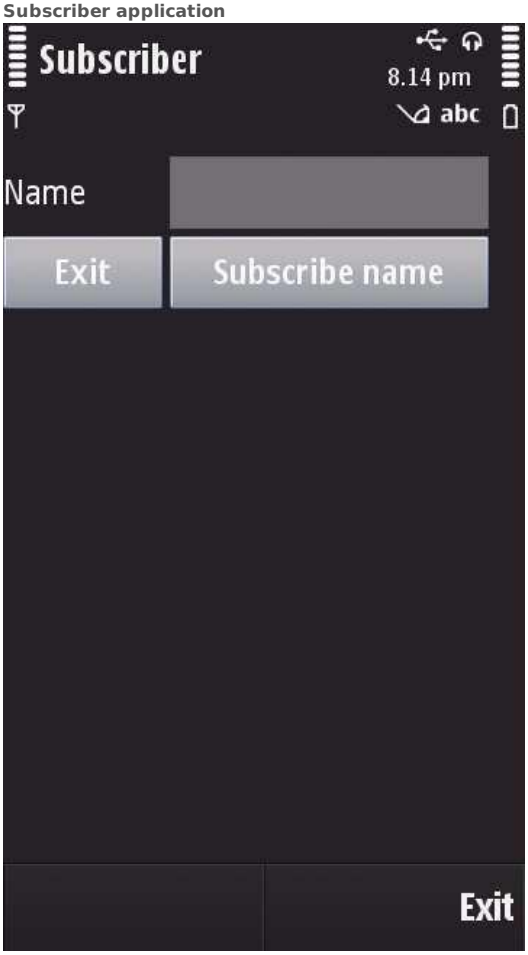
To test this code example, install both application, Publisher and Subscriber, on device. Type anything in editor of Publisher and push "publish name" button which publish data. After that switch to Subscriber, published value will be reflected there.

Post-conditions

The code snippet is expected to publish value using publisher application and subscribe value using subscriber application.

Publisher application

Subscriber application



Download Code Example

- Working Code Example is available to download from here.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.