Статическая сборка

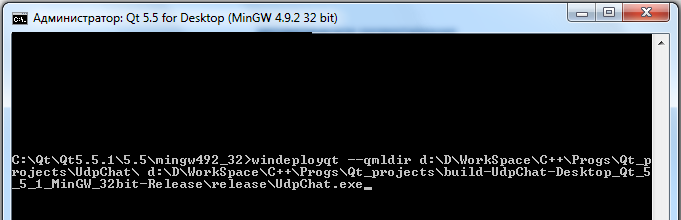
<https://habrahabr.ru/sandbox/95171/>

<https://atas2000.blogspot.com/2017/02/qt-58-mingw-windows.html>

автоматическое развертиівание

<https://bravikov.wordpress.com/tag/windeployqt/>

<https://evileg.com/ru/post/163/>



QML Signals to C++ Slots

//QML

import QtQuick 2.3

import QtQuick.Controls 1.2

ApplicationWindow {

visible: true

width: 640

height: 480

title: *qsTr*("Hello World")

Column{

Text{

text: *qsTr*("Yello World")

}

Button{

// anchors.verticalCenter: parent.verticalCenter

height: 30

text: "Chage Text"

onClicked: *messageClass*.doMessageChange();

}

}

}

//////////////////////////////////////////////////////////

/// message.h

#ifndef MESSAGE\_H

#define MESSAGE\_H

#include <QObject>

class Message : public QObject

{

Q\_OBJECT

public:

explicit Message(QObject \*parent = 0);

signals:

public slots:

void doMessageChange();

};

#endif // MESSAGE\_H

////////////////////////////////////////////

// message.cpp

#include "message.h"

#include <iostream>

using std::cout;

using std::endl;

Message::Message(QObject \*parent) : QObject(parent)

{

}

void Message::doMessageChange()

{

cout<< "We are in C++"<<endl;

}

//////////////////////////////////////////////////////

// main.cpp

#include <QApplication>

#include <QQmlApplicationEngine>

#include <QQmlContext>

#include "message.h"

int main(int argc, char \*argv[])

{

QApplication app(argc, argv);

QQmlApplicationEngine engine;

Message my\_msg;

auto root\_context = engine.rootContext();

root\_context->setContextProperty("messageClass", &my\_msg);

engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

return app.exec();

}

/////////////////////////////////////////////////////////////

QML Slots to C++ Signals

//QML

import QtQuick 2.3

import QtQuick.Controls 1.2

ApplicationWindow {

visible: true

width: 640

height: 480

title: *qsTr*("Hello World")

//sdvkdjkv

Connections{

target: *messageClass*

onMessageChanged: *textId*.text = value;

}

Column{

Text{

id: *textId*

text: *qsTr*("Yello World")

}

Button{

// anchors.verticalCenter: parent.verticalCenter

height: 30

text: "Chage Text"

onClicked: *messageClass*.doMessageChange();

}

}

}

//////////////////////////////////////////////////////////

/// message.h

#ifndef MESSAGE\_H

#define MESSAGE\_H

#include <QObject>

#include <QString>

class Message : public QObject

{

Q\_OBJECT

public:

explicit Message(QObject \*parent = 0);

signals:

void messageChanged(QString value);

public slots:

void doMessageChange();

private:

int m\_counter;

QString m\_message;

};

#endif // MESSAGE\_H

////////////////////////////////////////////

// message.cpp

#include "message.h"

#include <iostream>

Message::Message(QObject \*parent) : QObject(parent),

m\_counter(0),m\_message("Hello world! %1")

{

}

void Message::doMessageChange()

{

++m\_counter;

emit messageChanged(m\_message.arg(m\_counter));

}

//////////////////////////////////////////////////////

// main.cpp

#include <QApplication>

#include <QQmlApplicationEngine>

#include <QQmlContext>

#include "message.h"

int main(int argc, char \*argv[])

{

QApplication app(argc, argv);

QQmlApplicationEngine engine;

Message my\_msg;

auto root\_context = engine.rootContext();

root\_context->setContextProperty("messageClass", &my\_msg);

engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

return app.exec();

}

/////////////////////////////////////////////////////////////

QPROPERTY

Це саме можна зробити інакше

//QML

import QtQuick 2.3

import QtQuick.Controls 1.2

ApplicationWindow {

visible: true

width: 640

height: 480

title: *qsTr*("Hello World")

Column{

Text{

id: *textId*

text: *messageClass*.message; //!!!

}

Button{

// anchors.verticalCenter: parent.verticalCenter

height: 30

text: "Chage Text"

onClicked: *messageClass*.doMessageChange();

}

}

}//////////////////////////////////////////////////////////

макрос Q\_PROPERTY сделает доступным из QML какое-нибудь свойство

/// message.h

#ifndef MESSAGE\_H

#define MESSAGE\_H

#include <QObject>

#include <QString>

class Message : public QObject

{

Q\_OBJECT

Q\_PROPERTY(QString message READ message WRITE setMessage NOTIFY messageChanged)

public:

explicit Message(QObject \*parent = 0);

QString message()const {return m\_message;}

void setMessage(QString);

signals:

void messageChanged();

public slots:

void doMessageChange();

private:

int m\_counter;

QString m\_message;

};

#endif // MESSAGE\_H

////////////////////////////////////////////

Здесь QString message собственно само наше свойство, READ message – метод для чтения, WRITE setMessage – метод для записи. Если мы собираемся менять наше свойство из C++ то нам необходимо уведомлять об этом интерфейс на QML с помощью сигнала NOTIFY messageChanged.

/////////////////////////////\\\\\\

// message.cpp

#include "message.h"

#include <iostream>

Message::Message(QObject \*parent) : QObject(parent),

m\_counter(0),m\_message("First message")

{

}

void Message::doMessageChange()

{

setMessage("Hello %1");

}

void Message::setMessage(QString value)

{

m\_message = value.arg(++m\_counter);

emit messageChanged();

}

//////////////////////////////////////////////////////

// main.cpp

#include <QApplication>

#include <QQmlApplicationEngine>

#include <QQmlContext>

#include "message.h"

int main(int argc, char \*argv[])

{

QApplication app(argc, argv);

QQmlApplicationEngine engine;

Message my\_msg;

auto root\_context = engine.rootContext();

root\_context->setContextProperty("messageClass", &my\_msg);

engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

return app.exec();

}

/////////////////////////////////////////////////////////////

Q\_INVOKEBLE

Для того что бы иметь возможность вызывать из QML C++ методы их необходимо определять как слоты либо при помощи макроса Q\_INVOKABLE.

//QML

import QtQuick 2.3

import QtQuick.Controls 1.2

ApplicationWindow {

visible: true

width: 640

height: 480

title: *qsTr*("Hello World")

Column{

Text{

id: *textId*

text: *messageClass*.message;

}

Button{

// anchors.verticalCenter: parent.verticalCenter

height: 30

text: "Chage Text"

onClicked: *messageClass*.doMessageChange();

}

Button{

// change buttun hight

property int res : 0

height: 30 + *res*

text: "Talk to c++"

onClicked: {

*res* = *messageClass*.sendMesFromCpp(*text*);

}

}

}

}

//////////////////////////////////////////////////////////

/// message.h

#ifndef MESSAGE\_H

#define MESSAGE\_H

#include <QObject>

#include <QString>

class Message : public QObject

{

Q\_OBJECT

Q\_PROPERTY(QString message READ message WRITE setMessage NOTIFY messageChanged)

public:

explicit Message(QObject \*parent = 0);

QString message()const {return m\_message;}

void setMessage(QString);

Q\_INVOKABLE int sendMesFromCpp(const QString value);

signals:

void messageChanged();

public slots:

void doMessageChange();

private:

int m\_counter;

QString m\_message;

};

#endif // MESSAGE\_H

////////////////////////////////////////////

// message.cpp

#include "message.h"

#include <iostream>

Message::Message(QObject \*parent) : QObject(parent),

m\_counter(0),m\_message("First message")

{

}

void Message::doMessageChange()

{

setMessage("Hello %1");

}

void Message::setMessage(QString value)

{

m\_message = value.arg(++m\_counter);

emit messageChanged();

}

int Message::sendMesFromCpp(const QString value)

{

m\_message = value;

emit messageChanged();

return m\_counter;

}

//////////////////////////////////////////////////////

// main.cpp

#include <QApplication>

#include <QQmlApplicationEngine>

#include <QQmlContext>

#include "message.h"

int main(int argc, char \*argv[])

{

QApplication app(argc, argv);

QQmlApplicationEngine engine;

Message my\_msg;

auto root\_context = engine.rootContext();

root\_context->setContextProperty("messageClass", &my\_msg);

engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

return app.exec();

}

/////////////////////////////////////////////////////////////

есть интерфейс приложения на QML и C++ класс с логикой работы.

1. первое что нам нужно сделать для того что бы клас был доступен из QML – это унаследовать его от QObject
2. сделать доступным из QML какое-нибудь свойство, для этого существует макрос Q\_PROPERTY.

someProperty собственно само наше свойство, getSomeProperty – метод для чтения, setSomeProperty – метод для записи.

Если мы собираемся менять наше свойство из C++ то нам необходимо уведомлять об этом интерфейс на QML с помощью сигнала somePropertyChanged.

1. зарегистрировать наш класс в QML, для этого нам нужно вызвать в конструкторе QmlApplicationViewer, который создается QtCreator автоматически при создании нового QtQuick проекта, добавить вызов шаблонной функции qmlRegisterTypes.

qmlRegisterType<TestClass>("ModuleName", 1, 0, "TypeName");

Здесь TestClass – наш класс, ModuleName – имя импортируемого QML модуля, TypeName – имя типа объектов в QML. Теперь в QML файле импортируем наш класс, и создаем экземпляр.

import ModuleName 1.0

1. Для того что бы иметь возможность вызывать из QML C++ методы их необходимо определять как слоты либо при помощи макроса Q\_INVOKABLE.

Теперь должны зарегистрировать наш класс в QML, для этого нам нужно вызвать в конструкторе QmlApplicationViewer, который создается QtCreator автоматически при создании нового QtQuick проекта, добавить вызов шаблонной функции qmlRegisterTypes.