Ref:

http://imaginativethinking.ca/qtest-101-writing-unittests-qt-application/

# Creating Qt HelloWorld Application

Using Qt-Creator, **New project > Application > Qt Widgets Application**.

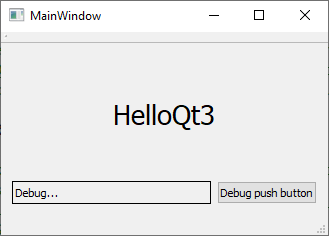
We create a new folder with our chosen project name. In this example we call it “HelloQt3”:

C:\Users\diana\OneDrive\Dokumente\QtProjects\HelloQt3

We use **Desktop Qt 5.13.0 MinGW 64-bit** as compiler.

We add a push button and a label for debugging purposes. Then we make a connection in **MainWindow** so that as we click on push button, a text is printed in label.

The application now looks like this:



# Project Hierarchy

In Visual Studio we have **Solution,** which can consist of several **Project**s. In Qt we have **Project** and **Subprojects,** as we see in below table:

|  |  |
| --- | --- |
| **Qt** | **Visual Studio** |
| Project | Solution |
| Subproject | Project |

In such structure, we will have a **Project**, with at least two **Subprojects**:

* Application
* Unit-test

In this section we send the application to a **Subproject**.

We create the empty file “HelloQt3Solution.pro”:

C:\Users\diana\OneDrive\Dokumente\QtProjects\HelloQt3\HelloQt3Solution.pro

And add below two lines to it:

TEMPLATE = subdirs

SUBDIRS += HelloQt3

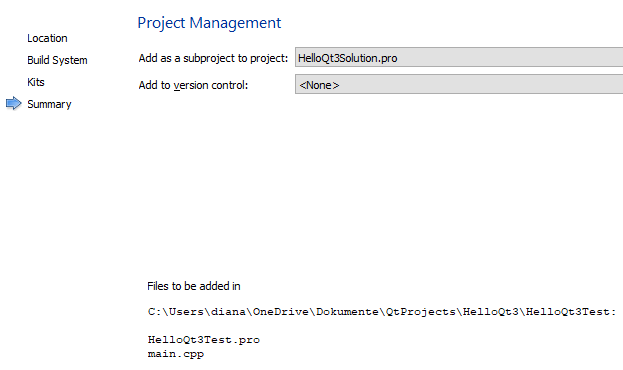
We save and close the file **HelloQt3Solution.pro**. Then we open the file with Qt Creator. We choose the same compiler as above (MinGW 64-bit) and run the project. We must be able to see the same window.

# Adding Test Subproject to Project

We right-click on **HelloQt3Solution** and in context menu we choose **New subproject…**. In coming wizard we need to choose:

* make system: qmake
* Kits: MinGW 64-bit

In the last step, we need to take care that we are adding the subproject to **HelloQt3Solution.pro**:



This subproject is going to be our test subproject. In order to give this ability to subproject, we modify the file **HelloQt3Test.pro** and add below lines to it:

QT += core testlib

QT -= gui

CONFIG += c++11 console

CONFIG -= app\_bundle

CONFIG += testcase

This subproject has just a main.cpp file. We need to add a test class to it. We right click on the subproject and add class **TestDebugWidgets** to it and inherit it from **QObject**.

Now we open main.cpp file and delete everything in it and add below lines to it:

#include <QCoreApplication>

#include <QTest>

#include "TestDebugWidgets.h"

QTEST\_MAIN(TestDebugWidgets)

So far we have a project with two subprojects. We can build both from Qt creator menu: **Build > Build all**. We can also build every single subproject by right clicking on subproject and **Build**.

After we build the test subproject, we run it by right clicking on subproject and **Run**. In application output we see below lines:

\*\*\*\*\*\*\*\*\* Start testing of TestDebugWidgets \*\*\*\*\*\*\*\*\*

Config: Using QtTest library 5.13.0, Qt 5.13.0 (x86\_64-little\_endian-llp64 shared (dynamic) debug build; by GCC 7.3.0)

PASS : TestDebugWidgets::initTestCase()

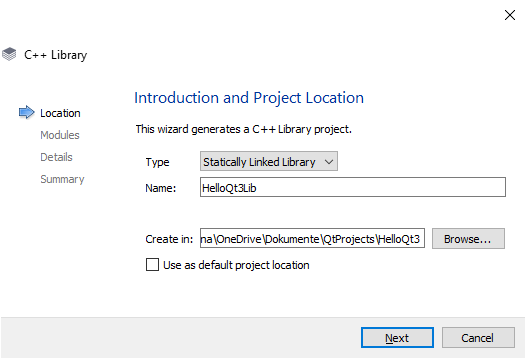
PASS : TestDebugWidgets::cleanupTestCase()

Totals: 2 passed, 0 failed, 0 skipped, 0 blacklisted, 3ms

\*\*\*\*\*\*\*\*\* Finished testing of TestDebugWidgets \*\*\*\*\*\*\*\*\*

# Adding Starically Linked Library

We right click on **HelloQt3Solution > Add Subproject… > Libraries > C++ Library**. And we call the library “HelloQt3Lib”.



Notice to choose **Statistically Linked Library** for **Type**. Between the modules, we just need QtCore.

# Moving Files to Library

We move all .h and .cpp files (except main.cpp) from:

C:\Users\diana\OneDrive\Dokumente\QtProjects\HelloQt3\HelloQt3

to:

C:\Users\diana\OneDrive\Dokumente\QtProjects\HelloQt3\HelloQt3Lib

Since the files are moved from **HelloQt3** library to **HelloQt3Lib** library, their .pro files must be edited. In **HelloQt3.pro** file we remove the file names that we have moved. Then in **HelloQt3Lib.pro** file we add highlighted parts:

SOURCES += \

Configuration.cpp \

MainWindow.cpp \

HelloQt3Lib.cpp

HEADERS += \

Configuration.h \

MainWindow.h \

HelloQt3Lib.h

FORMS += \

MainWindow.ui

We also need to add some to **HelloQt3Lib.pro** file. As we try to build the lib project, we may realize that some components are missing in **HelloQt3Lib.pro**. I needed to add below lines to **HelloQt3Lib.pro**:

QT += core gui

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

We add below lines to **HelloQt3Test.pro** and **HelloQt3.pro** in order to enable them to use library:

# Adds the HelloQt3Lib project path to the header file include lookup path

INCLUDEPATH += $$PWD/../HelloQt3Lib

# Adds the HelloQt3Lib.lib to the linker

win32:CONFIG(release, debug|release): LIBS += -L$$OUT\_PWD/../HelloQt3Lib/release/ -lHelloQt3Lib

else:win32:CONFIG(debug, debug|release): LIBS += -L$$OUT\_PWD/../HelloQt3Lib/debug/ -lHelloQt3Lib

In the next step, we want both other subprojects to be able to use this library. For that, we add below lines to **HelloQt3Solution.pro**:

HelloQt3.depends = HelloQt3Lib

HelloQt3Test.depends = HelloQt3Lib

At the end, we must be able to build all subprojects and run window and test separately.

# Testing Debug Widgets

We add a slot to class **TestDebugWidgets**. The header file will be like:

#ifndef TESTDEBUGWIDGETS\_H

#define TESTDEBUGWIDGETS\_H

#include <QObject>

class MainWindow;

class TestDebugWidgets : public QObject

{

Q\_OBJECT

public:

TestDebugWidgets(QObject\* parent = nullptr);

private slots:

void slotTestPushButton();

private:

MainWindow\* m\_mainWindow;

};

#endif // TESTDEBUGWIDGETS\_H

And the source file:

#include "TestDebugWidgets.h"

#include "MainWindow.h"

#include "Configuration.h"

#include <QPushButton>

#include <QLabel>

#include <QTest>

TestDebugWidgets::TestDebugWidgets(QObject\* parent) :

QObject(parent)

{

m\_mainWindow = new MainWindow();

m\_mainWindow->show();

}

void TestDebugWidgets::slotTestPushButton()

{

QPushButton\* pushButton = m\_mainWindow->findChild<QPushButton\*>("pushButtonDebug");

QLabel\* label = m\_mainWindow->findChild<QLabel\*>("labelDebug");

if (pushButton)

{

// Wait one second (qWait: events are processed. qSleep: events are not processed.)

QTest::qWait(1000);

QTest::mouseClick(pushButton,Qt::MouseButton::LeftButton);

QTest::qWait(1000);

}

else

{

// Force the test to fail

QFAIL("Invalid push button");

}

if (!label)

{

// Force the test to fail

QFAIL("Invalid label widget");

}

Configuration config;

QCOMPARE(label->text(), config.stringButtonClicked);

}