QT INSTALLATION FOR VISUAL STUDIO

FOR PROFESSIONAL & EXPRESS EDITIONS*

The steps for a successful integration of Qt in Visual Studio IDE are as follows:

PRE-REQUISITES:

- Make sure the platform SDK is installed if using Visual Studio Express.
- Visual Studio Express and the platform SDK can be downloaded freely from Microsoft downloads.

STEP 1: DOWNLOAD LIBRARIES

- Download the Qt library from the site: http://www.qtsoftware.com/downloads
- Select "LGPL/Free Downloads"
- Download the file "Download Qt SDK for windows" Note: They mention its based on MinGW and does not support VS compiler. Just follow the steps below and Qt can be made compatible with VS.
- Install the SDK completely.

STEP 2: SETTING ENVIRONMENT VARIABLES

- Add this 2 paths to the Environment Variables: "C:\Qt\2009.01\bin" and "C:\Qt\2009.01\qt\bin". Note: This path will depend where you install the Qt.
- To add to the path: Control Panel > System > Advanced > Environment Variables
- Select "PATH" under system variables. Click "EDIT" and Add this to the end of the line: C:\Qt\2009.01\bin; C:\Qt\2009.01\qt\bin (Remember to separate two paths with a ";")

STEP 3: CONFIGURE QT FOR YOUR MACHINE

- Run the Visual Studio Command prompt. Start > Program Files > Visual Studio > Visual Studio Tools > Visual Studio Command Prompt
- Go to the path of your installed Qt e.g. cd $C:\Qt\2009.01\qt$
- At the prompt type the command *configure*. E.g. *C*:*Qt*\2009.01\qt> configure
- This will configure Qt for your machine. In case you have multiple compilers installed, and want to build the Qt library using a specific compiler, you must specify a qmake specification. This is done by pasing -platform <spec> to configure. E.g. configure -platform win32-msvc

STEP 4: BUILDING QT FOR VISUAL STUDIO

- Run *nmake* at the command prompt
- This takes lots of time. Will build all the demos, examples and libraries.

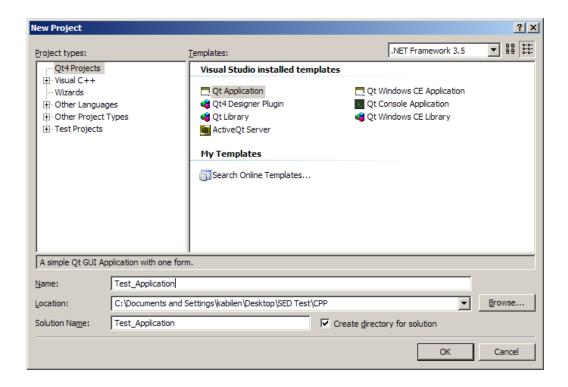
That's it QT is now installed. You can follow the tutorials to use Qt.

OPTIONAL STEP: INSTALLING A VISUAL STUDIO ADD-IN

- * For Express Editions (since this plugin doesn't get installed on Express edition), follow the instructions given at a later section. (Non-plugin Step)
 - This add-in facilitates the creation of a Qt project in Visual Studio.
 - Download the patch at: ftp://ftp.qtsoftware.com/vsaddin/
 - Download the latest one.
 - Install (make sure Visual Studio is closed before installing).
 - Once install is done. Open Visual Studio. Go to the "QT" menu as follows: Qt > Qt
 Options
 - Click on "ADD". Put a version number there and the path where Qt is. E.g.
 C:\Qt\2009.01\qt
 - Click ok.
 - Done. Now Qt template is integrated inside VS.

CREATING A PROJECT IN VISUAL STUDIO

- Open Visual Studio
- Create new project. Select the following template:



• Select "Qt Application", give a name to the project. Press Ok.



- Select Next.
- By default the Core and GUI library is included. In case more libraries are required, please select them accordingly.

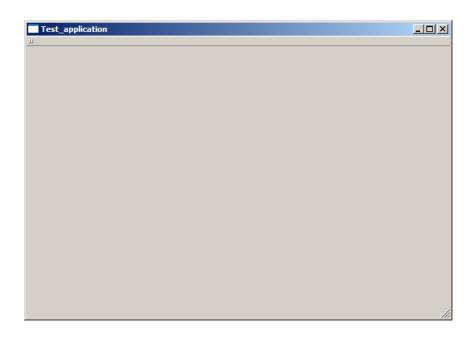


- Finish.
- This will create a plain project and main.cpp containing the following

```
#include <QtGui/QApplication>
#include "test_application.h"

int main(int argc, char *argv[])
{
     QApplication a(argc, argv);
     Test_application w;
     w.show();
     return a.exec();
}
```

- Press F7 to build the solution.
- And F5 to run it.
- You should get something like this running:

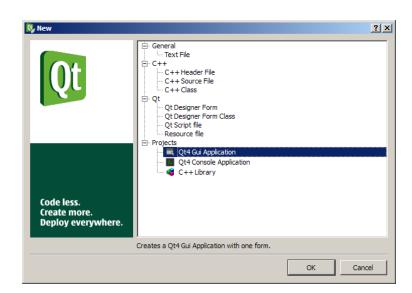


FOR VISUAL STUDIO EXPRESS USERS

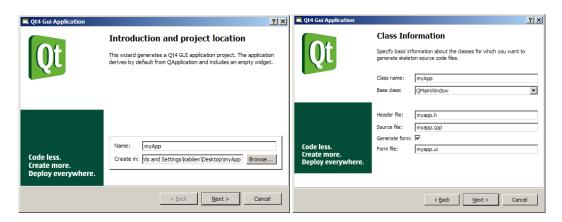
* Non-Plugin Step: Configuring Visual Studio Environment

If not using the *Visual Studio Add-In plugin* (for Express Editions Users), then you need to follow the steps here to create a project in Visual Studio.

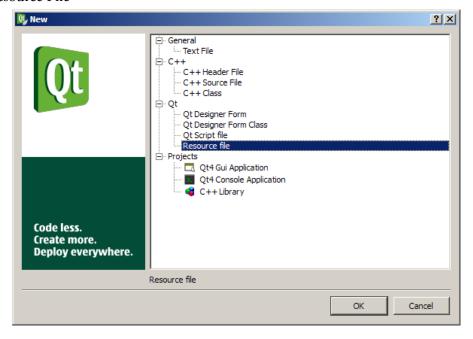
- Follow step 1, 2, 3 and 4 as given above to build Qt.
- Now, to ease use of Qt in VS, run QtCreator. Create a new project there. File->New
- Select Either GUI or Console application. Here I am creating a GUI one.



• Give the project a name and save it in an empty folder.

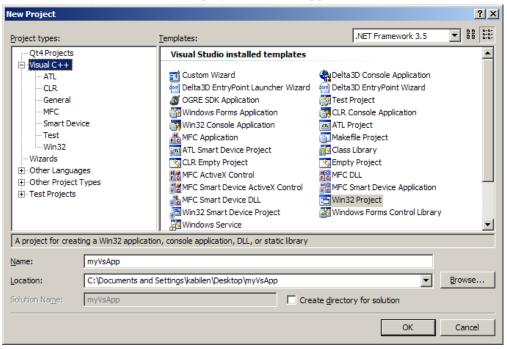


- Once created, it will create the necessary files and a UI file. Here you can play with your UI to create the desired GUI.
- We shall add a resource file to the project. *File->New*
- Select Resource File

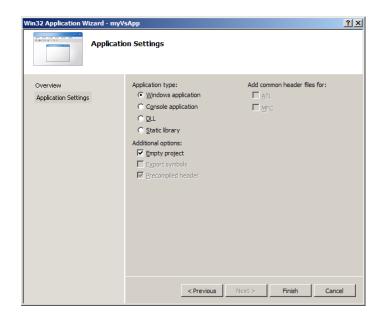


- Rename it. Here since I used myApp as the name all over, my resource file is named as myApp.grc
- Now build the project in QtCreator itself. *Build -> Build All*
- This step create the Q_Object generated files which is required when we going to use in VS.

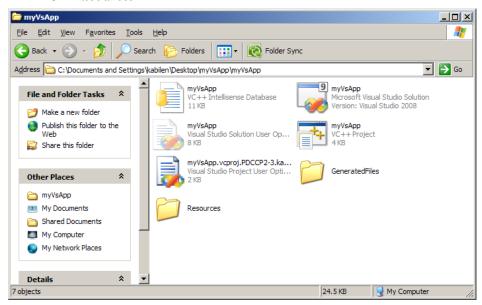
- Now we go to Visual Studio, create a new project.
- Create a Win 32 Project. I named the project as myVsApp



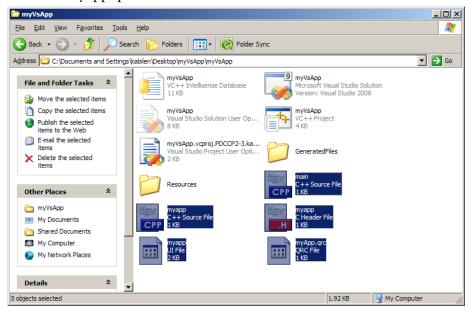
Remember to create an EMPTY PROJECT



- Go to the folder where the project is created and add the following directories to it:
 - GeneratedFiles
 - Resources



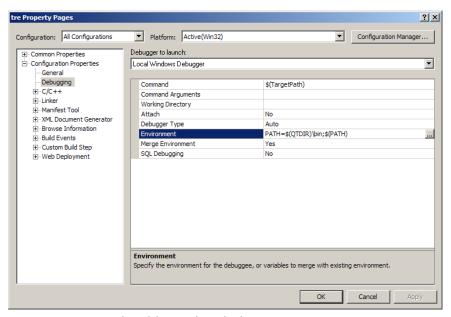
- Now copy the following files from the QtCreator Project folder to the myVsApp application folder:
 - o main.cpp
 - o myapp.cpp
 - o myapp.h
 - o myapp.ui
 - o myapp.qrc



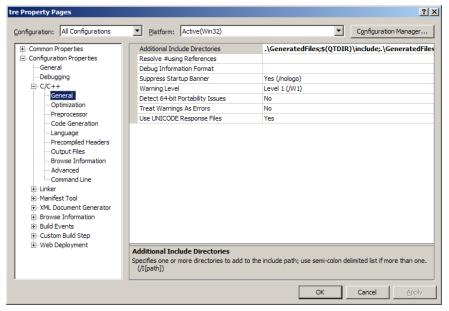
- Now copy the following files from the debug folder of the QtCreator Project folder to GeneratedFiles folder of the VS application folder
 - o moc_myapp.cpp
 - o qrc_myApp.cpp
- Now copy the file ui_myapp.h from the QtCreator Project folder to the GeneratedFiles folder
- Ok now we go back to our empty project in VS and import those files to our project.
 Create the necessary filters in the solution explorer window and add existing items.
 After adding the files you should get something like this:



- Now we need to link the libraries to the project. Follow the steps below to link the necessary files.
- Under Project->Properties, use Configuration: All Configurations.
- Under Configuration Properties -> Debugging -> Environment:
 - o PATH=\$(QTDIR)\bin;\$(PATH)

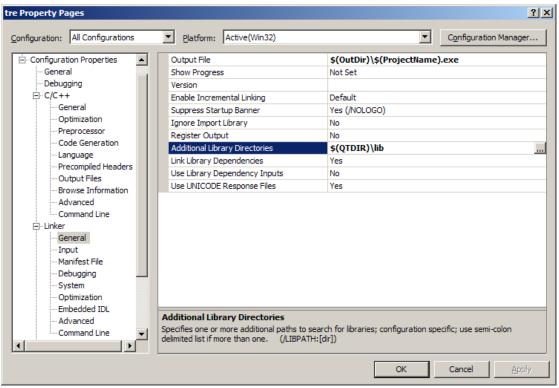


- Now under C/C++ -> General, Additional Include Directories:
 - o .\GeneratedFiles;\$(QTDIR)\include;.\GeneratedFiles\\$(ConfigurationName);;\$(QTDIR)\include\QtCore;\$(QTDIR)\include\QtGui



- Under C/C++ -> Preprocessor, Preprocessor Definitions, replace all with:
 - o UNICODE, WIN32, QT_THREAD_SUPPORT, QT_CORE_LIB, QT_GUI_LIB

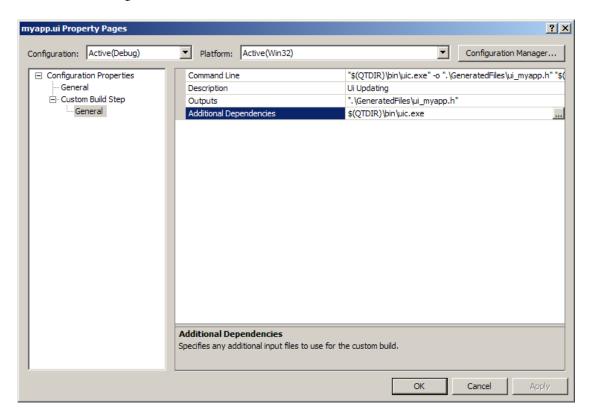
- Now, under Linker -> General, Additional Library Directories:
 - o \$(QTDIR)\lib



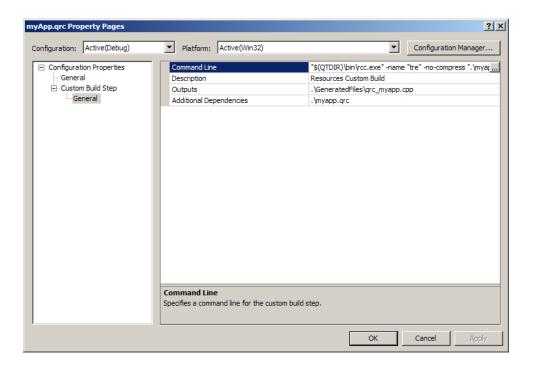
- Now under Linker->Input, Additional Dependencies:
 - o qtmaind.lib QtCored4.lib QtGuid4.lib
- Click OK to save these settings.
- That's it. Compile. Run. ② ... When we generated the project from QCreator, it had created the sample code in it. So to test here, just compile and run.
- Now, if u double click on myApp.ui from the solution explorer, it will send you to Qt
 designer. U can change design there, but since we not using the plugin, it won't
 automatically change the files here.
- So the best way is to open the PRO file with QCreator, and edit the UI file there and save and build. It will create the ui_myapp.h file also.
- Then replace the existing ui file, ui_myapp.h, and qrc_myapp.cpp in the project by the newly generated ones to reflect the update.

Note: An automated way to do the above is to create custom build events. Follow the following steps to do this:

- In Solutions explorer, right click on the myapp.ui and select properties
- Under Configuration Properties -> Custom Build Step -> General, Command Line:
 - o "\$(QTDIR)\bin\uic.exe" -o ".\GeneratedFiles\ui_myapp.h" "\$(InputPath)"
- Outputs:
 - o ".\GeneratedFiles\ui_myapp.h"
- Additional Dependencies:
 - o \$(QTDIR)\bin\uic.exe



- Similarly, for resources, Right Click on myapp.qrc and select properties
- Under Configuration Properties -> Custom Build Step -> General, Command Line:
 - o "\$(QTDIR)\bin\rcc.exe" -name "tre" -no-compress ".\myapp.qrc" -o .\GeneratedFiles\qrc_myapp.cpp
- Output:
 - o .\GeneratedFiles\qrc_myapp.cpp
- Additional Dependencies:
 - o .\myapp.qrc



 Now, whenever a change is done in the UI or resources, it automatically reflects on building the project.

For any queries or questions email me: kabilen@delinkx.com