

# Install Qt on a Mac with Xcode or QtCreator

par Michel LIBERADO

Date de publication : 07 february 2009

Dernière mise à jour : 13 february 2009

These few pages show you how to easily install Qt 4.4.3 on a Mac and use it either with Xcode or Qt Creator.



*Version en français*

(Version en français disponible)

I - Installing Xcode and compilation tools.....	3
I-A - Information.....	3
I-B - Installing Xcode.....	3
II - Installing Qt 4.4.3.....	6
II-A - Information.....	6
II-B - Installing Qt 4.4.3 Mac.....	6
III - Your first compilation from the terminal.....	11
III-A - Creation of a source file.....	11
III-B - Project file creation and compilation.....	12
III-C - Explanations of commands.....	12
IV - Your first compilation with Xcode.....	13
V - Your first compilation using Qt Creator.....	14
VI - sum up.....	15

## I - Installing Xcode and compilation tools

### I-A - Information

In order to use Qt and compile your sources, you must install a compiler first. If you don't, you can still use Qt, but obviously the *make* command won't be available as an internal command. Qt's documentation deals with that :

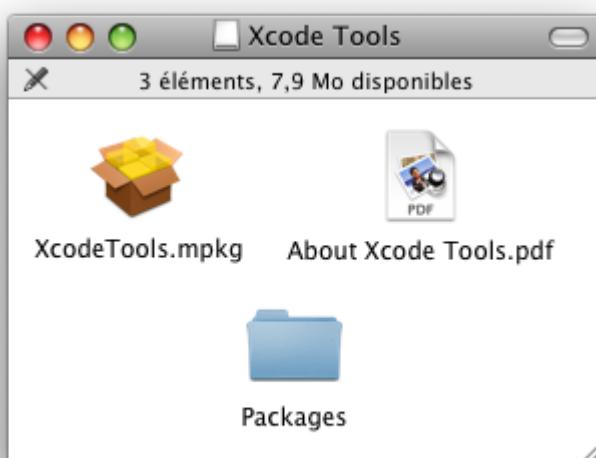
 *On Mac OS X, the binary package requires Mac OS X 10.4.x (Tiger) or later and GCC 4.0.1 to develop applications. Its applications will run on Mac OS X 10.3.9 and above.*

Installing Xcode can be done from the Mac OS X DVD that comes with your Mac. It's also available as a free download from Apple's website. The version used for this tutorial is 3.2.1 which can be found at this url (about 1 Go) : [Download last Xcode version on Apple's website](#)

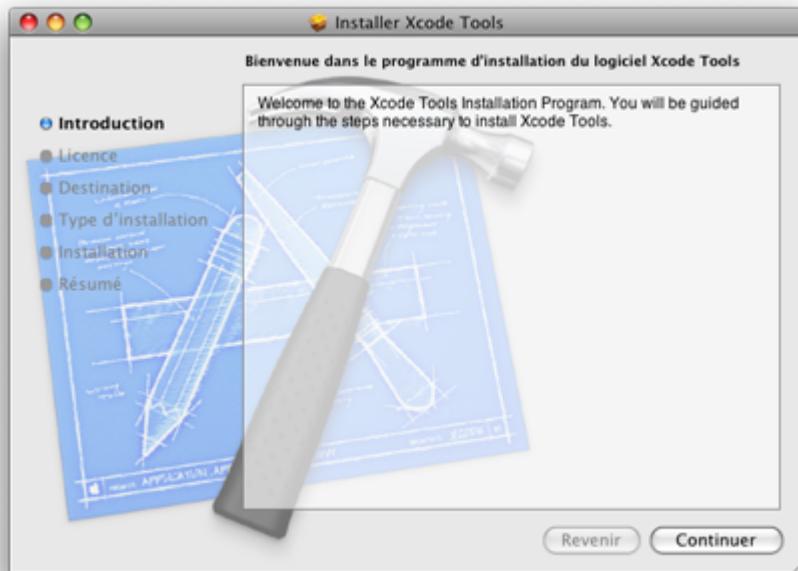
### I-B - Installing Xcode

Mount the given disk image (name : **Xcode312\_2621\_developerdvd.dmg**). Double click on *XcodeTool.mpkg* and follow the instructions. You should then have almost the same screenshots :

 *The installation should last about 5 minutes.*



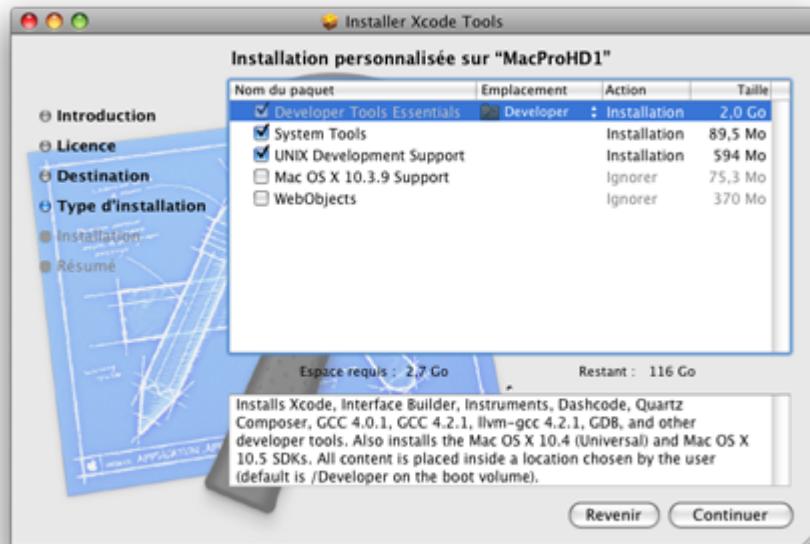
*Disk image content*



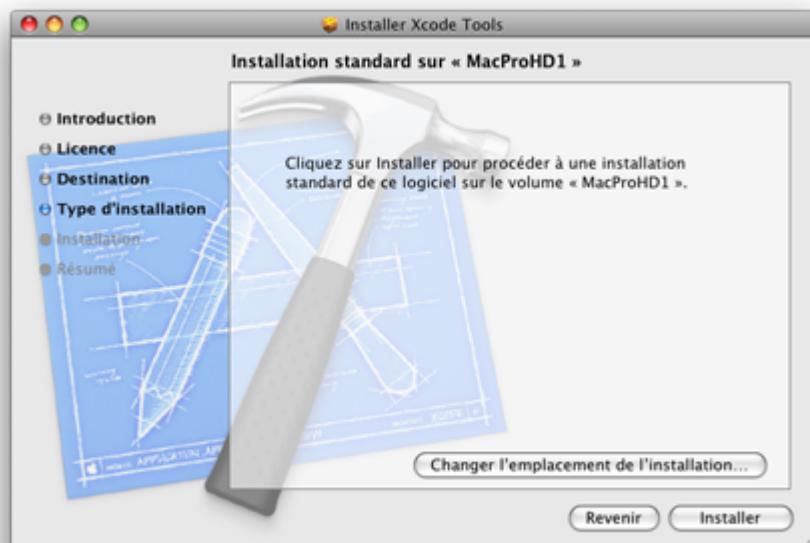
### Xcode installation : introduction



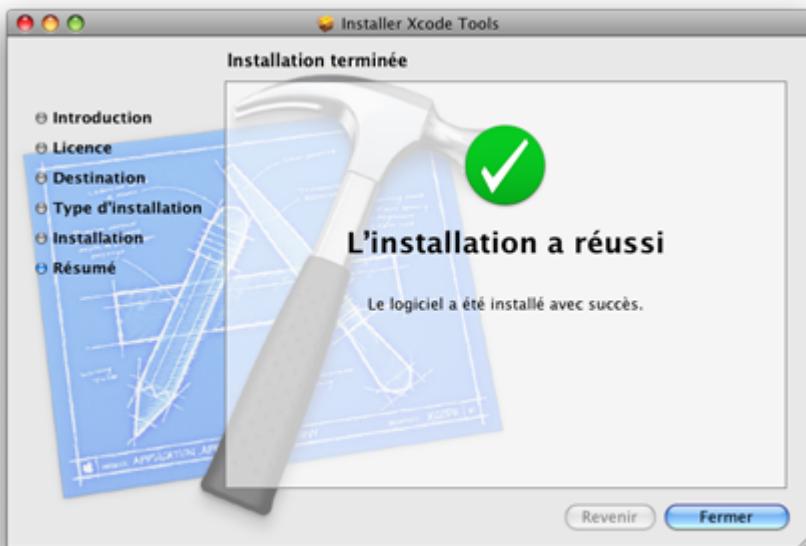
### Xcode installation : license



Xcode installation : Installation type



Xcode installation : writing files



### Xcode installation : end

The default directory where files are written is **/Developer**. No need to restart, you can go to the next step : Installing Qt 4.4.3

## II - Installing Qt 4.4.3

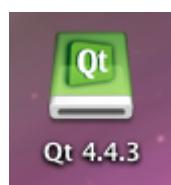
### II-A - Information

As for the PC version, you can download the sources and compile yourself, instructions are given by Qt. Here we will focus to the binaries given by Qt's website. The version used for this tutorial is 4.4.3 which can be found at this url (about 120 Mo) : [Download Qt 4.4.3 binaries \(.dmg\) on Qt's website](#)

 Download the disk image in the "Additional downloads" section.

 You should also download the debug libraries if you plan to debug (except if you are a C++ Master :D)

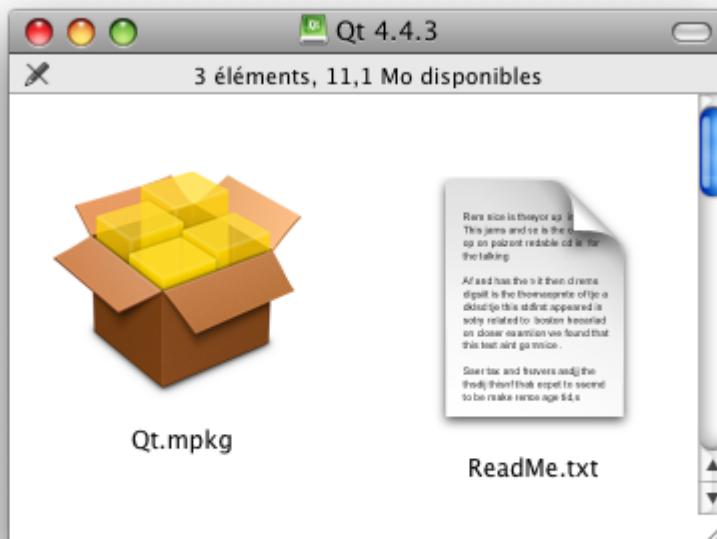
### II-B - Installing Qt 4.4.3 Mac



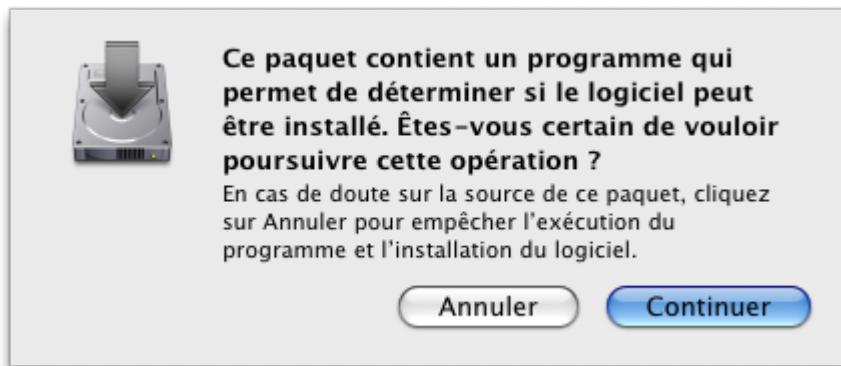
Qt 4.4.3 disk image mounted on the desktop

Double click on **Qt.mpkg** and follow the instructions. You should then have almost the same screenshots :

 The installation should take about 5 minutes.



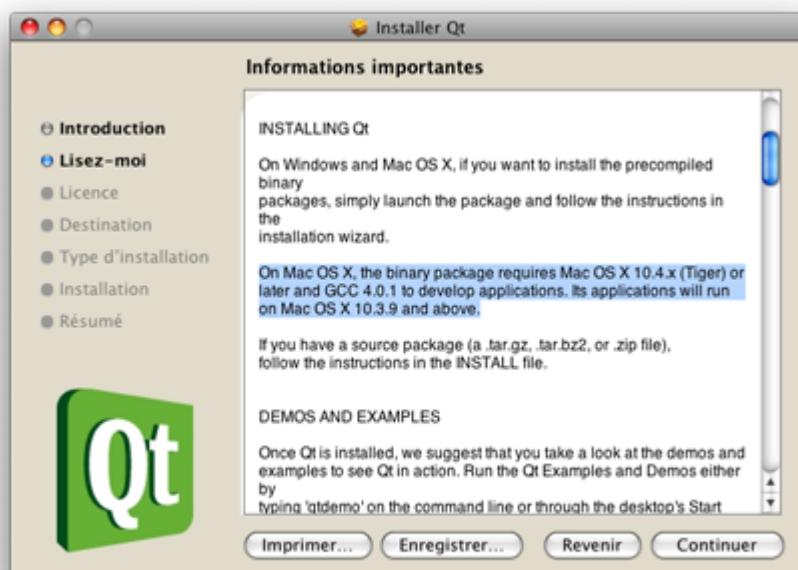
Qt 4.4.3 Mac disk image content : click on Qt.mpkg



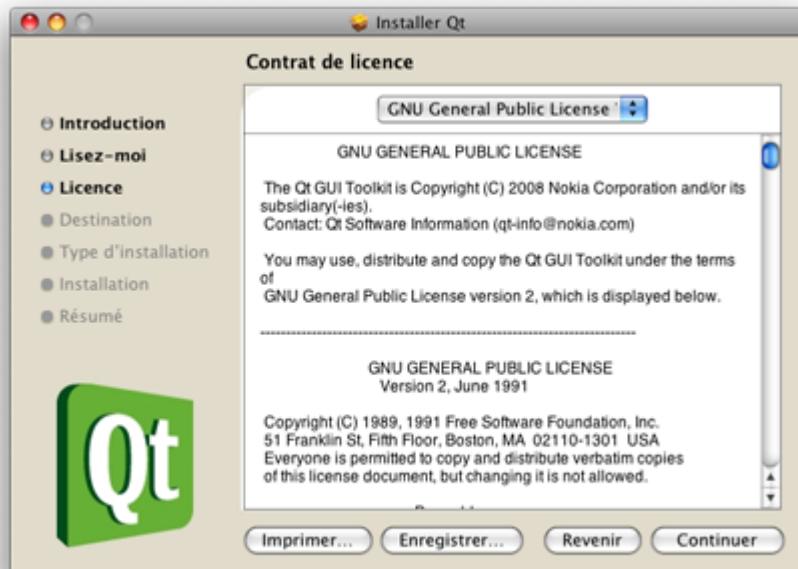
Warning : accept



*Installing Qt 4.4.3 Mac : introduction*



*Installing Qt 4.4.3 on a Mac : information*



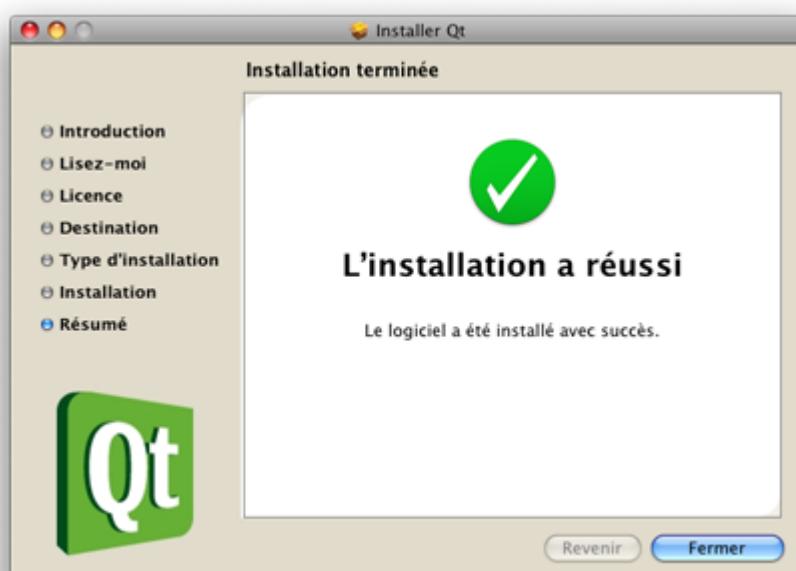
*Installing Qt 4.4.3 on a Mac : license*



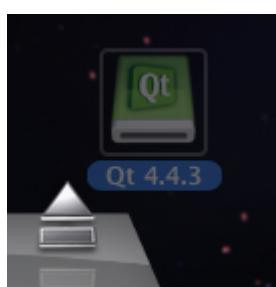
*Installing Qt 4.4.3 on a Mac : installation type and disk space required*



*Installing Qt 4.4.3 on a Mac : writing files*



*Installing Qt 4.4.3 on a Mac : end !*



*Installing Qt 4.4.3 on a Mac : eject disk image*

The default directory where files are written is **/Developer**.



### Installation Qt 4.4.3 Mac : fichiers et dossiers d'installation de Qt

No need to restart, you can go to the next step : your first compilation from the terminal

## III - Your first compilation from the terminal



It is possible to compile sources directly from the terminal, a little bit like under windows (DOS command window) by using the *make* command (*nmake* under windows if you are using Microsoft's compiler).

### III-A - Creation of a source file

Before creating a source file anywhere (usually on the desktop) and take the risk of having a complete mess after generating files , we will create a folder for the projet, what about 'QtProjects' in */Developer* and the famous test folder that every developper has created at least once in his life ? *01\_test* :D

 **WARNING : avoid files and folder names containing spaces !**

Create a new text file and save it as *main.cpp*. Then copy-paste the code below and save again :

```
main.cpp

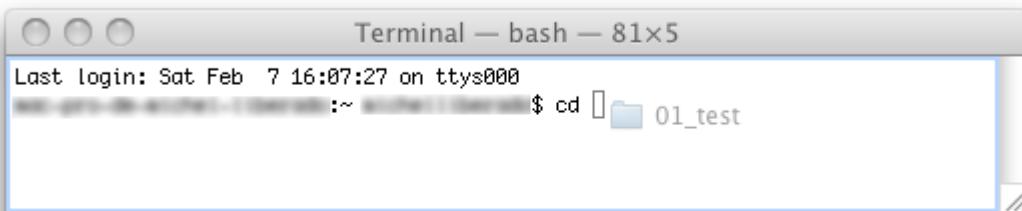
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QPushButton button("Ca marche !");
    button.show();
    return app.exec();
}
```

## III-B - Project file creation and compilation

Now we must create the project file, "qmake" command will do it for us. Open the terminal (call it with Spotlight or under /Applications/Utilities). Go to folder /Developer/QtProjects/01\_test with command "cd /Developer/QtProjects/01\_test".

 You can also type in "cd " and drag the folder 01\_test as shown on the following screenshot :



*Drag and drop of a folder into the terminal, very usefull to get rid of spaces in file/folder names*

Enter the following commands :

**qmake -project** to create the project file (by default 01\_test.pro)

**qmake -spec macx-g++** to create the makefile

**make** to compile

the executable file is created and appears in the same folder as the sources. Here is the result :



*Qt executable test file on Mac*

Before going to the next step, deleted all generated files (keep main.cpp), that is not required but it will train you to enter the commands.

## III-C - Explanations of commands

Before going to the next step, it would be wise to understand the *qmake* command options.

Documentation can be found on Trolltech's website (or Nokia if you prefer) **qmake-running**.

As said in the documentation, in the default mode, qmake will use the description in a project file to generate a Makefile, but it is also possible to use qmake to generate project files.

The fact of generating a project file or a Makefile is called the 'mode'.

The first command using *-project* explicitly specifies to create a project file.

The second one specifies *-spec*. Documentation says :

*-spec spec*

*qmake will use spec as a path to platform and compiler information, and the value of QMAKESPEC will be ignored*

**QMAKESPEC's documentation** says :

*The name of a platform-compiler combination. In this case, qmake will search in the directory specified by the mkspecs subdirectory of the data path specified when Qt was compiled (see QLibraryInfo::DataPath).*

We still have to find the supported combination platform-compiler on Mac. Here is the page : **supported platforms**  
We can find *macx-g++*.

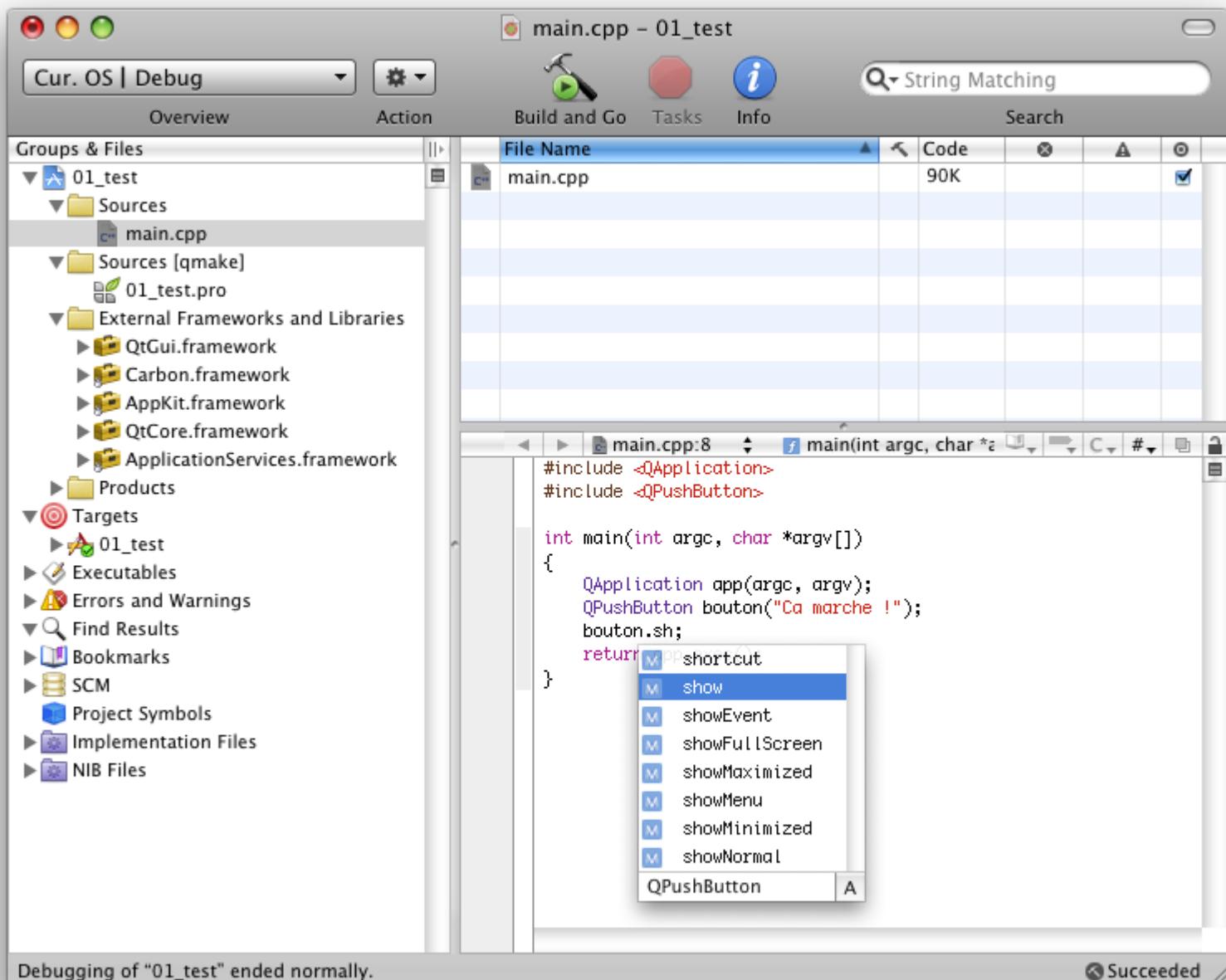
Now that I've made you search everywhere, here is another explanation : **qmake platform notes** :

*Where the source package typically uses the macx-g++ specification, the binary package is typically configured to use the macx-xcode specification* It means that if you don't specify *-spec macx-g++*, qmake will generate an Xcode project file by default.

## IV - Your first compilation with Xcode



The source file is identical to the one in the previous chapter. We'll start from III-B and enter the following commands :  
**qmake -project** to create the project file (by default 01\_test.pro)  
**qmake -spec macx-xcode** to create Xcode project file (01\_test.xcodeproj)  
 Then we launch Xcode project file and .. tada ! :



Qt's integration to Xcode, autocomplete works

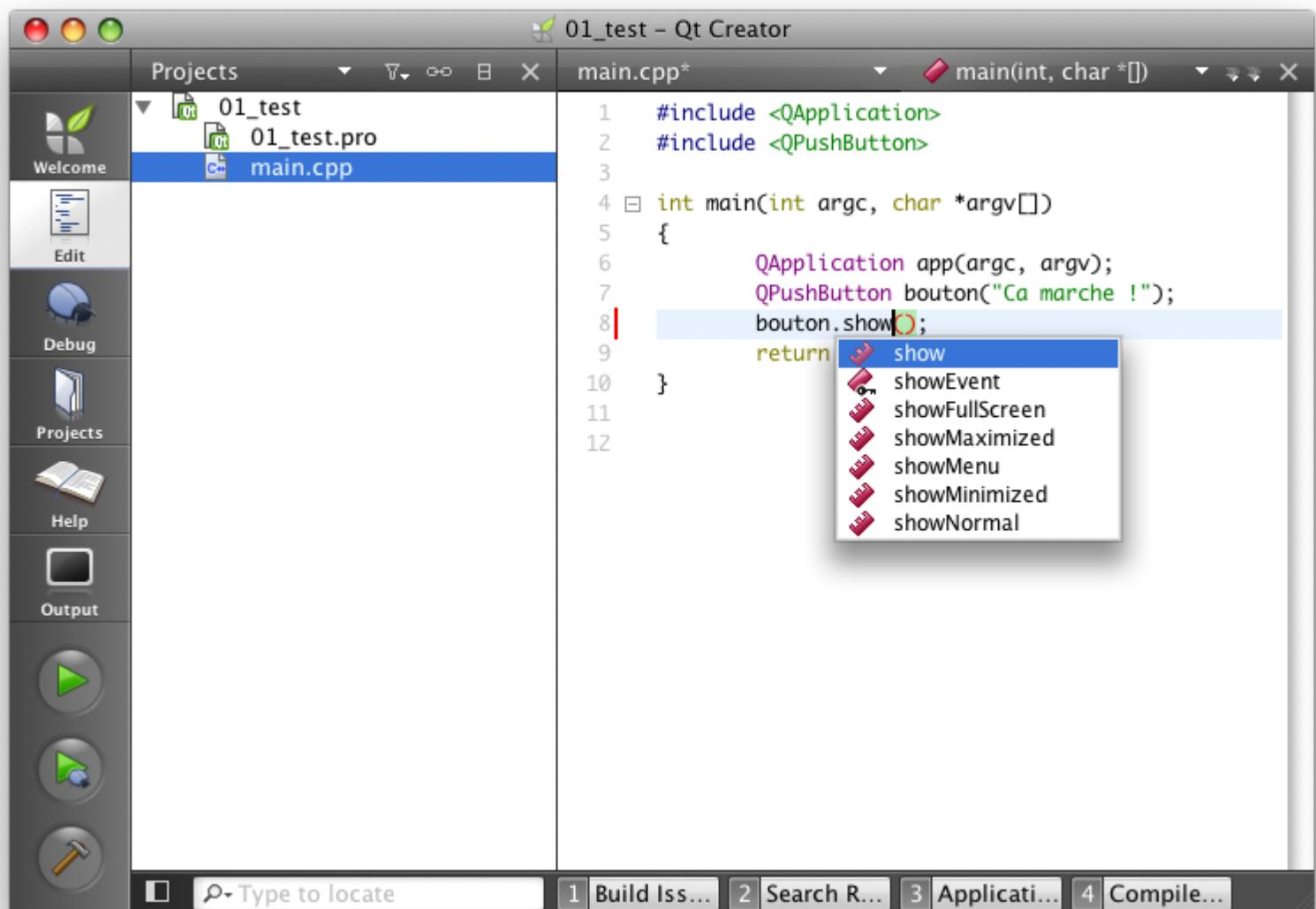
 *The integration is not perfect, the project file is not automatically handled so, when you'll add files to the project, you'll have to regenerate the .pro file. Anyway, sooner or later you'll have to manually edit the .pro file, so it's better to get used to that. However, you can still add an external target and specify the use of qmake with "Custom build command" and changing target dependencies. This tutorial will be updated asap with this trick.*

Click "Build and Go" and that's it !

## V - Your first compilation using Qt Creator



Go back to the two previous chapters but keep the source file AND the .pro project file. Open the .pro file with Qt Creator.



### QtCreator on a Mac

Click the "Run" button and that's it !

## VI - sum up

Whatever your developpement environnement is, you'll have to maintain the .pro project file yourself. Of course, it can be handled automatically if you're using a PC and have a commercial license of Qt with Visual Studio integration. By experience, I can tell that the project file that can be generated by the integration can become quickly useless if you're using environement variables. The consequence is simple : deployment to other platforms becomes much harder. You can find qmake documentation on Qt's website [here : qmake manual](#)

In this tutorial we focused on [Mac OS X qmake specifications](#).