Compilation, installation et utilisation de la librairie Mysql sous Qt

Introduction

Librairie Mysql

La librairie Mysql en C/C++ s'appelle *libmysql*. Elle est constituée de deux fichiers :

- *Libmysql.dll* : la librairie à proprement parlé. C'est une librairie dynamique et partageable
- *Libmysql.lib* : contient uniquement des références vers les fonctions et classes situées dans la dll. Il est nécessaire d'avoir le fichier lib même si on produit un exécutable sans la librairie (cas d'une librairie statique) de façon à pouvoir « linker » son code.

Driver QMYSQL

Indépendamment de la librairie Mysql, il faut charger un driver pour avoir accès à une base de données. Il faut voir le driver QMYSQL comme n'importe quel driver de matériel qu'on chercherait à exploiter dans son application.

Pour charger le driver QMYSQL, il faut l'avoir à sa disposition. Il n'est pas fourni d'origine à l'installation de Qt. On va donc devoir récupérer les sources du driver et compiler celui-ci sous Qt.

Le driver Mysql s'appelle: *qsqlmysql.dll*

En résumé :

La librairie Mysql nous permet d'effectuer les appels de fonctions réalisant des opérations vers la base de données.

Le serveur de base de données est vu comme un « matériel » par Windows.

Il faut donc un driver pour pouvoir y accéder, driver qu'on va recompiler. La première chose à faire dans son code, c'est de charger le driver.

Remarque :

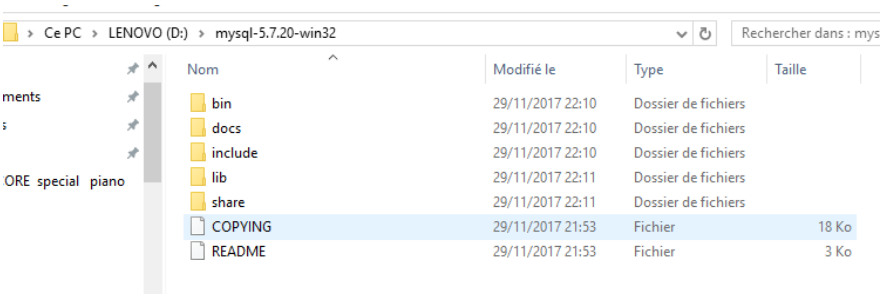
On aura toujours de ensembles de fichiers (librairies et drivers) : une version « Release » et une version « Debug ». En version « Debug », les noms de fichiers possèdent un « d » à la fin, par exemple : *libmysql.lib* pour la version « Release » et *libmysqld.lib* pour la version « Debug »

Librairie Mysql

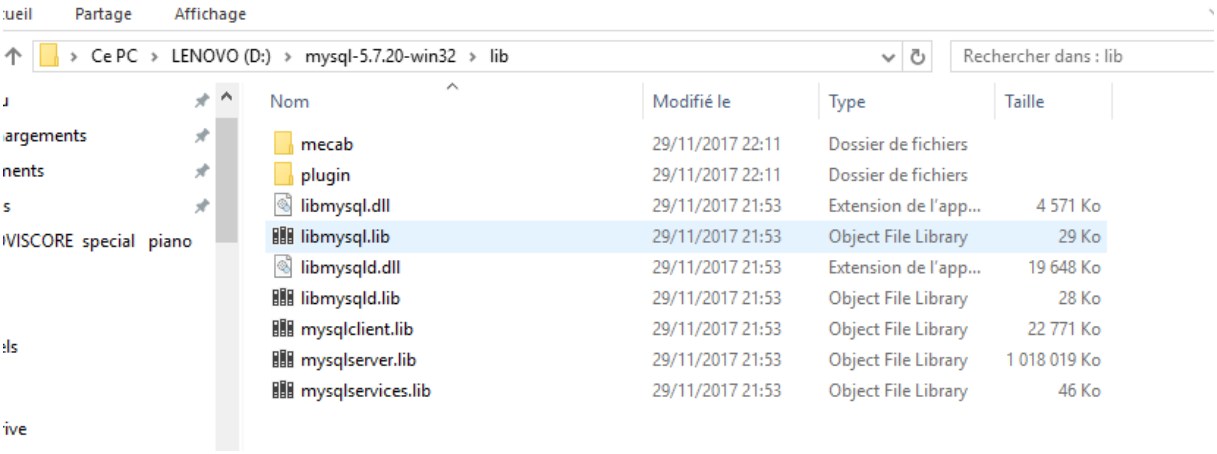
Récupérer le fichier zip sur le site d'Oracle : https://dev.mysql.com/downloads/mysql/

Choisir la dernière version, par exemple : *mysql-5.7.20-win32.zip*

Après l'avoir « dézippé », *placer* l'ensemble dans un répertoire adéquat. Prenez l'habitude de créer un répertoire à part de Qt où vous installez toutes vos librairies. Contenu du répertoire de la librairie :



Dans *lib* :



On retrouve bien les fichiers version « Release » :

libmysql.lib et *libmysql.dll*

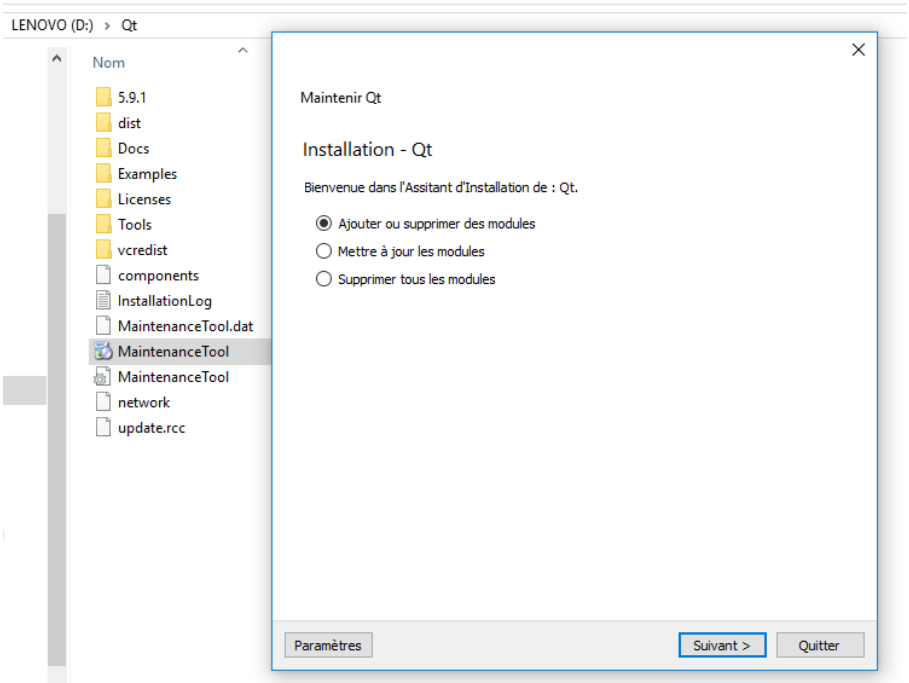
Et les fichiers en version « Debug » :

libmysqld.lib et *libmysqld.dll*

Installation des sources de Qt

Avant de procéder à la compilation du driver QMYSQL, il faut bien entendu disposer du code source de ce dernier.

*Lancer*à partir du répertoire de Qt, l'utilitaire *MaintenanceTools* :



Ici c'est la version 5.9.1 de Qt, mais quelque soit la version vous devriez avoir cet utilitaire dans le répertoire de Qt.

Pour récupérer les sources, il faut paramétrer l'utilitaire en lui donnant l'adresse d'un dépôt où il pourra récupérer ce qu'il manque.

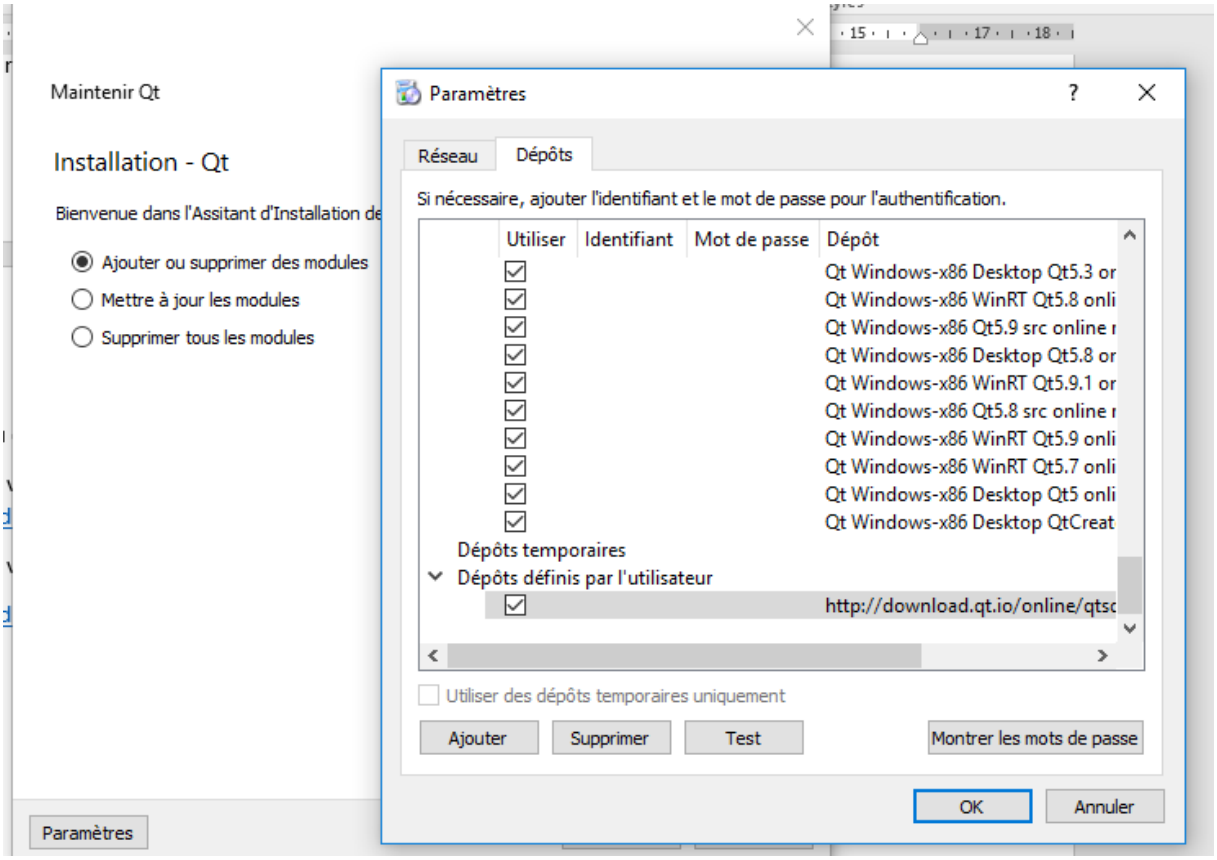
L'url du dépôt dépend de la version de Qt installée !

Pour la version Qt 5.9.1, l'adresse est :
http://download.qt.io/online/qtsdkrepository/windows_x86/desktop/qt5_591_src_doc_examples/

Pour la version Qt 5.8.0, l'adresse est :

http://download.qt.io/online/qtsdkrepository/windows_x86/desktop/qt5_58_src_doc_examples/

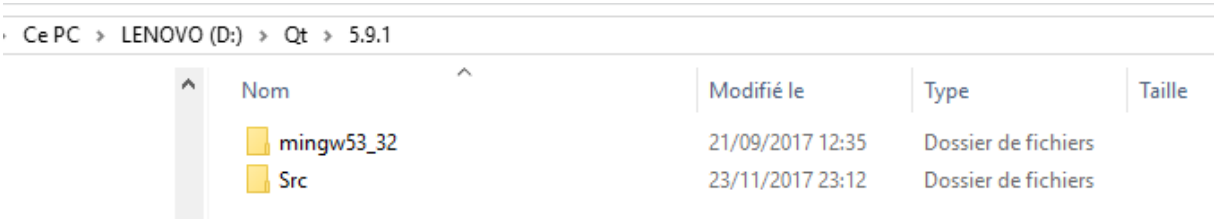
Ajouter l'url du dépôt si nécessaire. En effet, si vous avez un compte chez Qt (ce qui est mon cas) alors les url des dépôts sont déjà renseignés sinon vous pouvez ajouter une url temporaire ou utilisateur



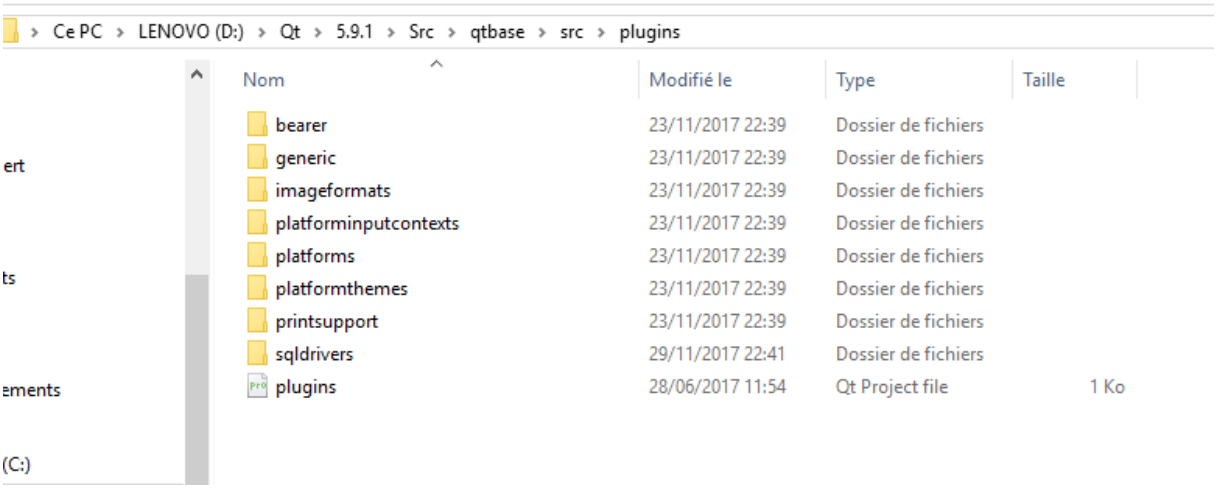
Sélectionner « Ajouter ou supprimer des modules » et cliquer sur « suivant ».

Ouvrir l'arborescence et *sélectionner* « Sources ». L'utilitaire devrait procéder au téléchargement du module sélectionné (ici « Sources ») et procéder à son installation dans le répertoire de Qt adéquat.

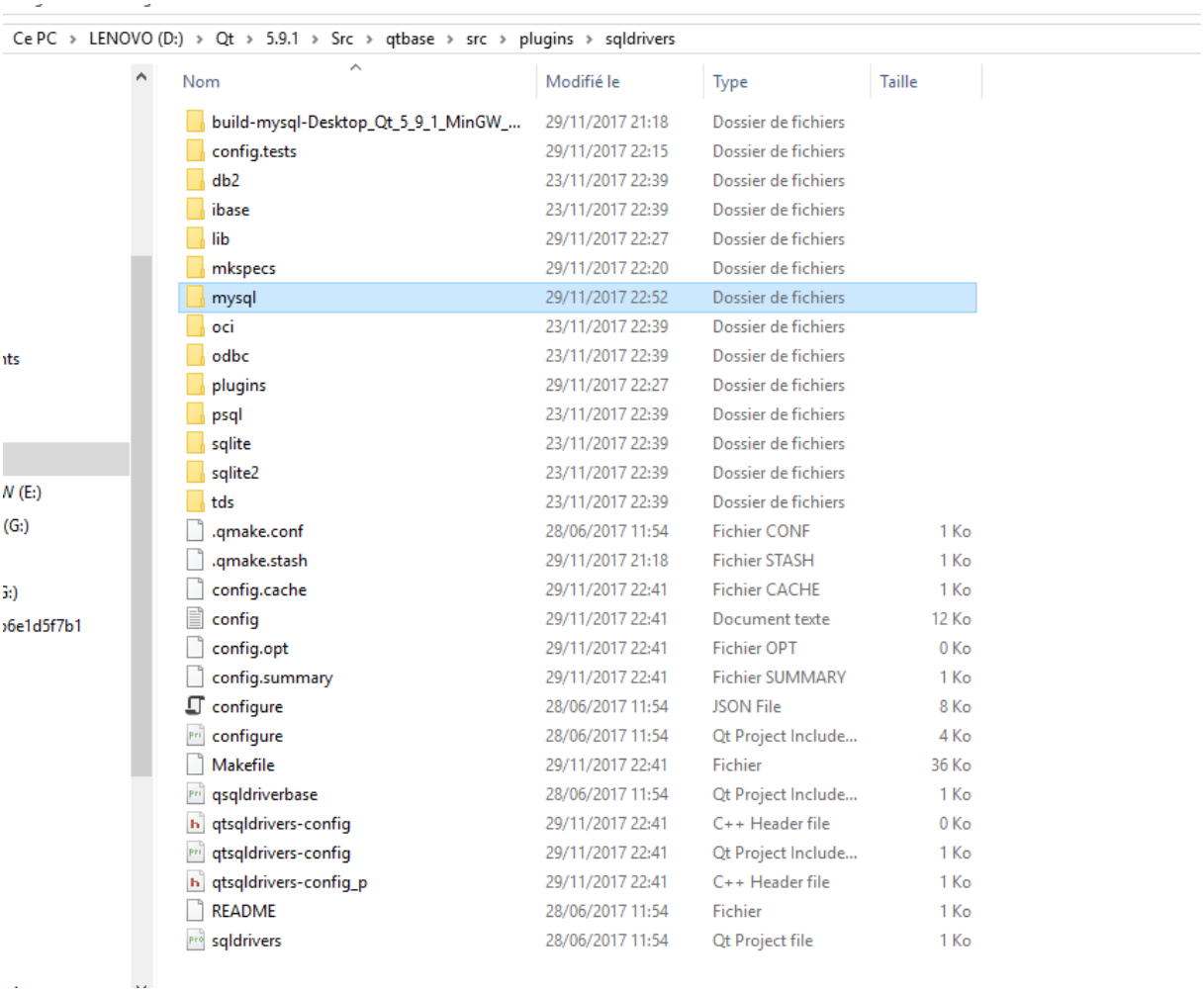
Logiquement, vous devriez avoir un répertoire « Src » supplémentaire dans Qt\5.8.0 (ou chez Qt\5.9.1) :



Désormais, ce qui va nous intéresser c'est ce qu'il y a dans ce répertoire et notamment :



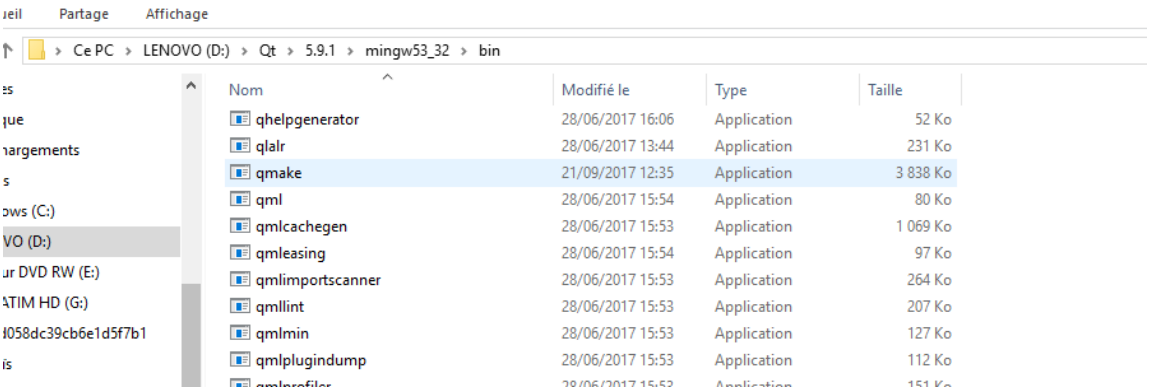
Dans le répertoire « Qt\5.8.0\Src\qtbase\src\plugins » on a un répertoire « sqldrivers » :



Dans ce répertoire « sqldrivers », il y a un fichier .pro « sqldrivers.pro ». Il va nous servir à produire un makefile temporaire qui sera utilisé pour configurer correctement le fichier .pro de mysql.

Faîtes en sorte de pouvoir utiliser le compilateur de fichier .pro de Qt (qmake) dans une fenêtre DOS. Si ça n'est pas le cas, *ajouter* dans le path de Windows le chemin d'accès à qmake (variable d'envrironnement système PATH Windows).

Chez moi c'est :

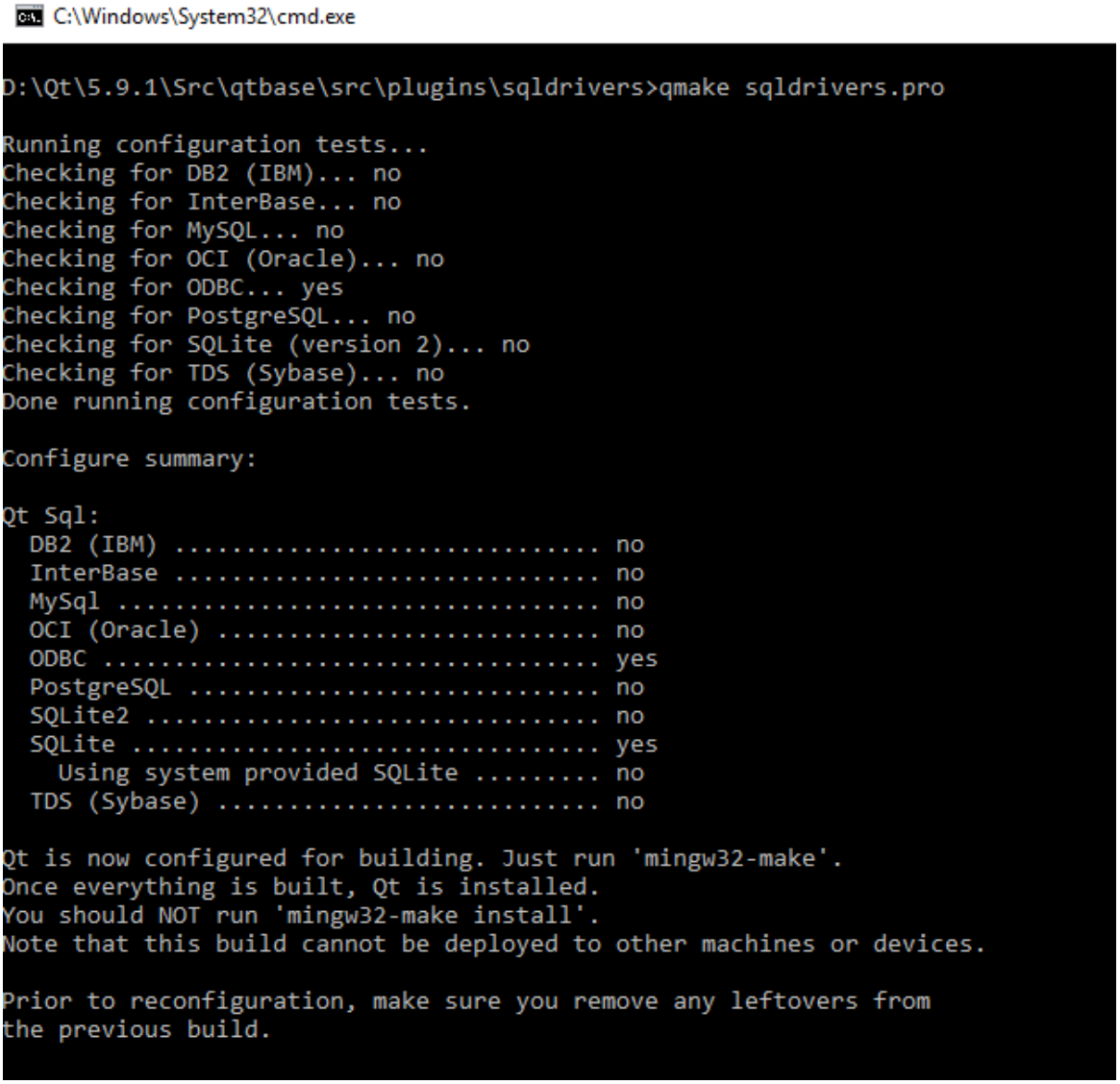


Compilation du driver QMYSQL

Génération des makefile

Ouvrir une fenêtre DOS et *placez vous* dans le répertoire où se situe « sqldrivers.pro »

Lancer la commande « qmake sqldrivers.pro »



A l'aide de « Notepad » ou équivalent, *ouvrez* le fichier « mysql.pro » situé dans le répertoire « sqldrivers » et *commentez* (ajout de # devant) la ligne « QMAKE_USE += mysql » :

```
1   TARGET = qsqlmysql
2
3   HEADERS += $$PWD/qsql_mysql_p.h
4   SOURCES += $$PWD/qsql_mysql.cpp $$PWD/main.cpp
5
6   #QMAKE_USE += mysql
7
8   OTHER_FILES += mysql.json
9
10  PLUGIN_CLASS_NAME = QMYSQLDriverPlugin
11  include(../qsqldriverbase.pri)
12
```

Sauvegarder le fichier « mysql.pro »

Retournez sous DOS, *déplacez vous* dans le répertoire « mysql » situé dans « sqldrivers » et *lancer* la commande « qmake mysql.pro » en précisant le répertoire « include » correspondant à la librairie *libmysql* et la librairie à proprement parler, à utiliser :

```
D:\Qt\5.9.1\Src\qtbase\src\plugins\sqldrivers\mysql>qmake "INCLUDEPATH+=D:\\mysql-5.7.20-win32\\include"
"LIBS+=D:\\mysql-5.7.20-win32\\lib\\libmysql.lib" mysql.pro
```

Attention à bien mettre les « \\ » comme séparateur de répertoires (écriture des chemins dans une chaîne de caractères de type C : pour pas confondre avec « \ » qu'on a quand on fait un « \n »)

Logiquement, vous devriez avoir un Makefile pour la version « Debug » et un Makefile pour la version « Release »

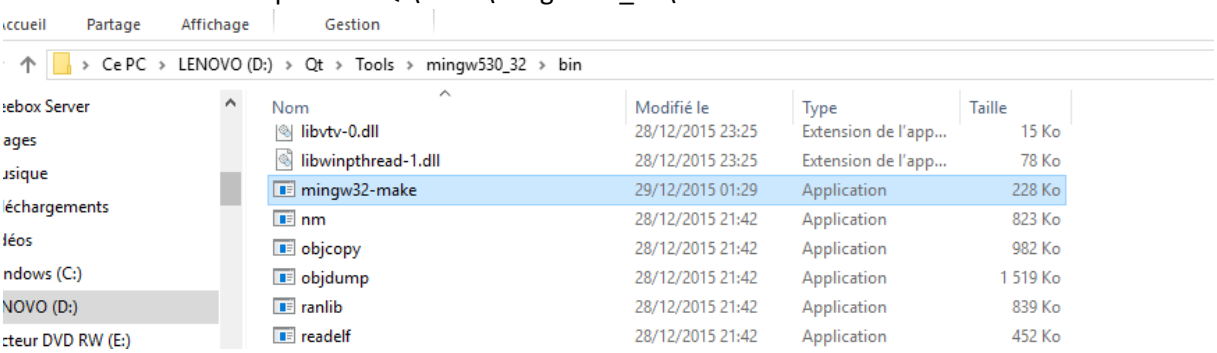
Compilation du driver

Grâce aux makefile produits, on va pouvoir compiler le driver de façon à obtenir le fichier *qsqlmysql.dll*

Toujours dans la fenêtre DOS :

De même que pour « qmake », a*ssurez-vous* de pouvoir lancer la commande « mingw32-make » depuis le DOS. *Ajouter* le chemin si nécessaire dans la variable d'environnement PATH.

Cela se situe dans le répertoire Qt\Tools\mingw530_32\bin :



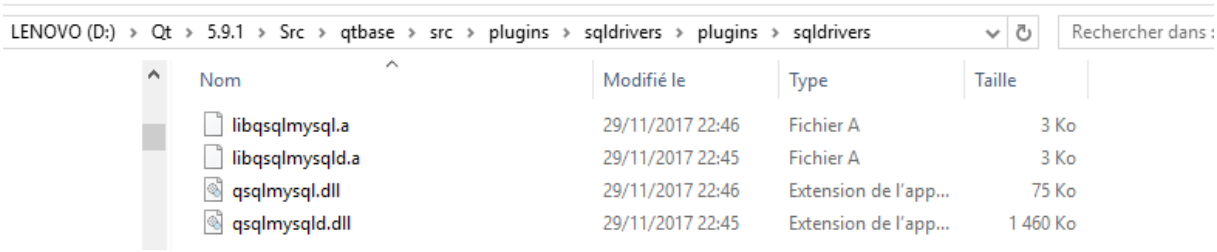
Lancer la compilation version « Debug » : commande « mingw32-make -f Makefile.Debug

```
D:\Qt\5.9.1\Src\qtbase\src\plugins\sqldrivers\mysql>mingw32-make -f Makefile.Debug
```

Procéder de la même façon avec le Makefile.Release

A partir de maintenant, le driver est compilé et utilisable sous Qt.

Vous devriez avoir des fichiers .dll dans le répertoire : ..Qt\5.9.1\Src\qtbase\src\plugins\sqldrivers\plugins\sqldrivers



Remarque : les fichiers .a sont destinés à être utilisés sous linux

Utilisation du driver et de la librairie *libmysql* dans un projet Qt

Arborescence nécessaire

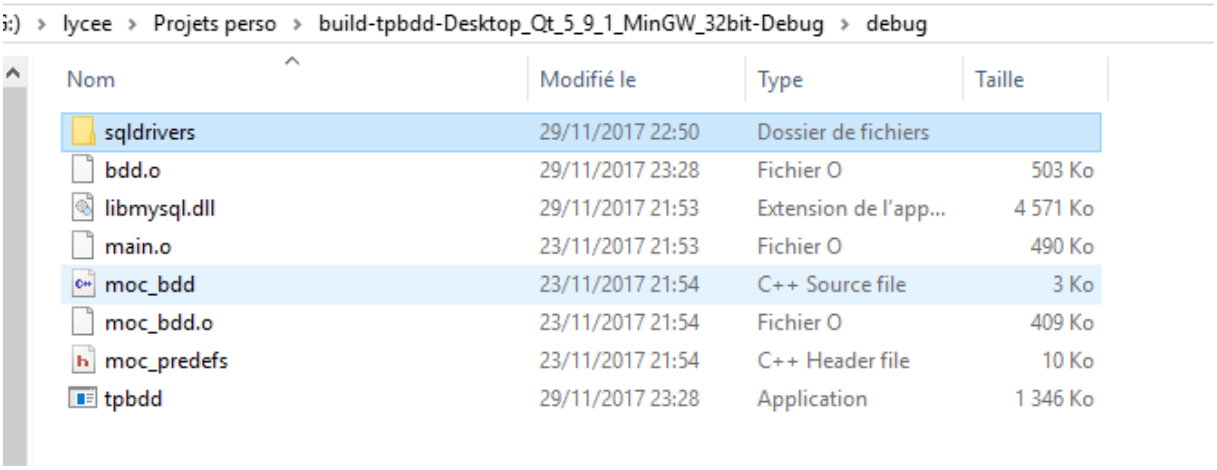
Lorsqu'on crée un projet Qt sous QtCreator, celui-ci génère un répertoire dont le nom commence par « build » et contient le nom du projet; il regroupe tous les fichiers nécessaires à l'exécution de votre projet dans QtCreator. L'exécutable se situera dans ce réperoire.

Ce qui est gênant, c'est qu'il faut produire une première fois l'application même bourrée d'erreurs, même pas finie, afin de l'obliger à créer ce fameux répertoire.

Si on produit le projet sous QtCreator en version « release », il génère un répertoire « build….release »

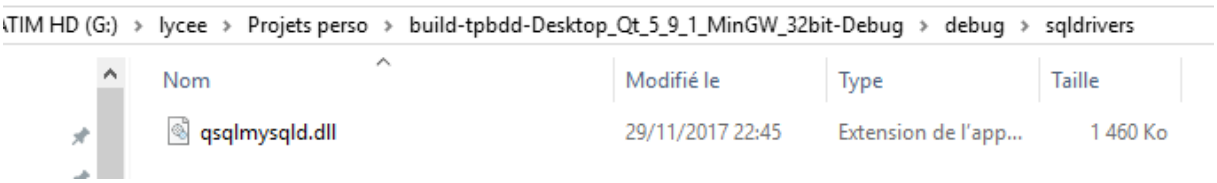
Si on produit le projet sous QtCreator en version « debug », il génère un répertoire « build….debug »

Au même niveau que l'exécutable, il faut créer un répertoire « sqldrivers » dans lequel on copie le fichier « qsqlmysql.dll » (ou qsqlmysqld.dll pour la version « debug »). C'est généralement dans le répertoire « debug » du répertoire « build…debug » :

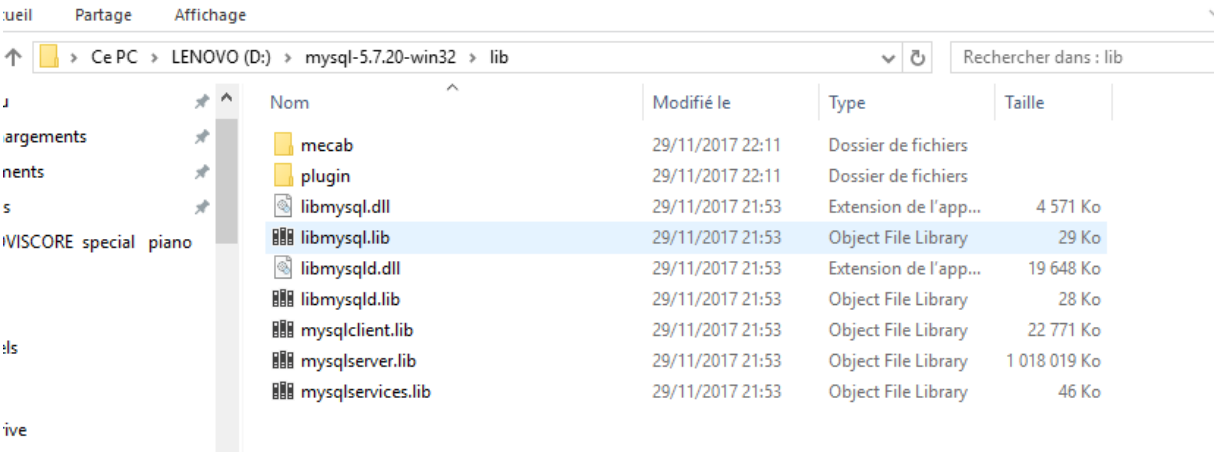


« tpbdd » est mon exécutable de mon projet « tpbdd ».

Dans le répertoire « sqldrivers » précédemment créé, j'ai ajouté le fichier « qsqlmysqld.dll » (ici version « debug ») :



Revenir dans le répertoire « debug » et *ajouter* la dll de la librairie *mysql* : *libmysql.dll* (ou *libmysqld.dll*) située dans le répertoire d'installation de la librairie *mysql* installée au début :



Remarque :

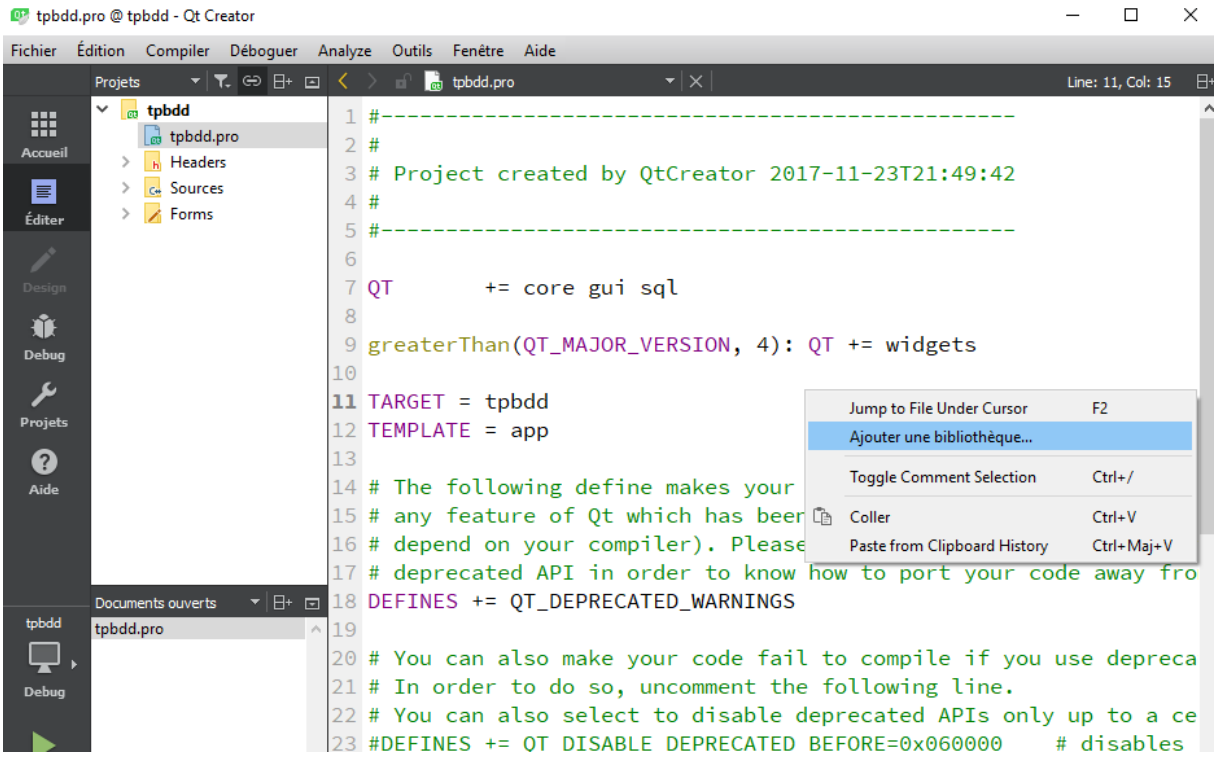
On peut copier la librairie *libmysql.dll* dans debug ou bien *libmysqld.dll* dans debug ; ça n'a pas d'importance à moins qu'on ne veuille débugger la librairie *libmysql*.

Au final on obtient dans le dossier du projet :

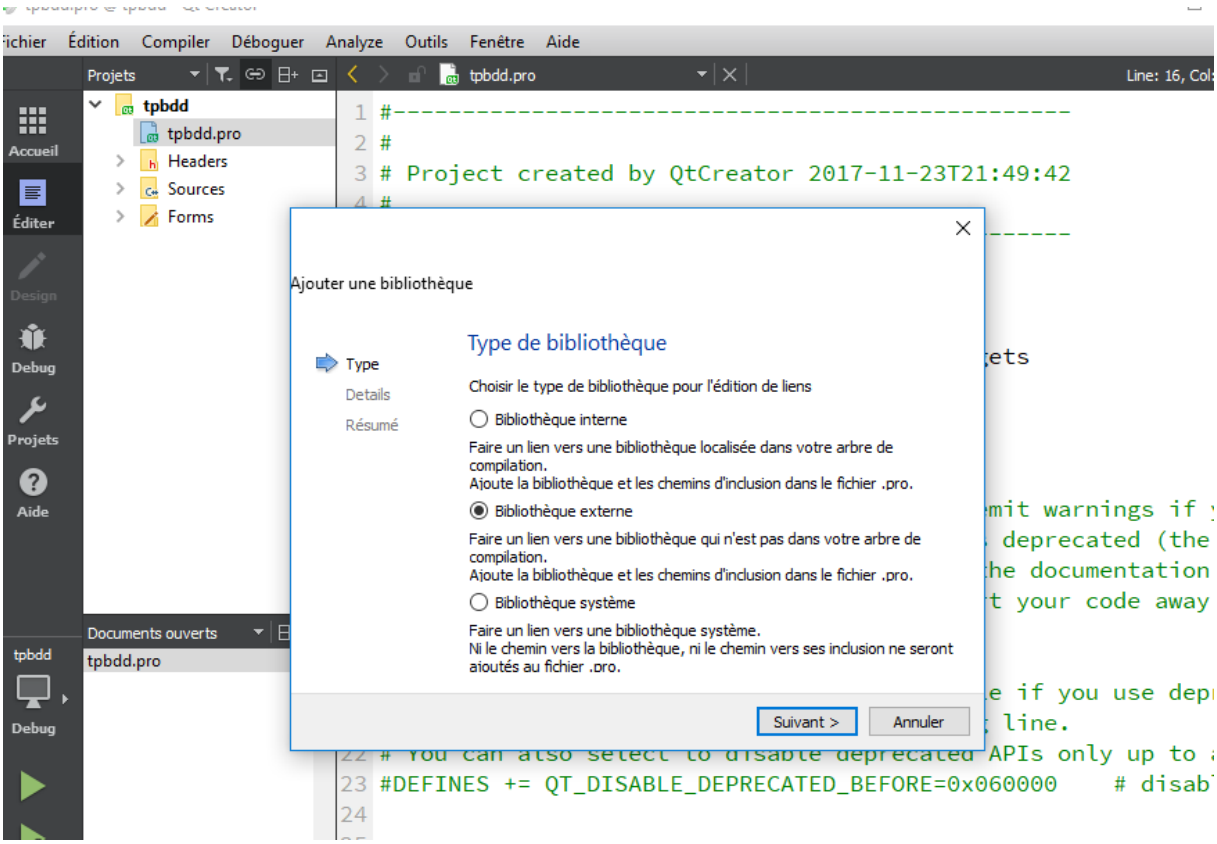
Utilisation de la librairie

Il faut, dans le projet, ajouter la librairie *libmysql*.

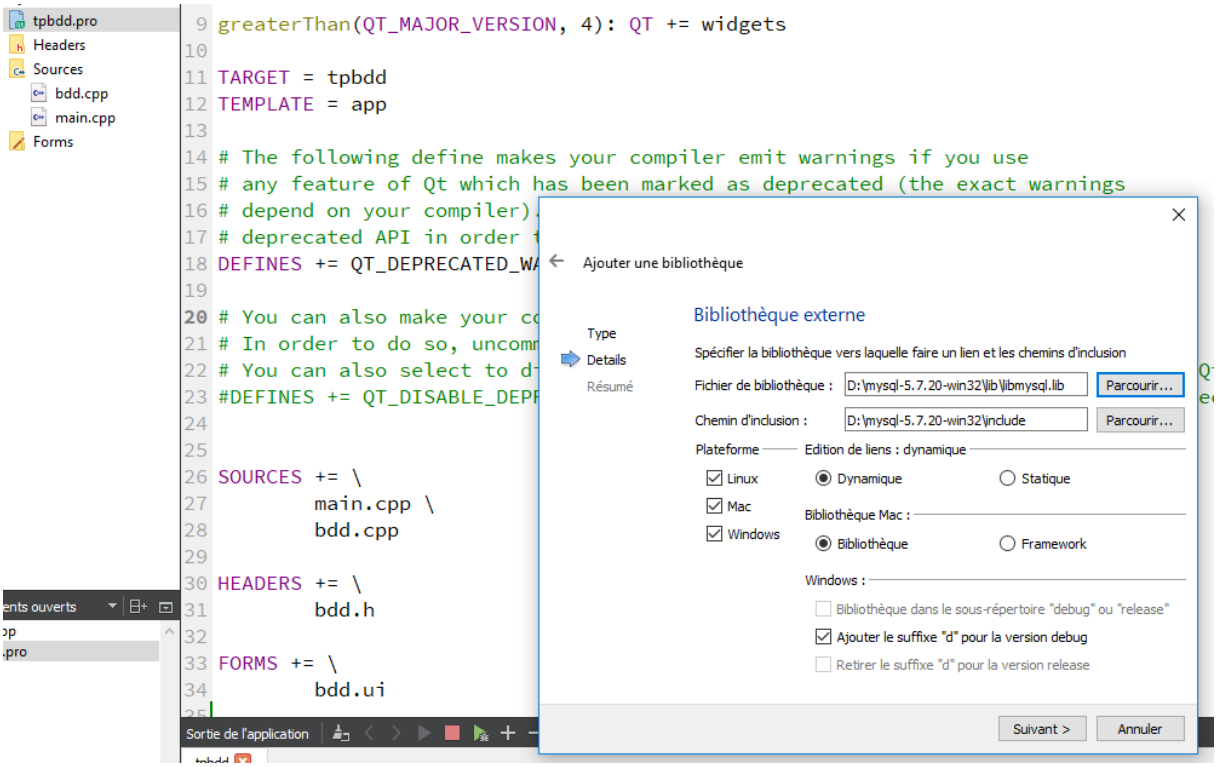
Dans Qtcreator, *ouvrir* le fichier .pro de votre projet. *Ajouter* une bibliothèque à l'aide du bouton droit de la souris :



Choisir une bibliothèque externe :



Cliquer sur « suivant »



Cliquer sur « Parcourir » et *ajouter* la librairie *libmysql.lib*

Essai de connexion à une base de données Mysql

Un code pour essayer :

```
#include "bdd.h"
#include "ui_bdd.h"
#include <QSqlDatabase>
#include <QSqlQuery>
#include <qdebug.h>

Bdd::Bdd(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Bdd)
{
    ui->setupUi(this);

    QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");
    db.setHostName("192.168.0.9");
    db.setDatabaseName("tp");
    db.setUserName("david");
    db.setPassword("david");

    bool ok = db.open();

    if (ok)
    {
        qDebug() << "Connexion Base de donnees OK" << endl;

        QSqlQuery query;
        query.exec("SELECT b FROM table1;");
        while (query.next()) {
                QString name = query.value(0).toString();
                qDebug() << name << endl;
            }
```

```
    }
    else
    {
        qDebug() << "Pas de connexion BDD" << endl;
    }
}
```

La méthode statique “AddDataBase” est celle qui va provoquer le chargement du driver QMYSQL.

Vérifier dans la fenêtre « Sortie de l’application » qu’il n’y a pas d’erreur de chargement du driver QMYSQL. Si tel est le cas alors le driver est soit inacessible (problème de répertoire) soit mal compilé (vérifier alors la trace de compilation dans la fenêtre DOS si elle n’est pas fermée)

« db » désormais représente le moyen de s’interfacer avec la base de données par l’entremise du driver chargé.

Ce code est utilisable si et seulement si vous disposez d’un serveur de base de données, d’une base de données et de tables.

Ici, mon application se connecte à un serveur Mysql ,situé sur la machine d’adresse IP 192.168.0.9 ; dans ce serveur, il y a une base de données « tp » ; dans cette base, il y a une table « table1 » comportant 4 colonnes « a, b, c, et d »)