برای این که بتونم دستور locate رو توی ویندوز اجرا کنم یک اسکریپت رو به صورت زیر دانلود کردم و در مسیر زیر قرار دادم.

C:\Users\naji\Downloads\Invoke-Locate.ps1 قرار دادم. و با استفاده از دستور زیر توی powerShell اجراش کردم تا بتونم یک دیتابیس داشته باشم که همه‌ی فایل‌هام درون آن باشند.

powershell -ExecutionPolicy ByPass -File .\Invoke-Locate.ps1 -install.

چندین دقیقه طول می‌کشه تا این فایل ایجاد بشه چون تعداد فایل‌هامون به شدت زیاده.

البته این روش جواب نداد و مجبور شدم برای ساختن دیتابیس به صورت زیر عمل کنم.

#include <QDirIterator>

#include <QFile>

#include <QString>

#include <QDebug>

#include <QTextStream>

using namespace std;

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

{

QDirIterator it("C:\Qt\Qt5.11.0", QStringList() << "\*\*", QDir::Files, QDirIterator::Subdirectories);

vector<QString> list;

QFile file("C:/file\_data/file\_name.txt");

file.*open*(QIODevice::WriteOnly);

QTextStream in(&file);

while (it.hasNext())

{

in<<it.next()<<endl;

}

}

و برای جستجو در دیتابیس نیز به صورت زیر عمل کنم.

#include <QDirIterator>

#include <QFile>

#include <QString>

#include <QDebug>

#include <QTextStream>

using namespace std;

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

{

QFile file("C:/file\_data/file\_name.txt");

file.*open*(QIODevice::ReadOnly);

QTextStream in(&file);

QString like="sql.dll";

while(!in.atEnd())

{

QString line=in.readLine();

if(line.contains(like))

{

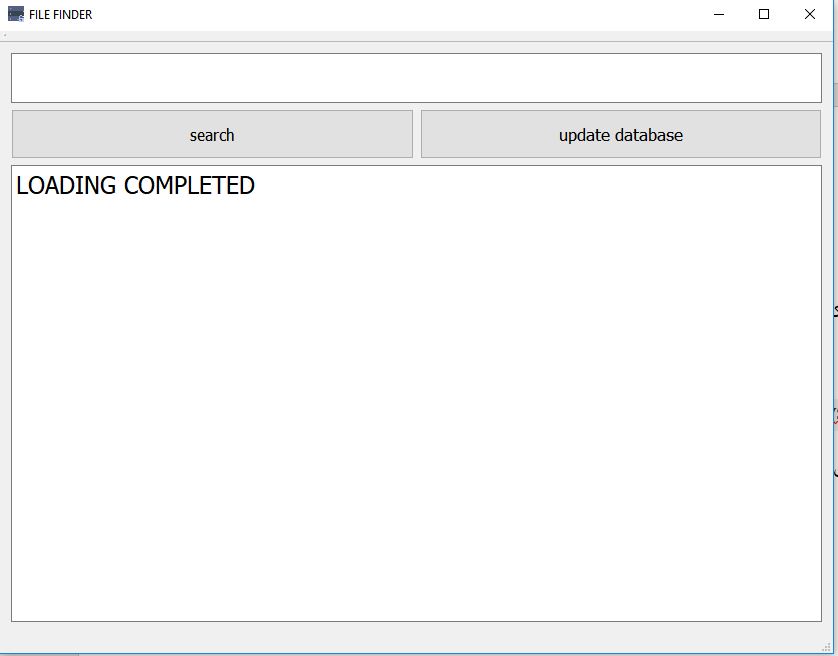
qDebug()<<line;

}

}

}

بعدا این فایل رو با برنامه زیر که به صورت گرافیکی کار می‌کند جایگزین کردم.



در ادامه یک مثال عملی از این نمونه رو خواهم گفت.

در هر صورت متوجه شدم که dll ای به نام libmysql رو ندارم.

با دانلود یک ورژن پایین تر تونستم پیدا بکنم این لایبراری رو. ولی با همدیگه تطابق ندارند و فعلا نتونستم خروجی بگیرم.

توی کیوت وقتی از پلاگین‌های کیوت استفاده می‌کنیم ممکنه که خب یه سری پیغام‌های کمک کننده به ما بده که باید برای دیدن این پیغام‌ها اون رو فعال بکنیم برای این منظور توی فایل main.cpp برنامه تون تک خط زیر رو اضافه کنید.(دقیقا بعد از main اضافه کردم).

qputenv("QT\_DEBUG\_PLUGINS", QByteArray("1"));

با یک بار اجرای برنامه یک پیغام به صورت زیر به ما میدهد.

Cannot load library C:\Qt\Qt5.9.3\5.9.3\msvc2015\_64\plugins\sqldrivers\qsqlmysqld.dll: The specified module could not be found.

QLibraryPrivate::loadPlugin failed on "C:/Qt/Qt5.9.3/5.9.3/msvc2015\_64/plugins/sqldrivers/qsqlmysqld.dll" : "Cannot load library C:\\Qt\\Qt5.9.3\\5.9.3\\msvc2015\_64\\plugins\\sqldrivers\\qsqlmysqld.dll: The specified module could not be found."

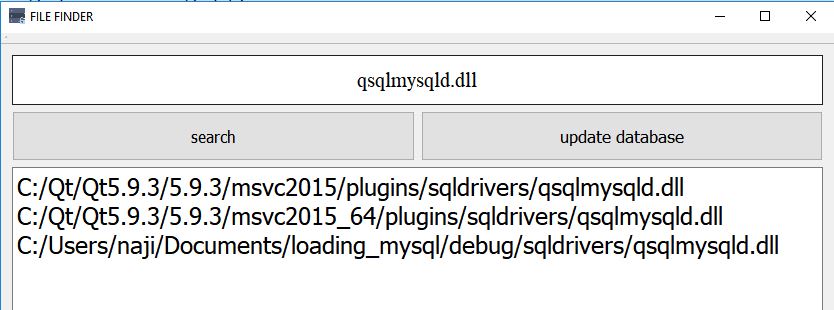
QSqlDatabase: QMYSQL driver not loaded

QSqlDatabase: available drivers: QSQLITE QMYSQL QMYSQL3 QODBC QODBC3 QPSQL QPSQL7

false

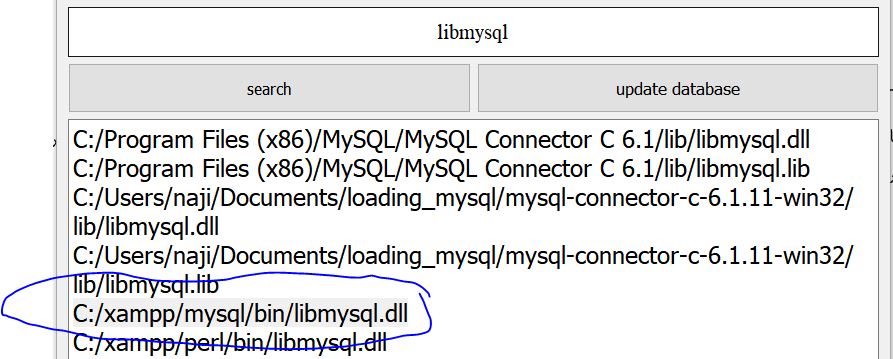
همانطور که مشخصه اشاره شده که سیستم توی مسیر "C:/Qt/Qt5.9.3/5.9.3/msvc2015\_64/plugins/sqldrivers/qsqlmysqld.dll"

به دنبال dll مربوطه بوده ولی نتوانسته آن را پیدا بکند حالا میایم و این dll رو توی سیستمی که خودمون نوشتیم جستجو می‌کنیم.



خب من چون از نسخه‌ی 64 بیتی استفاده می‌کنم لازم هست که دومی رو به اون مسیری که مورد احتیاجه اضافه کنم.

حال باید کانتکتور زبان mysql رو به زبان c نیز اضافه کنم که dllش رو قبلا دیدیم. در زیر من این فایل رو سرچ کردم و اونی که مارک کرده رو توی پوشه‌ی bin کیوت کپی کردم.



برای پیدا کردن depend های یک dll توی ویندوز می‌تونیم از ابزاری که visual studio در اختیار ما قرار داده استفاده کنیم.

C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin>dumpbin.exe /dependents C:\Qt\Qt5.9.3\5.9.3\msvc2015\_64\plugins\sqldrivers\qsqlmysql.dll

که این فایل خروجی زیر را به ما میدهد.

Image has the following dependencies:

Qt5Sql.dll

Qt5Core.dll

libmysql.dll

KERNEL32.dll

VCRUNTIME140.dll

api-ms-win-crt-heap-l1-1-0.dll

api-ms-win-crt-runtime-l1-1-0.dll

Summary

1000 .data

1000 .gfids

1000 .pdata

1000 .qtmetad

6000 .rdata

1000 .reloc

1000 .rsrc

8000 .text

1000 .tls

خب تا بدینجای کار یک چیز جالبی رو فهمیدم وقتی که وقتی connector من 32 بیتی هست پس باید بر روی کامپایلر 32 بیتی خروجی بگیرم.

خب من توی صقحه‌ی web م رفتم و پسورد رو برای اون دیتابیس برداشتم و با کد زیر بهش دسترسی پیدا کردم.

کد اول نشون دادن یک sql توی صفحه

QSqlDatabase db;

db=QSqlDatabase::addDatabase("QMYSQL");

db.setHostName("localhost");

db.setPort(3306);

db.setDatabaseName("OCR");

db.setUserName("mir");

db.setPassword("");

qDebug()<<db.open();

ui->lineEdit->setText(db.lastError().text());

QSqlTableModel \*model = new QSqlTableModel(parent,db);

model->*setTable*("plates");

model->*select*(); //< fetch data

ui->tableView->setModel(model);

نکته: کانکتور و یا libmysql توی حالت 64 بیتی معمولا کانکت نمیشه ولی می‌تونید خودتون این dll رو دانلود کنید و ازش استفاده کنید و اون رو توی پوشه‌ی bin در حالت 64 بیت کپی کنید(برای ریلیز و دیباگ هم جواب میده) [توی این لینک](http://s8.picofile.com/file/8342470600/libmysql.dll.html) می‌تونید این لایبراری رو دانلود کنید.

فرض کنید که بخواهیم ستون دوم رو به صورت کامل چاپ کنیم. برای این منظور می‌توانیم به صورت زیر کد بزنیم.

QSqlQuery query("SELECT \* FROM plates");

while (query.next()) {

QString country = query.value(1).toString();

qDebug()<<country;

}

اگر بخواهیم ستون دوم مقداری رو پرینت کند که در اون id برابر با یک هست.

QSqlQuery query("SELECT \* FROM plates WHERE(plates.id=1)");

while (query.next()) {

QString country = query.value(1).toString();

qDebug()<<country;

}

اضافه کردن یک مقدار به انتهای table به صورت زیر امکان پذیر می‌باشد.

QSqlQuery queryi;

queryi.prepare("INSERT INTO plates(plate\_num,plate\_letter) values(123456789,'abcdc678910a')");

queryi.exec();

انتخاب ستون pelak و codeMelli

QSqlQuery query("SELECT pelak,codeMelli FROM cars");

while (query.next()) {

QString pelak = query.value(0).toString();

QString codeMelli = query.value(1).toString();

}

برای افزایش سرعت برنامه می‌تونیم از متدهای prepare به صورت زیر استفاده کنیم.

QSqlQuery query;

query.prepare("SELECT pelak,codeMelli FROM cars WHERE pelak=:pel");

for (int i = 0; i < 9; ++i) {

double t1=getTickCount();

query.bindValue(":pel",QString("41065272%1").arg(i));

query.exec();

while (query.next()) {

QString pelak = query.value(0).toString();

QString codeMelli = query.value(1).toString();

qDebug()<<pelak<<","<<codeMelli;

}

cout<<"second query time="<<(getTickCount()-t1)/getTickFrequency()<<endl;

}

استفاده در حالت کلاسی: برای این منظور می‌بایست به صورت پوینتری از این کلاس‌ها استفاده شود. یعنی کلاس QSqlQuery را به صورت کلاسی استفاده کرد.

برای لایک گرفتن از دستور می‌بایست مواظب حرف ی و ک باشیم باید ی به صورت ي باشد که با استفاده از دستور shift+c می‌توان آن را زد.

SELECT \* FROM `cars` WHERE `name` LIKE "%رحيم معصومي%"

برای حذف همه‌ی درجیات در یک لیست از دستور زیر استفاده می‌شود.

query->prepare("TRUNCATE TABLE `files`.`c`");

query->exec();

برای ایجاد چند کانشکن به یک دیتابیس باید نام‌ آن‌ها را به صورت جدا از هم تعریف کنیم.

db=QSqlDatabase::addDatabase("QMYSQL","new\_s");

db2=QSqlDatabase::addDatabase("QMYSQL","new\_connection");

SELECT \* FROM `cars` WHERE SUBSTRING(`pelak`,2,1)='9' AND SUBSTRING(`pelak`,3,2)='12' AND SUBSTRING(`pelak`,5,1)='1' AND SUBSTRING(`pelak`,6,1)='9' AND system!='سايپا' AND system!='تيبا' AND system!='رنو' AND system!='سمند' AND system!='پرايد' LIMIT 0,300

اجرای windeployqt به صورت اتوماتیک : برای این کار می‌توانیم از تکه کد زیر در انتهای کد استفاده کنیم.

TARGET\_CUSTOM\_EXT = .exe

DEPLOY\_COMMAND = windeployqt

CONFIG( debug, debug|release ) {

# debug

DEPLOY\_TARGET = $$shell\_quote($$shell\_path($${OUT\_PWD}/debug/$${TARGET}$${TARGET\_CUSTOM\_EXT}))

} else {

# release

DEPLOY\_TARGET = $$shell\_quote($$shell\_path($${OUT\_PWD}/release/$${TARGET}$${TARGET\_CUSTOM\_EXT}))

}

QMAKE\_POST\_LINK = $${DEPLOY\_COMMAND} $${DEPLOY\_TARGET}

تردینگی که نوشته بودم به صورت زیر بود.

THREADS.clear();

for (int j = 0; j < plates.size(); j=j+4)

{

for (int i = j; i < mina(j+4,plates.size()); ++i)

{

THREADS.push\_back(std::thread(&OCR\_S::ocr\_of\_every\_signle\_plate,this,temp,plates[i],i));

}

for (int i = j; i < mina(j+4,plates.size()); ++i)

{

THREADS[i].join();

}

}

درون تابع نیز به صورت زیر می‌باشد.

switch (plate\_order%4) {

case 0:

result=detector1.detect(low\_res,1.0f/255.0,Size(416,416),Scalar(0,0,0),false,false,0.01);

break;

case 1:

result=detector2.detect(low\_res,1.0f/255.0,Size(416,416),Scalar(0,0,0),false,false,0.01);

break;

case 2:

result=detector3.detect(low\_res,1.0f/255.0,Size(416,416),Scalar(0,0,0),false,false,0.01);

break;

case 3:

result=detector4.detect(low\_res,1.0f/255.0,Size(416,416),Scalar(0,0,0),false,false,0.01);

break;

default:

break;

}

خل خوار بعی