网易云课堂

<https://study.163.com/course/courseMain.htm?courseId=1209407801>

# windows上使用git命令行

<https://blog.csdn.net/Aries406532/article/details/90703587>

**Qt学习之路13--布局管理器（QBoxLayout）**

<https://blog.csdn.net/tqs_1220/article/details/81841571>

弹簧

<https://blog.csdn.net/xiezhongyuan07/article/details/81111829>

# QT的自动滚动区QScrollArea的用法，图文详解

<https://blog.csdn.net/qq_31073871/article/details/83117430>

# Qt之QListView使用

**<https://blog.csdn.net/pingis58/article/details/82759207>**

# 用Qt实现基于网络请求的用户登录

**<https://blog.csdn.net/qq_35153254/article/details/78233002>**

编译错误：

10:14:27: 为项目QTPrjOne执行步骤 ...

**10:14:27: Could not create directory "C:\Users\JRWX\Desktop\MQL\QTPrjOne\Error in " Util.asciify("build-QTPrjOne-Desktop\_Qt\_5\_8\_0\_MSVC2015\_32bit-Debug")": TypeError: Property 'asciify' of object Core::Internal::UtilsJsExtension(0xbdba328) is not a function"**

Error while building/deploying project QTPrjOne (kit: Desktop Qt 5.8.0 MSVC2015 32bit)

The kit Desktop Qt 5.8.0 MSVC2015 32bit has configuration issues which might be the root cause for this problem.

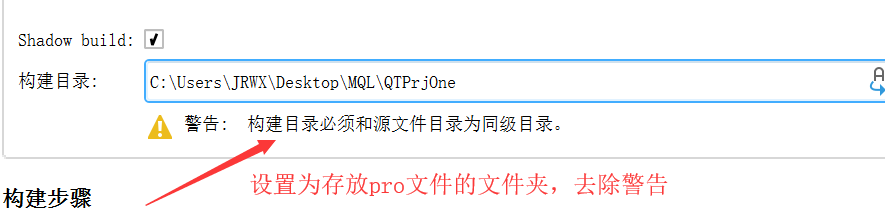
When executing step "qmake"

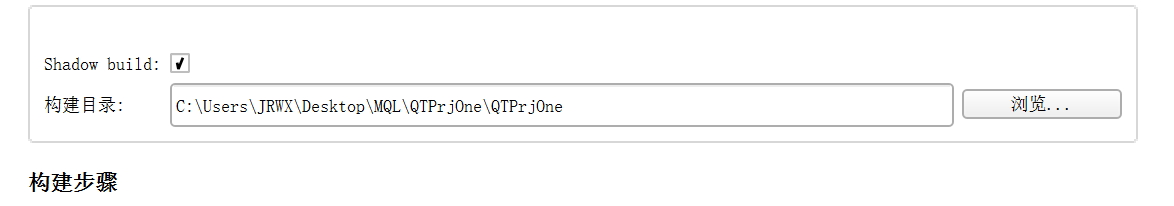
10:14:27: Elapsed time: 00:00.

解决方案：



将上面修改成绝对路径





C++中#ifndef, #define, #endif的作用和使用的注意事项，#开头的都是预编译指令，就是在正式编译之前，编译器做一些预处理的工作。防止重复引入，重复编译（报重定义的错误）。预处理指令是在编译器进行编译之前进行的操作.预处理过程扫描源代码，对其进行初步的转换，产生新的源代码提供给编译器。可见预处理过程先于编译器对源代码进行处理。在很多编程语言中，并没有任何内在的机制来完成如下一些功能：在编译时包含其他源文件、定义宏、根据条件决定编译时是否包含某些代码(防止重复包含某些文件)。要完成这些工作，就需要使用预处理程序。尽管在目前绝大多数编译器都包含了预处理程序，但通常认为它们是独立于编译器的。预处理过程读入源代码，检查包含预处理指令的语句和宏定义，并对源代码进行响应的转换。预处理过程还会删除程序中的注释和多余的空白字符。

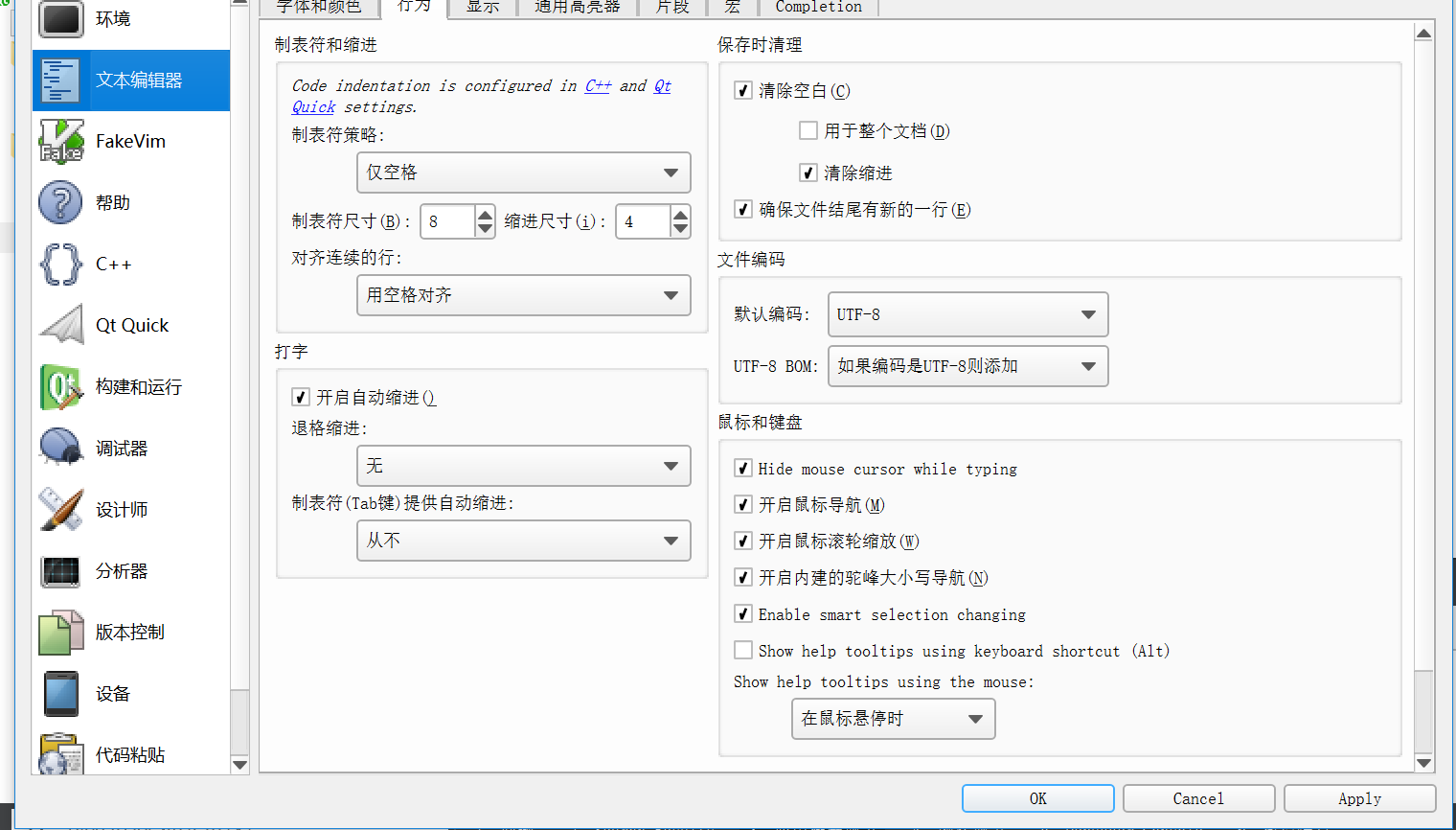
Q\_OBJECT宏，允许类中使用信号和槽的机制

编译错误：

文件是按钮utf-8编码保存的，如下：



并且作了QT Creater的公共设置，如下：



但是在cpp中存在中文时，编译出现错误“常量中有换行符”

解决方案:

原因定位在编译器，vs的编译器不能识别中文。再做了如上公共设置之后，需要再敲几个中文，再编译就好了。

编译好了，但是不管UI上的中文，还是控制台打印的中文，输出还是乱码，就需要在源码里使用编码函数解决了：

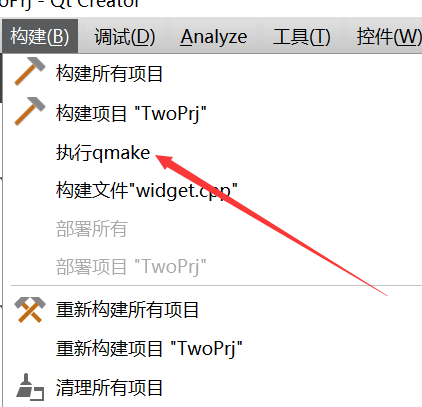
btn->setText(QStringLiteral("自定义按钮"));

编译错误：

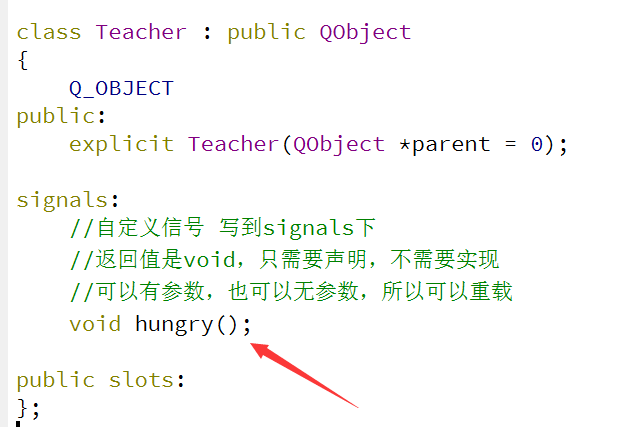
我创建了一个教师类，并且在widget中创建了该对象，编译时

widget.obj:-1: error: LNK2019: 无法解析的外部符号 "public: \_\_thiscall Teacher::Teacher(class QObject \*)" (??0Teacher@@QAE@PAVQObject@@@Z)，该符号在函数 "public: \_\_thiscall Widget::Widget(class QWidget \*)" (??0Widget@@QAE@PAVQWidget@@@Z) 中被引用

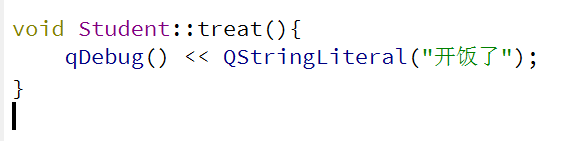
解决方案:

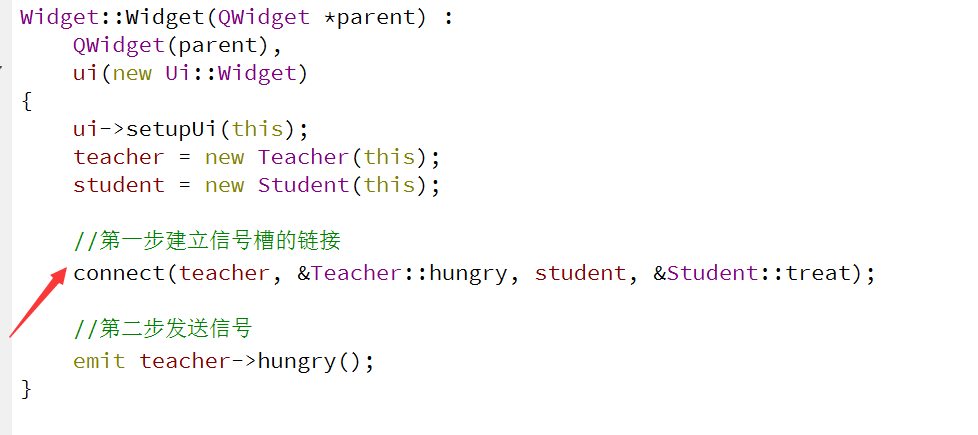


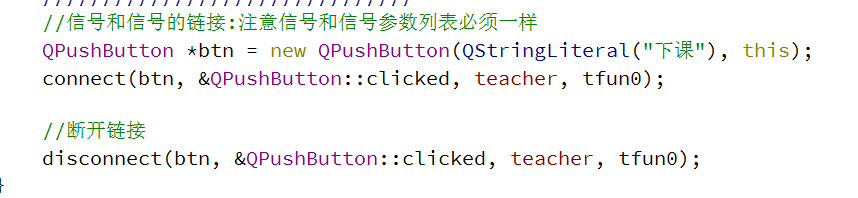
信号槽

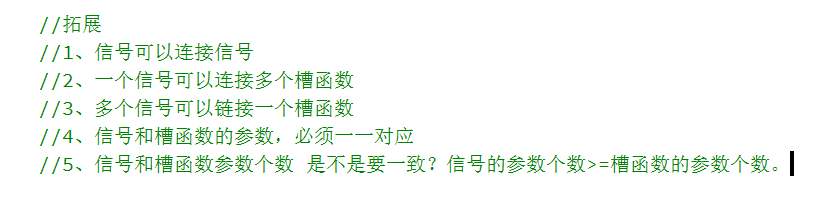






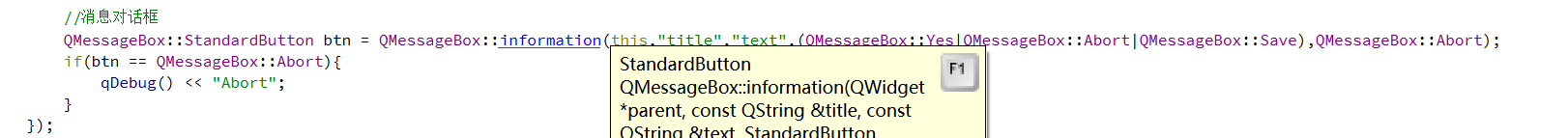














问题：

在可视化设计模板中的控件和运行起来后大小不一致

解决方案：

QT对高分辨率屏幕的显示支持问题，在mian函数中一开始加上#if (QT\_VERSION >= QT\_VERSION\_CHECK(5, 6, 0))  
QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  
#endif

void f(vector<int> &v)

{

v[5]; *// A*

v.at[5]; *// B*

}

如果v非空，A行和B行没有任何区别。如果v为空，B行会抛出std::out\_of\_range异常，A行的行为未定义。

c++标准不要求vector<T>::operator[]进行下标越界检查，原因是为了效率，总是强制下标越界检查会增加程序的性能开销。设计vector是用来代替内置数组的，所以效率问题也应该考虑。不过使用operator[]就要自己承担越界风险了。

如果需要下标越界检查，请使用at。但是请注意，这时候的性能也是响应的会受影响，因为越界检查增加了性能的开销。

重载的信号或重载的槽

QStringList strList;

strList.append(QStringLiteral("宝马"));

strList.append(QStringLiteral("奥迪"));

strList.append(QStringLiteral("奔驰"));

ui->comboBox->addItems(strList);

//void currentIndexChanged(int index);

//void currentIndexChanged(const QString &);

void (QComboBox::\*fun)(int) = &QComboBox::currentIndexChanged;

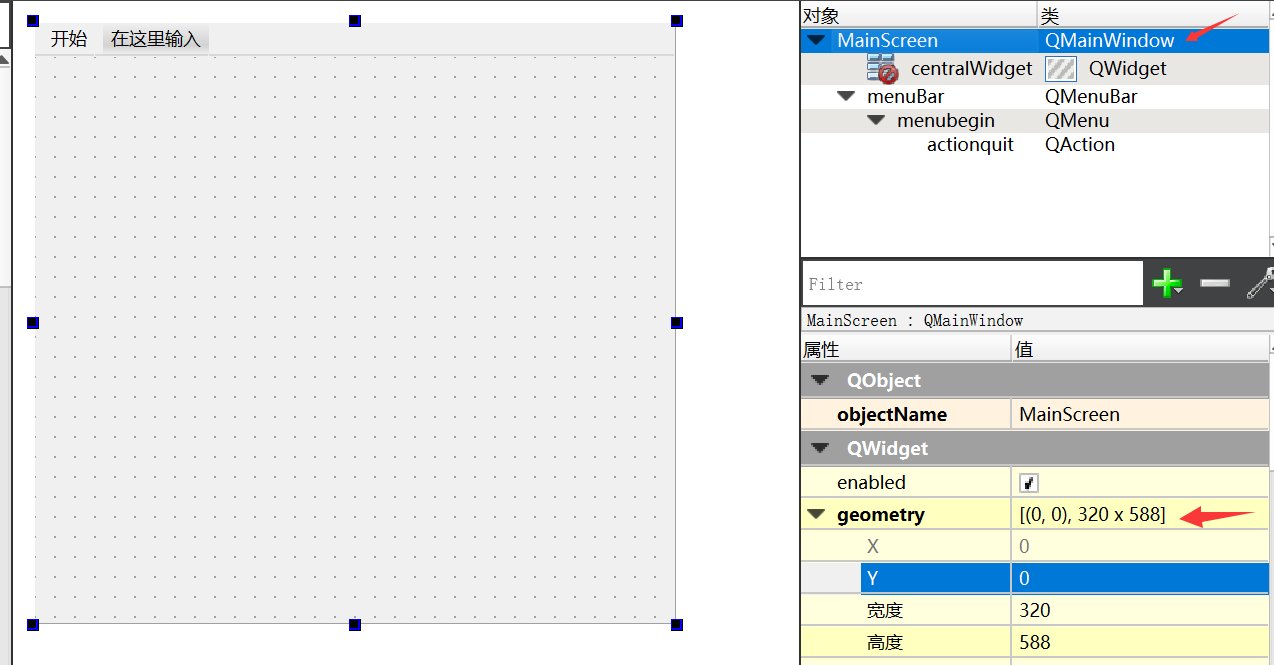
connect(ui->comboBox,fun, [=](int index){

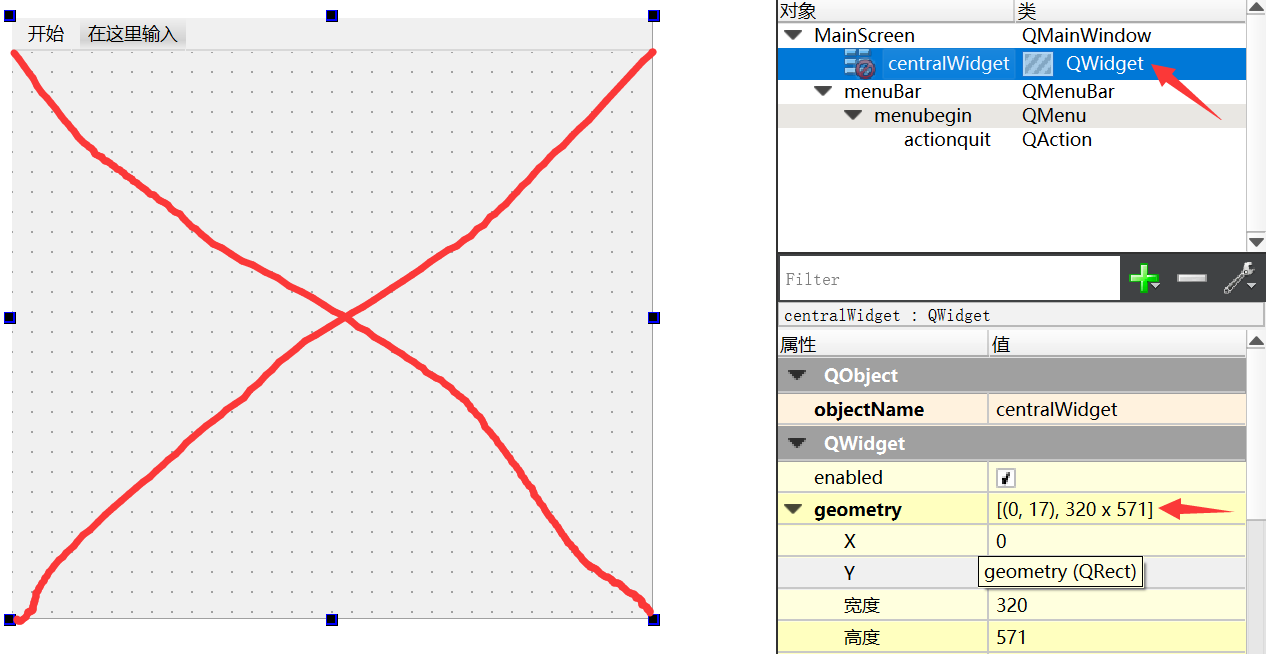
qDebug() << QString::number(index);

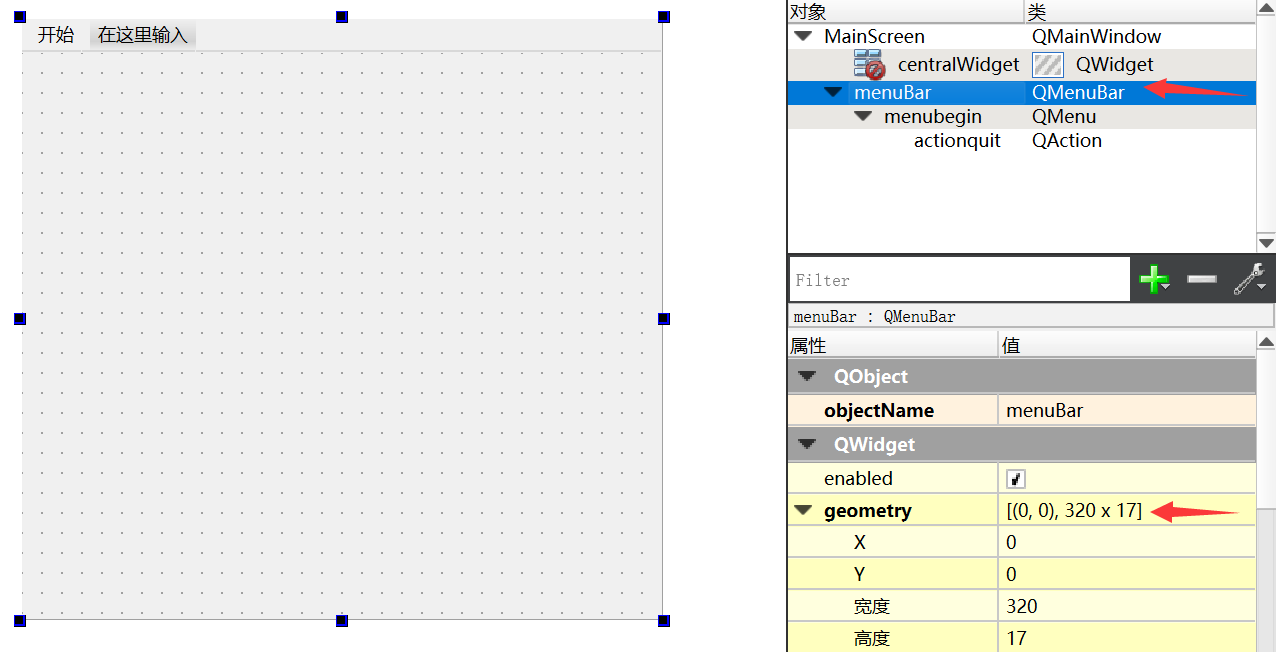
});

问题：

使用派生自QMainWindow的窗口







说明：



QFont font ( “Microsoft YaHei”, 10, 75); //第一个属性是字体（微软雅黑），第二个是大小，第三个是加粗（权重是75）

ui->label->setFont(font);

常见权重

QFont::Light - 25 高亮

QFont::Normal - 50 正常

QFont::DemiBold - 63 半粗体

QFont::Bold - 75 粗体

QFont::Black - 87 黑体

————————————————

控件样式：

//将QLabel设置自定义样式

label->setStyleSheet("QLabel{border-width: 1px 1px 1px 1px;"

"border-style: solid;"

"border-color: rgb(255,0,0);"

"background-color: rgba(213, 213, 213, 125);"

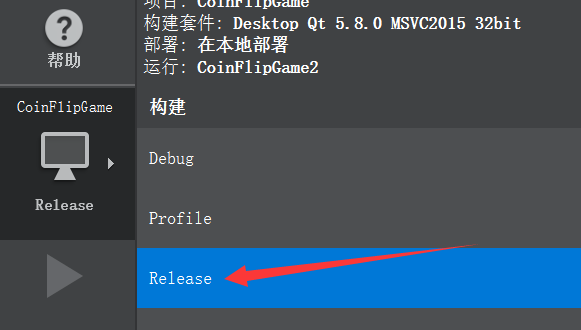
"border-radius:5px;}");

//将按钮设置为“不规则样式”

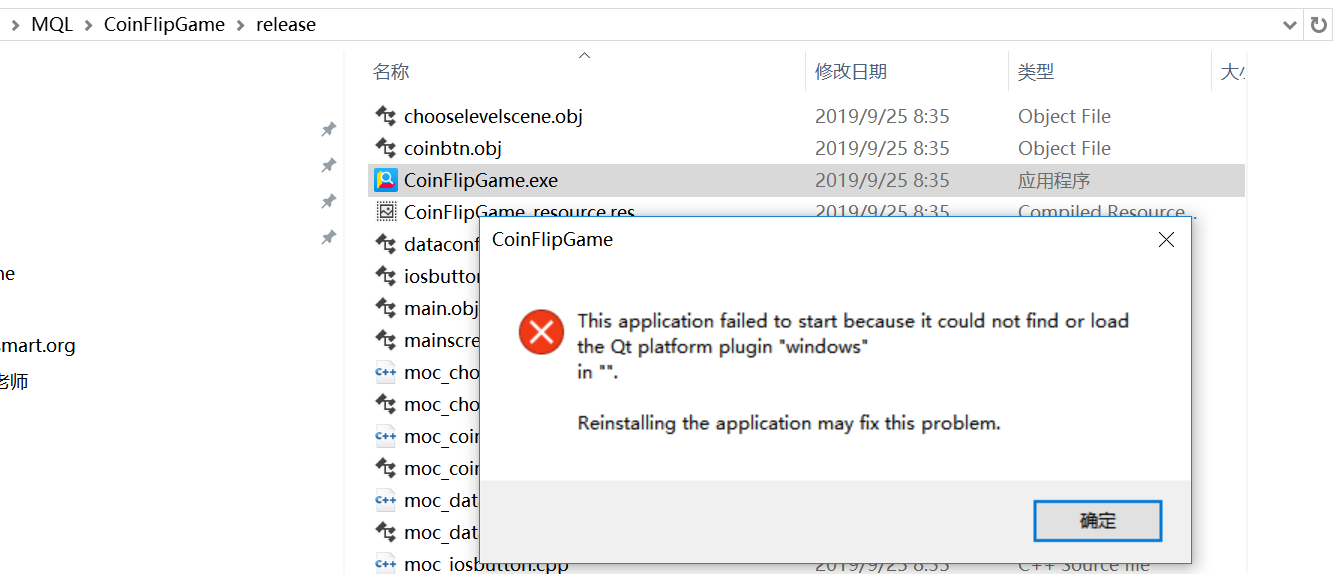
this->setStyleSheet("QPushButton{border:0px;}");

打包：

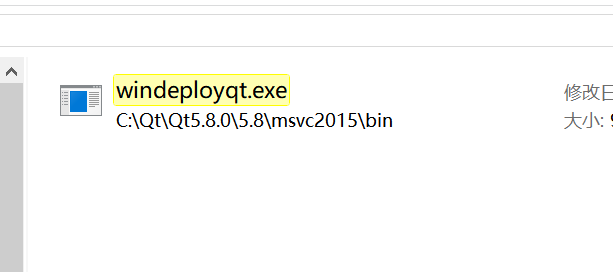
1. 构建release版本



第二，到release目录下双击exe程序，出现下面错误提示，



解决方案：



QT为了简化生成发布版本，特别提供了工具 "windeployqt.exe"，这个工具在 "...\Qt5.8.0\5.8\msvc2015\_64\bin"的目录下，通过该命令，可以解决上述错误。

1. 打开控制台窗口
2. 设置搜索路径以便系统可以搜索到 "windeployqt.exe"
3. 运行 "windeployqt.exe D:\work\project\sample\bin\Release\sample.exe", 所需要的QT运行库就自动拷贝到目标程序目录了。

* 其他

1. 以上方法也适合debug版本。
2. 如果程序使用了新的QT模块，重新执行一遍即可

eg:

C:\Users\JRWX\Desktop\release>windeployqt CoinFlipGame.exe

C:\Users\JRWX\Desktop\release\CoinFlipGame.exe 32 bit, release executable

Adding Qt5Svg for qsvgicon.dll

Skipping plugin qtvirtualkeyboardplugin.dll due to disabled dependencies.

Direct dependencies: Qt5Core Qt5Gui Qt5Multimedia Qt5Widgets

All dependencies : Qt5Core Qt5Gui Qt5Multimedia Qt5Network Qt5Widgets

To be deployed : Qt5Core Qt5Gui Qt5Multimedia Qt5Network Qt5Svg Qt5Widgets

Warning: Cannot find Visual Studio installation directory, VCINSTALLDIR is not set.

Updating Qt5Core.dll.

Updating Qt5Gui.dll.

Updating Qt5Multimedia.dll.

Updating Qt5Network.dll.

Updating Qt5Svg.dll.

Updating Qt5Widgets.dll.

Updating libGLESV2.dll.

Updating libEGL.dll.

Updating D3Dcompiler\_47.dll.

Updating opengl32sw.dll.

Patching Qt5Core.dll...

Creating directory C:/Users/JRWX/Desktop/release/audio.

Updating qtaudio\_windows.dll.

Creating directory C:/Users/JRWX/Desktop/release/bearer.

Updating qgenericbearer.dll.

Updating qnativewifibearer.dll.

Creating directory C:/Users/JRWX/Desktop/release/iconengines.

Updating qsvgicon.dll.

Creating directory C:/Users/JRWX/Desktop/release/imageformats.

Updating qgif.dll.

Updating qicns.dll.

Updating qico.dll.

Updating qjpeg.dll.

Updating qsvg.dll.

Updating qtga.dll.

Updating qtiff.dll.

Updating qwbmp.dll.

Updating qwebp.dll.

Creating directory C:/Users/JRWX/Desktop/release/mediaservice.

Updating dsengine.dll.

Updating qtmedia\_audioengine.dll.

Updating wmfengine.dll.

Creating directory C:/Users/JRWX/Desktop/release/platforms.

Updating qwindows.dll.

Creating directory C:/Users/JRWX/Desktop/release/playlistformats.

Updating qtmultimedia\_m3u.dll.

Creating C:\Users\JRWX\Desktop\release\translations...

Creating qt\_ca.qm...

Creating qt\_cs.qm...

Creating qt\_de.qm...

Creating qt\_en.qm...

Creating qt\_fi.qm...

Creating qt\_fr.qm...

Creating qt\_he.qm...

Creating qt\_hu.qm...

Creating qt\_it.qm...

Creating qt\_ja.qm...

Creating qt\_ko.qm...

Creating qt\_lv.qm...

Creating qt\_pl.qm...

Creating qt\_ru.qm...

Creating qt\_sk.qm...

Creating qt\_uk.qm...

# 手把手教NIS Edit安装向导的使用

<https://blog.csdn.net/autumn20080101/article/details/8893975>

窗口使用QVBoxLayout时，只能添加垂直方向上的弹簧，建议:弹簧-控件-弹簧，让控件始终在弹簧中间。

Widget::Widget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Widget)

{

ui->setupUi(this);

//添加垂直布局

QVBoxLayout \*layout = new QVBoxLayout();

layout->setContentsMargins(10, 10, 10, 10);

setLayout(layout);

addTopWidget();

}

void Widget::addTopWidget()

{

//创建顶部窗口

MyWidget \*widget = new MyWidget();

widget->setFixedHeight(100);

widget->setStyleSheet("background-color: rgb(255,0, 0)");

//获取当前布局

QVBoxLayout \*layout = (QVBoxLayout\*)this->layout();

//创建并添加widget top上面的弹簧

QSpacerItem \*item = new QSpacerItem(0, 0, QSizePolicy::Preferred, QSizePolicy::Fixed);

layout->addSpacerItem(item);

//将widget添加到布局

layout->addWidget(widget);

//创建并添加widget bottom上面的弹簧

item = new QSpacerItem(0, 0, QSizePolicy::Preferred, QSizePolicy::Expanding);

layout->addSpacerItem(item);

}

水平布局里的控件，当控件高度小于布局窗口高度时，垂直居中对齐；

垂直布局里的控件，水平居左对齐

通过布局嵌套，到达不同高度的顶部、底部、左边、右边对齐。